Package ‘pcds’

July 23, 2025

Type Package

Title Proximity Catch Digraphs and Their Applications

Version 0.1.8

Description Contains the functions for construction and visualization of various families of the proximity catch digraphs (PCDs), see (Ceyhan (2005) ISBN:978-3-639-19063-2), for computing the graph invariants for testing the patterns of segregation and association against complete spatial randomness (CSR) or uniformity in one, two and three dimensional cases. The package also has tools for generating points from these spatial patterns. The graph invariants used in testing spatial point data are the domination number (Ceyhan (2011) <doi:10.1080/03610921003597211>) and arc density (Ceyhan et al. (2006) <doi:10.1016/j.csda.2005.03.002>; Ceyhan et al. (2007) <doi:10.1002/cjs.5550350106>). The PCD families considered are Arc-Slice PCDs, Proportional-Edge PCDs, and Central Similarity PCDs.

License GPL-2

Encoding UTF-8

LazyData TRUE

Imports combinat, interp, gMOIP, plot3D, plotrix, Rdpack (>= 0.7)

Depends R (>= 3.5.0)

RdMacros Rdpack

Suggests knitr, scatterplot3d, spatstat.random, rmarkdown, bookdown, spelling

RoxygenNote 7.2.3

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Elvan Ceyhan [aut, cre]

Maintainer Elvan Ceyhan <`elvanceyhan@gmail.com`>

Repository CRAN

Date/Publication 2023-12-19 06:50:02 UTC

Contents

`pcds-package` *pcds: A package for Proximity Catch Digraphs and Their Applications*

Description

`pcds` is a package for construction and visualization of proximity catch digraphs (PCDs) and computation of two graph invariants of the PCDs and testing spatial patterns using these invariants.

Details

The PCD families considered are Arc-Slice (AS) PCDs, Proportional-Edge (PE) PCDs and Central Similarity (CS) PCDs.

The graph invariants used in testing spatial point data are the domination number (Ceyhan (2011)) and arc density (Ceyhan et al. (2006); Ceyhan et al. (2007)) of for two-dimensional data.

The `pcds` package also contains the functions for generating patterns of segregation, association, CSR (complete spatial randomness) and Uniform data in one, two and three dimensional cases, for testing these patterns based on two invariants of various families of the proximity catch digraphs (PCDs), (see (Ceyhan (2005)).

Moreover, the package has visualization tools for these digraphs for 1D-3D vertices. The AS-PCD and CS-PCD tools are provided for 1D and 2D data and PE-PCD related tools are provided for 1D-3D data.

The `pcds` functions

The `pcds` functions can be grouped as Auxiliary Functions, AS-PCD Functions, PE-PCD Functions, and CS-PCD Functions.

Auxiliary Functions

Contains the auxiliary (or utility) functions for constructing and visualizing Delaunay tessellations in 1D and 2D settings, computing the domination number, constructing the geometrical tools, such as equation of lines for two points, distances between lines and points, checking points inside the triangle etc., finding the (local) extrema (restricted to Delaunay cells or vertex or edge regions in them).

Arc-Slice PCD Functions

Contains the functions used in AS-PCD construction, estimation of domination number, arc density, etc in the 2D setting.

Proportional-Edge PCD Functions

Contains the functions used in PE-PCD construction, estimation of domination number, arc density, etc in the 1D-3D settings.

Central-Similarity PCD Functions

Contains the functions used in CS-PCD construction, estimation of domination number, arc density, etc in the 1D and 2D setting.

Point Generation Functions

Contains functions for generation of points from uniform (or CSR), segregation and association patterns.

In all these functions points are vectors, and data sets are either matrices or data frames.

Author(s)

Maintainer: Elvan Ceyhan `<elvanceyhan@gmail.com>`

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

`.onAttach` *.onAttach start message*

Description

.onAttach start message

Usage

```
.onAttach(libname, pkgname)
```

Arguments

`libname`	defunct
`pkgname`	defunct

Value

invisible()

`.onLoad` *.onLoad getOption package settings*

Description

.onLoad getOption package settings

Usage

```
.onLoad(libname, pkgname)
```

Arguments

`libname`	defunct
`pkgname`	defunct

Value

invisible()

Examples

```
getOption("pcds.name")
```

`angle.str2end` *The angles to draw arcs between two line segments*

Description

Gives the two pairs of angles in radians or degrees to draw arcs between two vectors or line segments for the `draw.arc` function in the `plotrix` package. The angles are provided with respect to the x-axis in the coordinate system. The line segments are $[ba]$ and $[bc]$ when the argument is given as `a,b,c` in the function.

`radian` is a logical argument (default=`TRUE`) which yields the angle in radians if `TRUE`, and in degrees if `FALSE`. The first pair of angles is for drawing arcs in the smaller angle between $[ba]$ and $[bc]$ and the second pair of angles is for drawing arcs in the counter-clockwise order from $[ba]$ to $[bc]$.

Usage

```
angle.str2end(a, b, c, radian = TRUE)
```

Arguments

`a, b, c`	Three 2D points which represent the intersecting line segments $[ba]$ and $[bc]$.
`radian`	A logical argument (default=`TRUE`). If `TRUE`, the smaller angle or counter-clockwise angle between the line segments $[ba]$ and $[bc]$ is provided in radians, else it is provided in degrees.

Value

A `list` with two elements

`small.arc.angles`	Angles of $[ba]$ and $[bc]$ with the x-axis so that difference between them is the smaller angle between $[ba]$ and $[bc]$
`ccw.arc.angles`	Angles of $[ba]$ and $[bc]$ with the x-axis so that difference between them is the counter-clockwise angle between $[ba]$ and $[bc]$

Author(s)

Elvan Ceyhan

See Also

`angle3pnts`

Examples

```
A<-c(.3,.2); B<-c(.6,.3); C<-c(1,1)

pts<-rbind(A,B,C)

Xp<-c(B[1]+max(abs(C[1]-B[1]),abs(A[1]-B[1])),0)

angle.str2end(A,B,C)
angle.str2end(A,B,A)

angle.str2end(A,B,C,radian=FALSE)

#plot of the line segments
ang.rad<-angle.str2end(A,B,C,radian=TRUE); ang.rad
ang.deg<-angle.str2end(A,B,C,radian=FALSE); ang.deg
ang.deg1<-ang.deg$s; ang.deg1
ang.deg2<-ang.deg$c; ang.deg2

rad<-min(Dist(A,B),Dist(B,C))

Xlim<-range(pts[,1],Xp[1],B+Xp,B[1]+c(+rad,-rad))
Ylim<-range(pts[,2],B[2]+c(+rad,-rad))
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot for the smaller arc
plot(pts,pch=1,asp=1,xlab="x",ylab="y",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(B,B,B); R<-rbind(A,C,B+Xp)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
plotrix::draw.arc(B[1],B[2],radius=.3*rad,angle1=ang.rad$s[1],angle2=ang.rad$s[2])
plotrix::draw.arc(B[1],B[2],radius=.6*rad,angle1=0, angle2=ang.rad$s[1],lty=2,col=2)
plotrix::draw.arc(B[1],B[2],radius=.9*rad,angle1=0,angle2=ang.rad$s[2],col=3)
txt<-rbind(A,B,C)
text(txt+cbind(rep(xd*.02,nrow(txt)),rep(-xd*.02,nrow(txt))),c("A","B","C"))

text(rbind(B)+.5*rad*c(cos(mean(ang.rad$s)),sin(mean(ang.rad$s))),
     paste(abs(round(ang.deg1[2]-ang.deg1[1],2))," degrees",sep=""))
text(rbind(B)+.6*rad*c(cos(ang.rad$s[1]/2),sin(ang.rad$s[1]/2)),paste(round(ang.deg1[1],2)),col=2)
text(rbind(B)+.9*rad*c(cos(ang.rad$s[2]/2),sin(ang.rad$s[2]/2)),paste(round(ang.deg1[2],2)),col=3)

#plot for the counter-clockwise arc
plot(pts,pch=1,asp=1,xlab="x",ylab="y",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(B,B,B); R<-rbind(A,C,B+Xp)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
```

```
plotrix::draw.arc(B[1],B[2],radius=.3*rad,angle1=ang.rad$c[1],angle2=ang.rad$c[2])
plotrix::draw.arc(B[1],B[2],radius=.6*rad,angle1=0, angle2=ang.rad$s[1],lty=2,col=2)
plotrix::draw.arc(B[1],B[2],radius=.9*rad,angle1=0,angle2=ang.rad$s[2],col=3)
txt<-pts
text(txt+cbind(rep(xd*.02,nrow(txt)),rep(-xd*.02,nrow(txt))),c("A","B","C"))

text(rbind(B)+.5*rad*c(cos(mean(ang.rad$c)),sin(mean(ang.rad$c))),
     paste(abs(round(ang.deg2[2]-ang.deg2[1],2))," degrees",sep=""))
text(rbind(B)+.6*rad*c(cos(ang.rad$s[1]/2),sin(ang.rad$s[1]/2)),paste(round(ang.deg1[1],2)),col=2)
text(rbind(B)+.9*rad*c(cos(ang.rad$s[2]/2),sin(ang.rad$s[2]/2)),paste(round(ang.deg1[2],2)),col=3)
```

`angle3pnts` *The angle between two line segments*

Description

Returns the angle in radians or degrees between two vectors or line segments at the point of intersection. The line segments are $[ba]$ and $[bc]$ when the arguments of the function are given as `a,b,c`. `radian` is a logical argument (default=`TRUE`) which yields the angle in radians if `TRUE`, and in degrees if `FALSE`. The smaller of the angle between the line segments is provided by the function.

Usage

```
angle3pnts(a, b, c, radian = TRUE)
```

Arguments

`a, b, c`	Three 2D points which represent the intersecting line segments $[ba]$ and $[bc]$. The smaller angle between these line segments is to be computed.
`radian`	A logical argument (default=`TRUE`). If `TRUE`, the (smaller) angle between the line segments $[ba]$ and $[bc]$ is provided in radians, else it is provided in degrees.

Value

angle in radians or degrees between the line segments $[ba]$ and $[bc]$

Author(s)

Elvan Ceyhan

See Also

`angle.str2end`

Examples

```
A<-c(.3,.2); B<-c(.6,.3); C<-c(1,1)
pts<-rbind(A,B,C)

angle3pnts(A,B,C)

angle3pnts(A,B,A)
round(angle3pnts(A,B,A),7)

angle3pnts(A,B,C,radian=FALSE)

#plot of the line segments
Xlim<-range(pts[,1])
Ylim<-range(pts[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

ang1<-angle3pnts(A,B,C,radian=FALSE)
ang2<-angle3pnts(B+c(1,0),B,C,radian=FALSE)

sa<-angle.str2end(A,B,C,radian=FALSE)$s #small arc angles
ang1<-sa[1]
ang2<-sa[2]

plot(pts,asp=1,pch=1,xlab="x",ylab="y",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(B,B); R<-rbind(A,C)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
plotrix::draw.arc(B[1],B[2],radius=xd*.1,deg1=ang1,deg2=ang2)
txt<-rbind(A,B,C)
text(txt+cbind(rep(xd*.05,nrow(txt)),rep(-xd*.02,nrow(txt))),c("A","B","C"))

text(rbind(B)+.15*xd*c(cos(pi*(ang2+ang1)/360),sin(pi*(ang2+ang1)/360)),
paste(round(abs(ang1-ang2),2)," degrees"))
```

`arcsAS` *The arcs of Arc Slice Proximity Catch Digraph (AS-PCD) for a 2D data set - multiple triangle case*

Description

An object of class `"PCDs"`. Returns arcs of AS-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the AS-PCD are the data points in `Xp` in the multiple triangle case.

AS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points, i.e., AS proximity regions are defined only for `Xp` points inside the convex hull of `Yp` points. That is, arcs may exist for points only inside the convex hull of `Yp` points. It also provides various descriptions and quantities about the arcs of the AS-PCD such as number of arcs, arc density, etc.

Vertex regions are based on the center `M="CC"` for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is `M="CC"` i.e., circumcenter of each triangle. `M` must be entered in barycentric coordinates unless it is the circumcenter.

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005, 2010)) for more on AS PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
arcsAS(Xp, Yp, M = "CC")
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangulation. The Delaunay triangles partition the convex hull of `Yp` points.
`M`	The center of the triangle. `"CC"` represents the circumcenter of each Delaunay triangle or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is `M="CC"` i.e., the circumcenter of each triangle. `M` must be entered in barycentric coordinates unless it is the circumcenter.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the center used to construct the vertex regions, default is circumcenter, denoted as `"CC"`, otherwise given in barycentric coordinates.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the Delaunay triangulation based on `Yp` points.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp`.
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of AS-PCD for 2D data set `Xp` in the multiple triangle case as the vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of AS-PCD for 2D data set `Xp` in the multiple triangle case as the vertices of the digraph
`mtitle`	Text for `"main"` title in the plot of the digraph

`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`arcsAStri`, `arcsPEtri`, `arcsCStri`, `arcsPE`, and `arcsCS`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-5;  #try also nx=20; nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

Arcs<-arcsAS(Xp,Yp,M) #try also the default M with Arcs<-arcsAS(Xp,Yp)
Arcs
summary(Arcs)
plot(Arcs)

arcsAS(Xp,Yp[1:3,],M)
```

`arcsAStri` *The arcs of Arc Slice Proximity Catch Digraph (AS-PCD) for 2D data - one triangle case*

Description

An object of class `"PCDs"`. Returns arcs of AS-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the AS-PCD are the data points in `Xp` in the one triangle case.

AS proximity regions are constructed with respect to the triangle `tri`, i.e., arcs may exist for points only inside `tri`. It also provides various descriptions and quantities about the arcs of the AS-PCD such as number of arcs, arc density, etc.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`. The different consideration of circumcenter vs any other interior center of the triangle is because the projections from circumcenter are orthogonal to the edges, while projections of `M` on the edges are on the extensions of the lines connecting `M` and the vertices.

See also (Ceyhan (2005, 2010)).

Usage

```
arcsAStri(Xp, tri, M = "CC")
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of `tri`.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the center used to construct the vertex regions.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support triangle `tri`.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp`.

`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of AS-PCD for 2D data set `Xp` as vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of AS-PCD for 2D data set `Xp` as vertices of the digraph
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`arcsAS`, `arcsPEtri`, `arcsCStri`, `arcsPE`, and `arcsCS`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also  M<-c(1.6,1.2) or M<-circumcenter.tri(Tr)

Arcs<-arcsAStri(Xp,Tr,M) #try also Arcs<-arcsAStri(Xp,Tr)
#uses the default center, namely circumcenter for M
Arcs
summary(Arcs)
plot(Arcs) #use plot(Arcs,asp=1) if M=CC

#can add vertex regions
#but we first need to determine center is the circumcenter or not,
```

```
#see the description for more detail.
CC<-circumcenter.tri(Tr)
M = as.numeric(Arcs$parameters[[1]])
if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-prj.cent2edges(Tr,M)
}
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

#now we add the vertex names and annotation
txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]+c(-.02,.03,.02,.03,.04,-.03,-.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.04,.05,-.07)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)
```

`arcsCS` *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 2D data - multiple triangle case*

Description

An object of class `"PCDs"`. Returns arcs of CS-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the CS-PCD are the data points in `Xp` in the multiple triangle case.

CS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $t > 0$ and edge regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (nonscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) for more on CS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
arcsCS(Xp, Yp, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle, default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the center used to construct the edge regions.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is Delaunay triangulation based on `Yp` points.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of CS-PCD for 2D data set `Xp` as vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of CS-PCD for 2D data set `Xp` as vertices of the digraph
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of triangles, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2014). “Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering.” *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`arcsCStri`, `arcsAS` and `arcsPE`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

tau<-1.5  #try also tau<-2

Arcs<-arcsCS(Xp,Yp,tau,M)
#or use the default center Arcs<-arcsCS(Xp,Yp,tau)
Arcs
summary(Arcs)
plot(Arcs)
```

`arcsCS1D` *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - multiple interval case*

Description

An object of class `"PCDs"`. Returns arcs of CS-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the CS-PCD are the 1D data points in `Xp` in the multiple interval case. `Yp` determines the end points of the intervals.

If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

For this function, CS proximity regions are constructed data points inside or outside the intervals based on `Yp` points with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$. That is, for this function, arcs may exist for points in the middle or end-intervals. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

Equivalent to function `arcsCS1D`.

See also (Ceyhan (2016)).

Usage

```
arcsCS1D(Xp, Yp, t, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the CS-PCD.
`Yp`	A set or `vector` of 1D points which constitute the end points of the intervals.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, they are expansion and centrality parameters.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the intervalization of the real line based on `Yp` points.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of CS-PCD for 1D data
`E`	Heads (or arrow ends) of the arcs of CS-PCD for 1D data
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

arcsCSend.int, arcsCSmid.int, arcsCS1D, and arcsPE1D

Examples

```
t<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Arcs<-arcsCS1D(Xp,Yp,t,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

arcsCS1D(Xp,Yp,t,c)

arcsCS1D(Xp,Yp+10,t,c)

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),
main="arcs of CS-PCD for points (jittered along y-axis)\n in middle intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)

t<-2
c<-.4
a<-0; b<-10;
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)
```

```
arcsCS1D(Xp,Yp,t,c)
```

`arcsCSend.int` *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - end-interval case*

Description

An object of class `"PCDs"`. Returns arcs of CS-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the CS-PCD are the 1D data points in `Xp` in the end-interval case. `Yp` determines the end points of the end-intervals.

For this function, CS proximity regions are constructed data points outside the intervals based on `Yp` points with expansion parameter $t > 0$. That is, for this function, arcs may exist for points only inside end-intervals. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

See also (Ceyhan (2016)).

Usage

```
arcsCSend.int(Xp, Yp, t)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the CS-PCD.
`Yp`	A set or `vector` of 1D points which constitute the end points of the intervals.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the expansion parameter.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the intervalization based on `Yp`.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitutes the vertices of the digraph
`S`	Tails (or sources) of the arcs of CS-PCD for 1D data in the end-intervals
`E`	Heads (or arrow ends) of the arcs of CS-PCD for 1D data in the end-intervals
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals (which is 2 for end-intervals), number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`arcsCSmid.int`, `arcsCS1D` , `arcsPEmid.int`, `arcsPEend.int` and `arcsPE1D`

Examples

```
t<-1.5
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.5

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

arcsCSend.int(Xp,Yp,t)

Arcs<-arcsCSend.int(Xp,Yp,t)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),pch=".",
main="arcs of CS-PCD with vertices (jittered along y-axis)\n in end-intervals ",
     xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit))
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)

arcsCSend.int(Xp,Yp,t)
```

`arcsCSint` *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - one interval case*

Description

An object of class `"PCDs"`. Returns arcs of CS-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the CS-PCD are the 1D data points in `Xp` in the one interval case. `int` determines the end points of the interval.

For this function, CS proximity regions are constructed data points inside or outside the interval based on `int` points with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$. That is, for this function, arcs may exist for points in the middle or end-intervals. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

Usage

```
arcsCSint(Xp, int, t, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the CS-PCD.
`int`	A `vector` of two 1D points which constitutes the end points of the interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, they are expansion and centrality parameters.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the intervalization of the real line based on `int` points.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of CS-PCD for 1D data
`E`	Heads (or arrow ends) of the arcs of CS-PCD for 1D data
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

`arcsCS1D`, `arcsCSmid.int`, `arcsCSend.int`, and `arcsPE1D`

Examples

```
tau<-2
c<-.4
a<-0; b<-10; int<-c(a,b);

#n is number of X points
n<-10; #try also n<-20

xf<-(int[2]-int[1])*.1

set.seed(1)
Xp<-runif(n,a-xf,b+xf)

Arcs<-arcsCSint(Xp,int,tau,c)
Arcs
summary(Arcs)
plot(Arcs)

Xp<-runif(n,a+10,b+10)
Arcs=arcsCSint(Xp,int,tau,c)
Arcs
summary(Arcs)
plot(Arcs)
```

`arcsCSmid.int` *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - middle intervals case*

Description

An object of class `"PCDs"`. Returns arcs of CS-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the CS-PCD are the 1D data points in `Xp` in the middle interval case.

For this function, CS proximity regions are constructed with respect to the intervals based on `Yp` points with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$. That is, for this function, arcs may exist for points only inside the intervals. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

Vertex regions are based on center M_c of each middle interval.

See also (Ceyhan (2016)).

Usage

```
arcsCSmid.int(Xp, Yp, t, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the CS-PCD.
`Yp`	A set or `vector` of 1D points which constitute the end points of the intervals.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, they are expansion and centrality parameters.
`tess.points`	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization based on `Yp` points.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, i.e., `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of CS-PCD for 1D data in the middle intervals
`E`	Heads (or arrow ends) of the arcs of CS-PCD for 1D data in the middle intervals
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`arcsPEend.int`, `arcsPE1D`, `arcsCSmid.int`, `arcsCSend.int` and `arcsCS1D`

Examples

```
t<-1.5
c<-.4
a<-0; b<-10

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

arcsCSmid.int(Xp,Yp,t,c)
arcsCSmid.int(Xp,Yp+10,t,c)

Arcs<-arcsCSmid.int(Xp,Yp,t,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),
main="arcs of CS-PCD whose vertices (jittered along y-axis)\n in middle intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)

t<-.5
c<-.4
a<-0; b<-10;
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

arcsCSmid.int(Xp,Yp,t,c)
```

`arcsCStri` *The arcs of Central Similarity Proximity Catch Digraphs (CS-PCD) for 2D data - one triangle case*

Description

An object of class "PCDs". Returns arcs of CS-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the CS-PCD are the data points in `Xp` in the one triangle case.

CS proximity regions are constructed with respect to the triangle `tri` with expansion parameter $t > 0$, i.e., arcs may exist for points only inside `tri`. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

Edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
arcsCStri(Xp, tri, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, the center `M` used to construct the edge regions and the expansion parameter `t`.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support triangle `tri`.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of CS-PCD for 2D data set `Xp` as vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of CS-PCD for 2D data set `Xp` as vertices of the digraph
`mtitle`	Text for "main" title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of triangles, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`arcsCS`, `arcsAStri` and `arcsPEtri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

t<-1.5  #try also t<-2

Arcs<-arcsCStri(Xp,Tr,t,M)
#or try with the default center Arcs<-arcsCStri(Xp,Tr,t); M= (Arcs$param)$c
Arcs
summary(Arcs)
plot(Arcs)

#can add edge regions
L<-rbind(M,M,M); R<-Tr
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

#now we can add the vertex names and annotation
txt<-rbind(Tr,M)
xc<-txt[,1]+c(-.02,.03,.02,.03)
yc<-txt[,2]+c(.02,.02,.03,.06)
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)
```

`arcsPE` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data - multiple triangle case*

Description

An object of class `"PCDs"`. Returns arcs of PE-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the PE-PCD are the data points in `Xp` in the multiple triangle case.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (nonscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)) for more on the PE-PCDs. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
arcsPE(Xp, Yp, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this, argument should be set as `M="CC"`), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, the center used to construct the vertex regions and the expansion parameter.

`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the Delaunay triangulation based on `Yp` points.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 2D data set `Xp` as vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 2D data set `Xp` as vertices of the digraph
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of triangles, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`arcsPEtri`, `arcsAS`, and `arcsCS`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
```

```
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

r<-1.5  #try also r<-2

Arcs<-arcsPE(Xp,Yp,r,M)
#or try with the default center Arcs<-arcsPE(Xp,Yp,r)
Arcs
summary(Arcs)
plot(Arcs)
```

`arcsPE1D` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - multiple interval case*

Description

An object of class `"PCDs"`. Returns arcs of PE-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the PE-PCD are the 1D data points in `Xp` in the multiple interval case. `Yp` determines the end points of the intervals.

If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

For this function, PE proximity regions are constructed data points inside or outside the intervals based on `Yp` points with expansion parameter $r \geq 1$ and centrality parameter $c \in (0,1)$. That is, for this function, arcs may exist for points in the middle or end-intervals. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

See also (Ceyhan (2012)).

Usage

```
arcsPE1D(Xp, Yp, r, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set or `vector` of 1D points which constitute the end points of the intervals.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0,1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, (a,b), the parameterized center is $M_c = a + c(b-a)$.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, they are expansion and centrality parameters.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the intervalization of the real line based on `Yp` points.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 1D data
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 1D data
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`arcsPEint`, `arcsPEmid.int`, `arcsPEend.int`, and `arcsCS1D`

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b);

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Arcs<-arcsPE1D(Xp,Yp,r,c)
Arcs
summary(Arcs)
```

```
plot(Arcs)
```

`arcsPEend.int` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - end-interval case*

Description

An object of class `"PCDs"`. Returns arcs of PE-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the PE-PCD are the 1D data points in `Xp` in the end-interval case. `Yp` determines the end points of the end-intervals.

For this function, PE proximity regions are constructed data points outside the intervals based on `Yp` points with expansion parameter $r \geq 1$. That is, for this function, arcs may exist for points only inside end-intervals. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

See also (Ceyhan (2012)).

Usage

```
arcsPEend.int(Xp, Yp, r)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set or `vector` of 1D points which constitute the end points of the intervals.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the expansion parameter.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the intervalization based on `Yp`.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitutes the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 1D data in the end-intervals
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 1D data in the end-intervals
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals (which is 2 for end-intervals), number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`arcsPEmid.int`, `arcsPE1D` , `arcsCSmid.int`, `arcsCSend.int` and `arcsCS1D`

Examples

```
r<-2
a<-0; b<-10; int<-c(a,b);

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.5

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)  #try also Yp<-runif(ny,a,b)+c(-10,10)

Arcs<-arcsPEend.int(Xp,Yp,r)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),pch=".",
main="arcs of PE-PCDs for points (jittered along y-axis)\n in end-intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit))
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)
```

`arcsPEint` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - one interval case*

Description

An object of class `"PCDs"`. Returns arcs of PE-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the PE-PCD are the 1D data points in `Xp` in the one interval case. `int` determines the end points of the interval.

For this function, PE proximity regions are constructed data points inside or outside the interval based on `int` points with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$. That is, for this function, arcs may exist for points in the middle or end-intervals. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

See also (Ceyhan (2012)).

Usage

```
arcsPEint(Xp, int, r, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the PE-PCD.
`int`	A `vector` of two 1D points which constitutes the end points of the interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, they are expansion and centrality parameters.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the end points of the support interval `int`.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 1D data
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 1D data
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`arcsPE1D`, `arcsPEmid.int`, `arcsPEend.int`, and `arcsCS1D`

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b);

#n is number of X points
n<-10; #try also n<-20

xf<-(int[2]-int[1])*.1

set.seed(1)
Xp<-runif(n,a-xf,b+xf)

Arcs<-arcsPEint(Xp,int,r,c)
Arcs
summary(Arcs)
plot(Arcs)
```

`arcsPEmid.int` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - middle intervals case*

Description

An object of class `"PCDs"`. Returns arcs of PE-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the PE-PCD are the 1D data points in `Xp` in the middle interval case.

For this function, PE proximity regions are constructed with respect to the intervals based on `Yp` points with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$. That is, for this function, arcs may exist for points only inside the intervals. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

Vertex regions are based on center M_c of each middle interval. If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

See also (Ceyhan (2012)).

Usage

```
arcsPEmid.int(Xp, Yp, r, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set or `vector` of 1D points which constitute the end points of the intervals.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, they are expansion and centrality parameters.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the intervalization based on `Yp` points.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, i.e., `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 1D data in the middle intervals
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 1D data in the middle intervals
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`arcsPEend.int`, `arcsPE1D`, `arcsCSmid.int`, `arcsCSend.int` and `arcsCS1D`

Examples

```
r<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

Arcs<-arcsPEmid.int(Xp,Yp,r,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

arcsPEmid.int(Xp,Yp,r,c)
arcsPEmid.int(Xp,Yp+10,r,c)

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),
main="arcs of PE-PCD for points (jittered along y-axis)\n in middle intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)
```

`arcsPEtri` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data - one triangle case*

Description

An object of class "PCDs". Returns arcs of PE-PCD as tails (or sources) and heads (or arrow ends) and related parameters and the quantities of the digraph. The vertices of the PE-PCD are the data points in `Xp` in the one triangle case.

PE proximity regions are constructed with respect to the triangle `tri` with expansion parameter $r \geq 1$, i.e., arcs may exist only for points inside `tri`. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`. When the center is the circumcenter, `CC`, the vertex regions are constructed based on the orthogonal projections to the edges, while with any interior center `M`, the vertex regions are constructed using the extensions of the lines combining vertices with `M`. `M`-vertex regions are recommended spatial inference, due to geometry invariance property of the arc density and domination number the PE-PCDs based on uniform data.

See also (Ceyhan (2005); Ceyhan et al. (2006)).

Usage

```
arcsPEtri(Xp, tri, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.

Value

A `list` with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, the center `M` used to construct the vertex regions and the expansion parameter `r`.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support triangle `tri`.
`tess.name`	Name of the tessellation points `tess.points`
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitutes the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 2D data set `Xp` as vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 2D data set `Xp` as vertices of the digraph
`mtitle`	Text for `"main"` title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of triangles, number of arcs, and arc density.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`arcsPE`, `arcsAStri`, and `arcsCStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5  #try also r<-2

Arcs<-arcsPEtri(Xp,Tr,r,M)
#or try with the default center Arcs<-arcsPEtri(Xp,Tr,r); M= (Arcs$param)$cent
Arcs
summary(Arcs)
plot(Arcs)

#can add vertex regions
#but we first need to determine center is the circumcenter or not,
#see the description for more detail.
CC<-circumcenter.tri(Tr)
if (isTRUE(all.equal(M,CC)))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-prj.cent2edges(Tr,M)
}
L<-rbind(cent,cent,cent); R<-Ds
```

```
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

#now we can add the vertex names and annotation
txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]+c(-.02,.02,.02,.02,.03,-.03,-.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.04,.05,-.07)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)
```

`area.polygon` *The area of a polygon in R^2*

Description

Returns the area of the polygon, `h`, in the real plane R^2; the vertices of the polygon `h` must be provided in clockwise or counter-clockwise order, otherwise the function does not yield the area of the polygon. Also, the polygon could be convex or non-convex. See (Weisstein (2019)).

Usage

```
area.polygon(h)
```

Arguments

`h`	A `vector` of n 2D points, stacked row-wise, each row representing a vertex of the polygon, where n is the number of vertices of the polygon.

Value

area of the polygon `h`

Author(s)

Elvan Ceyhan

References

Weisstein EW (2019). "Polygon Area." From MathWorld — A Wolfram Web Resource, http://mathworld.wolfram.com/PolygonArea.html.

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(0.5,.8);
Tr<-rbind(A,B,C);
area.polygon(Tr)

A<-c(0,0); B<-c(1,0); C<-c(.7,.6); D<-c(0.3,.8);
h1<-rbind(A,B,C,D);
#try also h1<-rbind(A,B,D,C) or h1<-rbind(A,C,B,D) or h1<-rbind(A,D,C,B);
area.polygon(h1)

Xlim<-range(h1[,1])
Ylim<-range(h1[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(h1,xlab="",ylab="",main="A Convex Polygon with Four Vertices",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(h1)
xc<-rbind(A,B,C,D)[,1]+c(-.03,.03,.02,-.01)
yc<-rbind(A,B,C,D)[,2]+c(.02,.02,.02,.03)
txt.str<-c("A","B","C","D")
text(xc,yc,txt.str)

#when the triangle is degenerate, it gives zero area
B<-A+2*(C-A);
T2<-rbind(A,B,C)
area.polygon(T2)
```

`as.basic.tri` *The labels of the vertices of a triangle in the basic triangle form*

Description

Labels the vertices of triangle, `tri`, as ABC so that AB is the longest edge, BC is the second longest and AC is the shortest edge (the order is as in the basic triangle).

The standard basic triangle form is $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2+c_2^2 \leq 1$. Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle is useful for simulation studies under the uniformity hypothesis.

The option `scaled` a logical argument for scaling the resulting triangle or not. If `scaled=TRUE`, then the resulting triangle is scaled to be a regular basic triangle, i.e., longest edge having unit length, else (i.e., if `scaled=FALSE` which is the default), the new triangle $T(A,B,C)$ is nonscaled, i.e., the longest edge AB may not be of unit length. The vertices of the resulting triangle (whether scaled or not) is presented in the order of vertices of the corresponding basic triangle, however when scaled the triangle is equivalent to the basic triangle T_b up to translation and rotation. That is, this function converts any triangle to a basic triangle (up to translation and rotation), so that the output triangle is $T(A',B',C')$ so that edges in decreasing length are $A'B'$, $B'C'$, and $A'C'$. Most of the times, the resulting triangle will still need to be translated and/or rotated to be in the standard basic triangle form.

Usage

```
as.basic.tri(tri, scaled = FALSE)
```

Arguments

`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`scaled`	A logical argument for scaling the resulting basic triangle. If `scaled=TRUE`, then the resulting triangle is scaled to be a regular basic triangle, i.e., longest edge having unit length, else the new triangle $T(A,B,C)$ is nonscaled. The default is `scaled=FALSE`.

Value

A `list` with three elements

`tri`	The vertices of the basic triangle stacked row-wise and labeled row-wise as A, B, C.
`desc`	Description of the edges based on the vertices, i.e., `"Edges (in decreasing length are) AB, BC, and AC"`.
`orig.order`	Row order of the input triangle, `tri`, when converted to the scaled version of the basic triangle

Author(s)

Elvan Ceyhan

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);

as.basic.tri(rbind(A,B,C))
as.basic.tri(rbind(B,C,A))

A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
as.basic.tri(rbind(A,B,C))
as.basic.tri(rbind(A,C,B))
as.basic.tri(rbind(B,A,C))
```

`ASarc.dens.tri` *Arc density of Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns the arc density of AS-PCD whose vertex set is the given 2D numerical data set, `Xp`, (some of its members are) in the triangle `tri`.

AS proximity regions are defined with respect to `tri` and vertex regions are defined with the center `M="CC"` for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M="CC"` i.e., circumcenter of `tri`. For the number of arcs, loops are not allowed so arcs are only possible for points inside `tri` for this function.

`in.tri.only` is a logical argument (default is `FALSE`) for considering only the points inside the triangle or all the points as the vertices of the digraph. if `in.tri.only=TRUE`, arc density is computed only for the points inside the triangle (i.e., arc density of the subdigraph induced by the vertices in the triangle is computed), otherwise arc density of the entire digraph (i.e., digraph with all the vertices) is computed.

See also (Ceyhan (2005, 2010)).

Usage

```
ASarc.dens.tri(Xp, tri, M = "CC", in.tri.only = FALSE)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M="CC"` i.e., the circumcenter of `tri`.
`in.tri.only`	A logical argument (default is `in.tri.only=FALSE`) for computing the arc density for only the points inside the triangle, `tri`. That is, if `in.tri.only=TRUE` arc density of the induced subdigraph with the vertices inside `tri` is computed, otherwise otherwise arc density of the entire digraph (i.e., digraph with all the vertices) is computed.

Value

Arc density of AS-PCD whose vertices are the 2D numerical data set, `Xp`; AS proximity regions are defined with respect to the triangle `tri` and CC-vertex regions.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`ASarc.dens.tri`, `CSarc.dens.tri`, and `num.arcsAStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

set.seed(1)
n<-10  #try also n<-20

Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

narcs = num.arcsAStri(Xp,Tr,M)$num.arcs; narcs/(n*(n-1))
ASarc.dens.tri(Xp,Tr,M)
ASarc.dens.tri(Xp,Tr,M,in.tri.only = FALSE)

ASarc.dens.tri(Xp,Tr,M)
```

`center.nondegPE` *Centers for non-degenerate asymptotic distribution of domination number of Proportional Edge Proximity Catch Digraphs (PE-PCDs)*

Description

Returns the centers which yield nondegenerate asymptotic distribution for the domination number of PE-PCD for uniform data in a triangle, `tri`$= T(v_1, v_2, v_3)$.

PE proximity region is defined with respect to the triangle `tri` with expansion parameter `r` in $(1, 1.5]$.

Vertex regions are defined with the centers that are output of this function. Centers are stacked row-wise with row number is corresponding to the vertex row number in `tri` (see the examples for an illustration). The center labels 1,2,3 correspond to the vertices M_1, M_2, and M_3 (which are the three centers for `r` in $(1, 1.5)$ which becomes center of mass for $r = 1.5$.).

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
center.nondegPE(tri, r)
```

Arguments

`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ for this function.

Value

The centers (stacked row-wise) which give nondegenerate asymptotic distribution for the domination number of PE-PCD for uniform data in a triangle, `tri`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
r<-1.35

Ms<-center.nondegPE(Tr,r)
Ms

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",
main="Centers of nondegeneracy\n for the PE-PCD in a triangle",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Ms,pch=".",col=1)
polygon(Ms,lty = 2)

xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.03)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

xc<-Ms[,1]+c(-.04,.04,.03)
yc<-Ms[,2]+c(.02,.02,.05)
txt.str<-c("M1","M2","M3")
text(xc,yc,txt.str)
```

`centerMc` *Parameterized center of an interval*

Description

Returns the (parameterized) center, M_c, of the interval, `int`$=(a,b)$, parameterized by $c \in (0,1)$ so that $100c$ % of the length of interval is to the left of M_c and $100(1-c)$ % of the length of the interval is to the right of M_c. That is, for the interval, `int`$=(a,b)$, the parameterized center is $M_c = a + c(b-a)$.

See also (Ceyhan (2012, 2016)).

Usage

```
centerMc(int, c = 0.5)
```

Arguments

`int`	A `vector` with two entries representing an interval.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

(parameterized) center inside `int`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`centersMc`

Examples

```
c<-.4
a<-0; b<-10
int = c(a,b)
centerMc(int,c)

c<-.3
a<-2; b<-4; int<-c(a,b)
centerMc(int,c)
```

`centersMc` *Parameterized centers of intervals*

Description

Returns the centers of the intervals based on 1D points in `Yp` parameterized by $c \in (0, 1)$ so that $100c$ % of the length of interval is to the left of M_c and $100(1 - c)$ % of the length of the interval is to the right of M_c. That is, for an interval (a, b), the parameterized center is $M_c = a + c(b - a)$ `Yp` is a `vector` of 1D points, not necessarily sorted.

See also (Ceyhan (2012, 2016)).

Usage

```
centersMc(Yp, c = 0.5)
```

Arguments

`Yp`	A `vector` real numbers that constitute the end points of intervals.
`c`	A positive real number in $(0, 1)$ parameterizing the centers inside the intervals with the default `c=.5`. For the interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

(parameterized) centers of the intervals based on `Yp` points as a vector

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`centerMc`

Examples

```
n<-10
c<-.4  #try also c<-runif(1)
Yp<-runif(n)
centersMc(Yp,c)

c<-.3  #try also c<-runif(1)
Yp<-runif(n,0,10)
centersMc(Yp,c)
```

`circumcenter.basic.tri` *Circumcenter of a standard basic triangle form*

Description

Returns the circumcenter of a standard basic triangle form $T_b = T((0,0),(1,0),(c_1,c_2))$ given c_1, c_2 where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \le 1$.

Any given triangle can be mapped to the standard basic triangle form by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle form is useful for simulation studies under the uniformity hypothesis.

See (Weisstein (2019); Ceyhan (2010)) for triangle centers and (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for the standard basic triangle form.

Usage

```
circumcenter.basic.tri(c1, c2)
```

Arguments

`c1, c2`	Positive real numbers representing the top vertex in standard basic triangle form $T_b = T((0,0),(1,0),(c_1,c_2))$, c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \le 1$.

Value

circumcenter of the standard basic triangle form $T_b = T((0,0),(1,0),(c_1,c_2))$ given c_1, c_2 as the arguments of the function.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics*

& Data Analysis, **50(8)**, 1925-1964.

Weisstein EW (2019). “Triangle Centers.” From MathWorld — A Wolfram Web Resource, http://mathworld.wolfram.com/TriangleCenter.html.

See Also

circumcenter.tri

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
#the vertices of the standard basic triangle form Tb
Tb<-rbind(A,B,C)
CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter
CC

D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1])
Ylim<-range(Tb[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

oldpar <- par(pty = "s")
plot(A,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(rbind(CC))
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.04,.03,.06,.06,-.03,0)
yc<-txt[,2]+c(.02,.02,.03,-.03,.02,.04,-.03)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

#for an obtuse triangle
c1<-.4; c2<-.3;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
#the vertices of the standard basic triangle form Tb
Tb<-rbind(A,B,C)
CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter
CC

D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1],CC[1])
Ylim<-range(Tb[,2],CC[2])
```

```
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(rbind(CC))
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.03,.03,.07,.07,-.05,0)
yc<-txt[,2]+c(.02,.02,.04,-.03,.03,.04,.06)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
par(oldpar)
```

`circumcenter.tetra` *Circumcenter of a general tetrahedron*

Description

Returns the circumcenter a given tetrahedron `th` with vertices stacked row-wise.

Usage

```
circumcenter.tetra(th)
```

Arguments

`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.

Value

circumcenter of the tetrahedron `th`

Author(s)

Elvan Ceyhan

See Also

`circumcenter.tri`

Examples

```
set.seed(123)
A<-c(0,0,0)+runif(3,-.2,.2);
B<-c(1,0,0)+runif(3,-.2,.2);
C<-c(1/2,sqrt(3)/2,0)+runif(3,-.2,.2);
D<-c(1/2,sqrt(3)/6,sqrt(6)/3)+runif(3,-.2,.2);
tetra<-rbind(A,B,C,D)

CC<-circumcenter.tetra(tetra)
CC

Xlim<-range(tetra[,1],CC[1])
Ylim<-range(tetra[,2],CC[2])
Zlim<-range(tetra[,3],CC[3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(tetra[,1],tetra[,2],tetra[,3], phi =0,theta=40, bty = "g",
main="Illustration of the Circumcenter\n in a Tetrahedron",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(CC[1],CC[2],CC[3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3],
labels=c("A","B","C","D"), add=TRUE)

D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-matrix(rep(CC,6),byrow = TRUE,ncol=3)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty = 2)

plot3D::text3D(CC[1],CC[2],CC[3], labels="CC", add=TRUE)
```

`circumcenter.tri` *Circumcenter of a general triangle*

Description

Returns the circumcenter a given triangle, `tri`, with vertices stacked row-wise. See (Weisstein (2019); Ceyhan (2010)) for triangle centers.

Usage

```
circumcenter.tri(tri)
```

Arguments

`tri`	A 3 × 2 matrix with each row representing a vertex of the triangle.

Value

circumcenter of the triangle `tri`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Weisstein EW (2019). "Triangle Centers." From MathWorld — A Wolfram Web Resource, `http://mathworld.wolfram.com/TriangleCenter.html`.

See Also

`circumcenter.basic.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);  #the vertices of the triangle Tr

CC<-circumcenter.tri(Tr)  #the circumcenter
CC

D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],CC[1])
Ylim<-range(Tr[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,asp=1,pch=".",xlab="",ylab="",main="Circumcenter of a triangle",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(rbind(CC))
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.08,.08,.08,.12,-.09,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.03,-.06,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
```

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);  #the vertices of the equilateral triangle Te
circumcenter.tri(Te)  #the circumcenter

A<-c(0,0); B<-c(0,1); C<-c(2,0);
Tr<-rbind(A,B,C);  #the vertices of the triangle T
circumcenter.tri(Tr)  #the circumcenter
```

`cl2CCvert.reg` *The closest points to circumcenter in each CC-vertex region in a triangle*

Description

An object of class `"Extrema"`. Returns the closest data points among the data set, `Xp`, to circumcenter, CC, in each CC-vertex region in the triangle `tri` $= T(A, B, C) =$(vertex 1,vertex 2,vertex 3).

`ch.all.intri` is for checking whether all data points are inside `tri` (default is `FALSE`). If some of the data points are not inside `tri` and `ch.all.intri=TRUE`, then the function yields an error message. If some of the data points are not inside `tri` and `ch.all.intri=FALSE`, then the function yields the closest points to CC among the data points in each CC-vertex region of `tri` (yields `NA` if there are no data points inside `tri`).

See also (Ceyhan (2005, 2012)).

Usage

```
cl2CCvert.reg(Xp, tri, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`ch.all.intri`	A logical argument (default=`FALSE`) to check whether all data points are inside the triangle `tri`. So, if it is `TRUE`, the function checks if all data points are inside the closure of the triangle (i.e., interior and boundary combined) else it does not.

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances from closest points to CC ..."`

`type`	Type of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points to CC in each CC-vertex region
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is `tri`
`cent`	The center point used for construction of vertex regions
`ncent`	Name of the center, `cent`, it is `"CC"` for this function
`regions`	Vertex regions inside the triangle, `tri`, provided as a `list`
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside `tri`
`dist2ref`	Distances from closest points in each CC-vertex region to CC.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`cl2CCvert.reg.basic.tri`, `cl2edges.vert.reg.basic.tri`, `cl2edgesMvert.reg`, `cl2edgesCMvert.reg`, and `fr2edgesCMedge.reg.std.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.tri(n,Tr)$g

Ext<-cl2CCvert.reg(Xp,Tr)
Ext
summary(Ext)
plot(Ext)

c2CC<-Ext
```

```
CC<-circumcenter.tri(Tr)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",
main="Closest Points in CC-Vertex Regions \n to the Circumcenter",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(c2CC$ext,pch=4,col=2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.07,.08,.06,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.03,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

Xp2<-rbind(Xp,c(.2,.4))
cl2CCvert.reg(Xp2,Tr,ch.all.intri = FALSE)
#gives an error message if ch.all.intri = TRUE since not all points are in the triangle
```

---

cl2CCvert.reg.basic.tri

The closest points to circumcenter in each CC-vertex region in a standard basic triangle

---

Description

An object of class "Extrema". Returns the closest data points among the data set, `Xp`, to circumcenter, CC, in each CC-vertex region in the standard basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2))$ =(vertex 1,vertex 2,vertex 3). `ch.all.intri` is for checking whether all data points are inside T_b (default is `FALSE`).

See also (Ceyhan (2005, 2012)).

Usage

```
cl2CCvert.reg.basic.tri(Xp, c1, c2, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle. adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2+c_2^2 \leq 1$
`ch.all.intri`	A logical argument for checking whether all data points are inside T_b (default is `FALSE`).

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances from closest points to ..."`.
`type`	Type of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points to CC in each vertex region.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is T_b.
`cent`	The center point used for construction of vertex regions
`ncent`	Name of the center, `cent`, it is `"CC"` for this function.
`regions`	Vertex regions inside the triangle, T_b, provided as a `list`.
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside T_b.
`dist2ref`	Distances from closest points in each vertex region to CC.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`cl2CCvert.reg`, `cl2edges.vert.reg.basic.tri`, `cl2edgesMvert.reg`, `cl2edgesCMvert.reg`, and `fr2edgesCMedge.reg.std.tri`

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-15

set.seed(1)
Xp<-runif.basic.tri(n,c1,c2)$g

Ext<-cl2CCvert.reg.basic.tri(Xp,c1,c2)
Ext
summary(Ext)
plot(Ext)

c2CC<-Ext

CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",
main="Closest Points in CC-Vertex Regions \n to the Circumcenter",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(Xp)
points(c2CC$ext,pch=4,col=2)

txt<-rbind(Tb,CC,Ds)
xc<-txt[,1]+c(-.03,.03,.02,.07,.06,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,-.01,.03,.03,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

Xp2<-rbind(Xp,c(.2,.4))
cl2CCvert.reg.basic.tri(Xp2,c1,c2,ch.all.intri = FALSE)
#gives an error message if ch.all.intri = TRUE
#since not all points are in the standard basic triangle
```

`cl2edges.std.tri` *The closest points in a data set to edges in the standard equilateral triangle*

Description

An object of class `"Extrema"`. Returns the closest points from the 2D data set, `Xp`, to the edges in the standard equilateral triangle $T_e = T(A = (0, 0), B = (1, 0), C = (1/2, \sqrt{3}/2))$.

`ch.all.intri` is for checking whether all data points are inside T_e (default is `FALSE`).

If some of the data points are not inside T_e and `ch.all.intri=TRUE`, then the function yields an error message. If some of the data points are not inside T_e and `ch.all.intri=FALSE`, then the function yields the closest points to edges among the data points inside T_e (yields `NA` if there are no data points inside T_e).

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan and Priebe (2007)).

Usage

```
cl2edges.std.tri(Xp, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`ch.all.intri`	A logical argument (default=`FALSE`) to check whether all data points are inside the standard equilateral triangle T_e. So, if it is `TRUE`, the function checks if all data points are inside the closure of the triangle (i.e., interior and boundary combined) else it does not.

Value

A `list` with the elements

`txt1`	Edge labels as $AB = 3$, $BC = 1$, and $AC = 2$ for T_e (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances to Edges ..."`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, i.e., closest points to edges
`X`	The input data, `Xp`, which can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, i.e., the standard equilateral triangle T_e
`cent`	The center point used for construction of edge regions, not required for this extrema, hence it is `NULL` for this function

`ncent`	Name of the center, `cent`, not required for this extrema, hence it is `NULL` for this function
`regions`	Edge regions inside the triangle, T_e, not required for this extrema, hence it is `NULL` for this function
`region.names`	Names of the edge regions, not required for this extrema, hence it is `NULL` for this function
`region.centers`	Centers of mass of the edge regions inside T_e, not required for this extrema, hence it is `NULL` for this function
`dist2ref`	Distances from closest points in each edge region to the corresponding edge

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`cl2edges.vert.reg.basic.tri`, `cl2edgesMvert.reg`, `cl2edgesCMvert.reg` and `fr2edgesCMedge.reg.std.tri`

Examples

```
n<-20  #try also n<-100
Xp<-runif.std.tri(n)$gen.points

Ext<-cl2edges.std.tri(Xp)
Ext
summary(Ext)
plot(Ext,asp=1)

ed.clo<-Ext

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3
p1<-(A+B)/2
p2<-(B+C)/2
p3<-(A+C)/2
```

```
Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),
ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(Xp,xlab="",ylab="")
points(ed.clo$ext,pty=2,pch=4,col="red")

txt<-rbind(Te,p1,p2,p3)
xc<-txt[,1]+c(-.03,.03,.03,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,0,0,0)
txt.str<-c("A","B","C","re=1","re=2","re=3")
text(xc,yc,txt.str)
```

`cl2edges.vert.reg.basic.tri` *The closest points among a data set in the vertex regions to the corresponding edges in a standard basic triangle*

Description

An object of class `"Extrema"`. Returns the closest data points among the data set, `Xp`, to edge i in `M`-vertex region i for $i = 1, 2, 3$ in the standard basic triangle $T_b = T(A = (0, 0), B = (1, 0), C = (c_1, c_2))$ where c_1 is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$. Vertex labels are $A = 1$, $B = 2$, and $C = 3$, and corresponding edge labels are $BC = 1$, $AC = 2$, and $AB = 3$.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle T_b or based on the circumcenter of T_b.

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
cl2edges.vert.reg.basic.tri(Xp, c1, c2, M)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.

`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle adjacent to the shorter edges; c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2+c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle T_b or the circumcenter of T_b.

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances to Edges in the Respective \eqn{M}-Vertex Regions"`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points to edges in the corresponding vertex region.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is T_b.
`cent`	The center point used for construction of vertex regions
`ncent`	Name of the center, `cent`, it is `"M"` or `"CC"` for this function
`regions`	Vertex regions inside the triangle, T_b.
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside T_b.
`dist2ref`	Distances of closest points in the vertex regions to corresponding edges.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

`cl2edgesCMvert.reg`, `cl2edgesMvert.reg`, and `cl2edges.std.tri`

Examples

```
c1<-.4;  c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);

set.seed(1)
n<-20
Xp<-runif.basic.tri(n,c1,c2)$g

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.3)

Ext<-cl2edges.vert.reg.basic.tri(Xp,c1,c2,M)
Ext
summary(Ext)
plot(Ext)

cl2e<-Ext

Ds<-prj.cent2edges.basic.tri(c1,c2,M)

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,pch=".",xlab="",ylab="",
main="Closest Points in M-Vertex Regions \n to the Opposite Edges",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(Xp,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

xc<-Tb[,1]+c(-.02,.02,0.02)
yc<-Tb[,2]+c(.02,.02,.02)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.04,-.03,0)
yc<-txt[,2]+c(-.02,.02,.02,-.03)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
```

`cl2edgesCCvert.reg` *The closest points in a data set to edges in each CC-vertex region in a triangle*

Description

An object of class `"Extrema"`. Returns the closest data points among the data set, `Xp`, to edge j in CC-vertex region j for $j = 1, 2, 3$ in the triangle, `tri`= $T(A, B, C)$, where CC stands for circumcenter. Vertex labels are $A = 1$, $B = 2$, and $C = 3$, and corresponding edge labels are $BC = 1$, $AC = 2$, and $AB = 3$. Function yields `NA` if there are no data points in a CC-vertex region.

See also (Ceyhan (2005, 2010)).

Usage

```
cl2edgesCCvert.reg(Xp, tri)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances to Edges in the Respective CC-Vertex Regions"`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points to edges in the respective vertex region.
`ind.ext`	Indices of the extrema points,`ext`.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is `tri`
`cent`	The center point used for construction of vertex regions
`ncent`	Name of the center, `cent`, it is `"CC"` for this function
`regions`	Vertex regions inside the triangle, `tri`, provided as a `list`
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside `tri`
`dist2ref`	Distances of closest points in the vertex regions to corresponding edges

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`cl2edges.vert.reg.basic.tri`, `cl2edgesCMvert.reg`, `cl2edgesMvert.reg`, and `cl2edges.std.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-20  #try also n<-100
set.seed(1)
Xp<-runif.tri(n,Tr)$g

Ext<-cl2edgesCCvert.reg(Xp,Tr)
Ext
summary(Ext)
plot(Ext)

cl2e<-Ext

CC<-circumcenter.tri(Tr);
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1],CC[1])
Ylim<-range(Tr[,2],Xp[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,asp=1,pch=".",xlab="",ylab="",
main="Closest Points in CC-Vertex Regions \n to the Opposite Edges",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)

xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)
```

```
points(Xp,pch=1,col=1)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

txt<-rbind(CC,Ds)
xc<-txt[,1]+c(-.04,.04,-.03,0)
yc<-txt[,2]+c(-.05,.04,.06,-.08)
txt.str<-c("CC","D1","D2","D3")
text(xc,yc,txt.str)
```

`cl2edgesCMvert.reg` *The closest points in a data set to edges in each CM-vertex region in a triangle*

Description

An object of class `"Extrema"`. Returns the closest data points among the data set, `Xp`, to edge j in CM-vertex region j for $j = 1, 2, 3$ in the triangle, `tri`$= T(A, B, C)$, where CM stands for center of mass. Vertex labels are $A = 1$, $B = 2$, and $C = 3$, and corresponding edge labels are $BC = 1$, $AC = 2$, and $AB = 3$. Function yields `NA` if there are no data points in a CM-vertex region.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2010, 2011)).

Usage

```
cl2edgesCMvert.reg(Xp, tri)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances to Edges in the Respective CM-Vertex Regions"`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points to edges in the respective vertex region.

`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is `tri`
`cent`	The center point used for construction of vertex regions
`ncent`	Name of the center, `cent`, it is `"CM"` for this function
`regions`	Vertex regions inside the triangle, `tri`, provided as a `list`
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside `tri`
`dist2ref`	Distances of closest points in the vertex regions to corresponding edges

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE (2007). “On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs.” *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`cl2edges.vert.reg.basic.tri`, `cl2edgesCCvert.reg`, `cl2edgesMvert.reg`, and `cl2edges.std.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-20  #try also n<-100
set.seed(1)
Xp<-runif.tri(n,Tr)$g

Ext<-cl2edgesCMvert.reg(Xp,Tr)
Ext
summary(Ext)
plot(Ext)

cl2e<-Ext
```

```
CM<-(A+B+C)/3;
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",
main="Closest Points in CM-Vertex Regions \n to the Opposite Edges",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)

xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

points(Xp,pch=1,col=1)
L<-matrix(rep(CM,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

txt<-rbind(CM,Ds)
xc<-txt[,1]+c(-.04,.04,-.03,0)
yc<-txt[,2]+c(-.05,.04,.06,-.08)
txt.str<-c("CM","D1","D2","D3")
text(xc,yc,txt.str)
```

`cl2edgesMvert.reg` *The closest points among a data set in the vertex regions to the respective edges in a triangle*

Description

An object of class `"Extrema"`. Returns the closest data points among the data set, `Xp`, to edge i in `M`-vertex region i for $i = 1, 2, 3$ in the triangle `tri`= $T(A, B, C)$. Vertex labels are $A = 1$, $B = 2$, and $C = 3$, and corresponding edge labels are $BC = 1$, $AC = 2$, and $AB = 3$.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`.

Two methods of finding these extrema are provided in the function, which can be chosen in the logical argument `alt`, whose default is `alt=FALSE`. When `alt=FALSE`, the function sequentially finds the vertex region of the data point and then updates the minimum distance to the opposite edge and the relevant extrema objects, and when `alt=TRUE`, it first partitions the data set according which vertex regions they reside, and then finds the minimum distance to the opposite edge and the relevant extrema on each partition. Both options yield equivalent results for the extrema points and indices, with the default being slightly ~ 20

See also (Ceyhan (2005, 2010)).

Usage

```
cl2edgesMvert.reg(Xp, tri, M, alt = FALSE)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; which may be entered as "CC" as well;
`alt`	A logical argument for alternative method of finding the closest points to the edges, default `alt=FALSE`. When `alt=FALSE`, the function sequentially finds the vertex region of the data point and then the minimum distance to the opposite edge and the relevant extrema objects, and when `alt=TRUE`, it first partitions the data set according which vertex regions they reside, and then finds the minimum distance to the opposite edge and the relevant extrema on each partition.

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances to Edges in the Respective \eqn{M}-Vertex Regions"`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points to edges in the respective vertex region.
`ind.ext`	The data indices of extrema points, `ext`.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is `tri`
`cent`	The center point used for construction of vertex regions
`ncent`	Name of the center, `cent`, it is `"M"` or `"CC"` for this function
`regions`	Vertex regions inside the triangle, `tri`, provided as a `list`
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside `tri`
`dist2ref`	Distances of closest points in the `M`-vertex regions to corresponding edges.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

`cl2edges.vert.reg.basic.tri`, `cl2edgesCMvert.reg`, and `cl2edges.std.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-20  #try also n<-100

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

Ext<-cl2edgesMvert.reg(Xp,Tr,M)
Ext
summary(Ext)
plot(Ext)

cl2e<-Ext

Ds<-prj.cent2edges(Tr,M)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",
main="Closest Points in M-Vertex Regions \n to the Opposite Edges",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
```

```
polygon(Tr)
points(Xp,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

xc<-Tr[,1]+c(-.02,.03,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.05,-.02,-.01)
yc<-txt[,2]+c(-.03,.02,.08,-.07)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
```

`cl2faces.vert.reg.tetra` *The closest points among a data set in the vertex regions to the respective faces in a tetrahedron*

Description

An object of class `"Extrema"`. Returns the closest data points among the data set, `Xp`, to face i in `M`-vertex region i for $i = 1, 2, 3, 4$ in the tetrahedron $th = T(A, B, C, D)$. Vertex labels are $A = 1$, $B = 2$, $C = 3$, and $D = 4$ and corresponding face labels are $BCD = 1$, $ACD = 2$, $ABD = 3$, and $ABC = 4$.

Vertex regions are based on center `M` which can be the center of mass (`"CM"`) or circumcenter (`"CC"`) of `th`.

Usage

```
cl2faces.vert.reg.tetra(Xp, th, M = "CM")
```

Arguments

`Xp`	A set of 3D points representing the set of data points.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, $C = 3$, and $D = 4$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances from Closest Points to Faces ..."`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points to faces in the respective vertex region.
`ind.ext`	The data indices of extrema points, `ext`.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is `th`
`cent`	The center point used for construction of vertex regions, it is circumcenter of center of mass for this function
`ncent`	Name of the center, it is circumcenter `"CC"` or center of mass `"CM"` for this function.
`regions`	Vertex regions inside the tetrahedron `th` provided as a list.
`region.names`	Names of the vertex regions as `"vr=1","vr=2","vr=3","vr=4"`
`region.centers`	Centers of mass of the vertex regions inside `th`.
`dist2ref`	Distances from closest points in each vertex region to the corresponding face.

Author(s)

Elvan Ceyhan

See Also

`fr2vertsCCvert.reg`, `fr2edgesCMedge.reg.std.tri`, `fr2vertsCCvert.reg.basic.tri` and `kfr2vertsCCvert.reg`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0);
D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
set.seed(1)
tetra<-rbind(A,B,C,D)+matrix(runif(12,-.25,.25),ncol=3)
n<-10  #try also n<-20
Cent<-"CC"  #try also "CM"

n<-20  #try also n<-100
Xp<-runif.tetra(n,tetra)$g  #try also Xp<-cbind(runif(n),runif(n),runif(n))

Ext<-cl2faces.vert.reg.tetra(Xp,tetra,Cent)
```

```
Ext
summary(Ext)
plot(Ext)

clf<-Ext$ext

if (Cent=="CC") {M<-circumcenter.tetra(tetra)}
if (Cent=="CM") {M<-apply(tetra,2,mean)}

Xlim<-range(tetra[,1],Xp[,1],M[1])
Ylim<-range(tetra[,2],Xp[,2],M[2])
Zlim<-range(tetra[,3],Xp[,3],M[3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Xp[,1],Xp[,2],Xp[,3], phi =0,theta=40, bty = "g",
main="Closest Pointsin CC-Vertex Regions \n to the Opposite Faces",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
plot3D::points3D(clf[,1],clf[,2],clf[,3], pch=4,col="red", add=TRUE)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3],
labels=c("A","B","C","D"), add=TRUE)

#for center of mass use #Cent<-apply(tetra,2,mean)
D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2;
D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-rbind(M,M,M,M,M,M)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)
```

`cl2Mc.int` *The closest points to center in each vertex region in an interval*

Description

An object of class `"Extrema"`. Returns the closest data points among the data set, `Xp`, in each M_c-vertex region i.e., finds the closest points from right and left to M_c among points of the 1D data set `Xp` which reside in in the interval `int`$= (a, b)$.

M_c is based on the centrality parameter $c \in (0, 1)$, so that $100c$ % of the length of interval is to the left of M_c and $100(1 - c)$ % of the length of the interval is to the right of M_c. That is, for the interval (a, b), $M_c = a + c(b - a)$. If there are no points from `Xp` to the left of M_c in the interval, then it yields `NA`, and likewise for the right of M_c in the interval.

See also (Ceyhan (2012)).

Usage

```
cl2Mc.int(Xp, int, c)
```

Arguments

`Xp`	A set or `vector` of 1D points from which closest points to M_c are found in the interval `int`.
`int`	A `vector` of two real numbers representing an interval.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A `list` with the elements

`txt1`	Vertex Labels are $a = 1$ and $b = 2$ for the interval (a, b).
`txt2`	A short description of the distances as `"Distances from ..."`
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points to M_c in each vertex region
`ind.ext`	The data indices of extrema points, `ext`.
`X`	The input data vector, `Xp`.
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is `int`.
`cent`	The (parameterized) center point used for construction of vertex regions.
`ncent`	Name of the (parameterized) center, `cent`, it is `"Mc"` for this function.
`regions`	Vertex regions inside the interval, `int`, provided as a list.
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`
`region.centers`	Centers of mass of the vertex regions inside `int`.
`dist2ref`	Distances from closest points in each vertex region to M_c.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`cl2CCvert.reg.basic.tri` and `cl2CCvert.reg`

Examples

```
c<-.4
a<-0; b<-10; int<-c(a,b)

Mc<-centerMc(int,c)

nx<-10
xr<-range(a,b,Mc)
xf<-(xr[2]-xr[1])*.5

Xp<-runif(nx,a,b)

Ext<-cl2Mc.int(Xp,int,c)
Ext
summary(Ext)
plot(Ext)

cMc<-Ext

Xlim<-range(a,b,Xp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),xlab="",pch=".",
main=paste("Closest Points in Mc-Vertex Regions \n to the Center Mc = ",Mc,sep=""),
  xlim=Xlim+xd*c(-.05,.05))
  abline(h=0)
abline(v=c(a,b,Mc),col=c(1,1,2),lty=2)
points(cbind(Xp,0))
points(cbind(c(cMc$ext),0),pch=4,col=2)
text(cbind(c(a,b,Mc)-.02*xd,-0.05),c("a","b",expression(M[c])))
```

`CSarc.dens.test` *A test of segregation/association based on arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 2D data*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the convex hull of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points) and association (where `Xp` points cluster around `Yp` points) based on the normal approximation of the arc density of the CS-PCD for uniform 2D data in the convex hull of `Yp` points.

The function yields the test statistic, p-value for the corresponding `alternative`, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the convex hull of `Yp` points, arc density of CS-PCD whose vertices are `Xp` points equals to its expected value under the uniform distribution and `alternative` could be two-sided, or left-sided (i.e., data is accumulated around the `Yp` points, or association) or right-sided (i.e., data is accumulated around the centers of the triangles, or segregation).

CS proximity region is constructed with the expansion parameter $t > 0$ and CM-edge regions (i.e., the test is not available for a general center M at this version of the function).

Caveat: This test is currently a conditional test, where `Xp` points are assumed to be random, while `Yp` points are assumed to be fixed (i.e., the test is conditional on `Yp` points). Furthermore, the test is a large sample test when `Xp` points are substantially larger than `Yp` points, say at least 5 times more. This test is more appropriate when supports of `Xp` and `Yp` has a substantial overlap. Currently, the `Xp` points outside the convex hull of `Yp` points are handled with a convex hull correction factor, `ch.cor`, which is derived under the assumption of uniformity of `Xp` and `Yp` points in the study window, (see the description below and the function code.) However, in the special case of no `Xp` points in the convex hull of `Yp` points, arc density is taken to be 1, as this is clearly a case of segregation. Removing the conditioning and extending it to the case of non-concurring supports is an ongoing line of research of the author of the package.

`ch.cor` is for convex hull correction (default is `"no convex hull correction"`, i.e., `ch.cor=FALSE`) which is recommended when both `Xp` and `Yp` have the same rectangular support.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
CSarc.dens.test(
  Xp,
  Yp,
  t,
  ch.cor = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`ch.cor`	A logical argument for convex hull correction, default `ch.cor=FALSE`, recommended when both `Xp` and `Yp` have the same rectangular support.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the arc density of CS-PCD based on the 2D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`
`conf.int`	Confidence interval for the arc density at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., arc density
`null.value`	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). “Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering.” *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`PEarc.dens.test` and `CSarc.dens.test1D`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-100; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

plotDelaunay.tri(Xp,Yp,xlab="",ylab = "")

CSarc.dens.test(Xp,Yp,t=.5)
```

```
CSarc.dens.test(Xp,Yp,t=.5,ch=TRUE)
#try also t=1.0 and 1.5 above
```

`CSarc.dens.test.int` *A test of uniformity of 1D data in a given interval based on Central Similarity Proximity Catch Digraph (CS-PCD)*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of uniformity of 1D data in one interval based on the normal approximation of the arc density of the CS-PCD with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$.

The function yields the test statistic, p-value for the corresponding `alternative`, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

The null hypothesis is that data is uniform in a finite interval (i.e., arc density of CS-PCD equals to its expected value under uniform distribution) and `alternative` could be two-sided, or left-sided (i.e., data is accumulated around the end points) or right-sided (i.e., data is accumulated around the mid point or center M_c).

See also (Ceyhan (2016)).

Usage

```
CSarc.dens.test.int(
  Xp,
  int,
  t,
  c = 0.5,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of CS-PCD.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the arc density of CS-PCD based on the 1D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`
`conf.int`	Confidence interval for the arc density at the given level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., arc density
`null.value`	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`PEarc.dens.test.int`

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

n<-10
Xp<-runif(n,a,b)

num.arcsCSmid.int(Xp,int,t,c)
CSarc.dens.test.int(Xp,int,t,c)

num.arcsCSmid.int(Xp,int,t,c=.3)
CSarc.dens.test.int(Xp,int,t,c=.3)

num.arcsCSmid.int(Xp,int,t=1.5,c)
CSarc.dens.test.int(Xp,int,t=1.5,c)

Xp<-runif(n,a-1,b+1)
num.arcsCSmid.int(Xp,int,t,c)
CSarc.dens.test.int(Xp,int,t,c)
```

```
c<-.4
t<-.5
a<-0; b<-10; int<-c(a,b)
n<-10  #try also n<-20
Xp<-runif(n,a,b)

CSarc.dens.test.int(Xp,int,t,c)
```

`CSarc.dens.test1D` *A test of segregation/association based on arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the range (i.e., range) of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points) and association (where `Xp` points cluster around `Yp` points) based on the normal approximation of the arc density of the CS-PCD for uniform 1D data.

The function yields the test statistic, p-value for the corresponding `alternative`, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the range of `Yp` points, arc density of CS-PCD whose vertices are `Xp` points equals to its expected value under the uniform distribution and `alternative` could be two-sided, or left-sided (i.e., data is accumulated around the `Yp` points, or association) or right-sided (i.e., data is accumulated around the centers of the intervals, or segregation).

CS proximity region is constructed with the expansion parameter $t > 0$ and centrality parameter `c` which yields M-vertex regions. More precisely, for a middle interval $(y_{(i)}, y_{(i+1)})$, the center is $M = y_{(i)} + c(y_{(i+1)} - y_{(i)})$ for the centrality parameter $c \in (0, 1)$. If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

Caveat: This test is currently a conditional test, where `Xp` points are assumed to be random, while `Yp` points are assumed to be fixed (i.e., the test is conditional on `Yp` points). Furthermore, the test is a large sample test when `Xp` points are substantially larger than `Yp` points, say at least 5 times more. This test is more appropriate when supports of `Xp` and `Yp` have a substantial overlap. Currently, the `Xp` points outside the range of `Yp` points are handled with a range correction (or end-interval correction) factor (see the description below and the function code.) However, in the special case of no `Xp` points in the range of `Yp` points, arc density is taken to be 1, as this is clearly a case of segregation. Removing the conditioning and extending it to the case of non-concurring supports is an ongoing line of research of the author of the package.

`end.int.cor` is for end-interval correction, recommended when both `Xp` and `Yp` have the same interval support (default is "no end-interval correction", i.e., `end.int.cor=FALSE`).

Usage

```
CSarc.dens.test1D(
  Xp,
  Yp,
  t,
  c = 0.5,
  support.int = NULL,
  end.int.cor = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 1D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 1D points which constitute the end points of the partition intervals.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number which serves as the centrality parameter in CS proximity region; must be in $(0, 1)$ (default `c=.5`).
`support.int`	Support interval (a, b) with $a < b$. Uniformity of `Xp` points in this interval is tested. Default is `NULL`.
`end.int.cor`	A logical argument for end-interval correction, default is `FALSE`, recommended when both `Xp` and `Yp` have the same interval support.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the arc density CS-PCD whose vertices are the 1D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`.
`conf.int`	Confidence interval for the arc density at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., arc density
`null.value`	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

`CSarc.dens.test` and `CSarc.dens.test.int`

Examples

```
tau<-2
c<-.4
a<-0; b<-10; int=c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

CSarc.dens.test1D(Xp,Yp,tau,c,int)
CSarc.dens.test1D(Xp,Yp,tau,c,int,alt="l")
CSarc.dens.test1D(Xp,Yp,tau,c,int,alt="g")

CSarc.dens.test1D(Xp,Yp,tau,c,int,end.int.cor = TRUE)

Yp2<-runif(ny,a,b)+11
CSarc.dens.test1D(Xp,Yp2,tau,c,int)

n<-10  #try also n<-20
Xp<-runif(n,a,b)
CSarc.dens.test1D(Xp,Yp,tau,c,int)
```

`CSarc.dens.tri` *Arc density of Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns the arc density of CS-PCD whose vertex set is the given 2D numerical data set, `Xp`, (some of its members are) in the triangle `tri`.

CS proximity regions is defined with respect to `tri` with expansion parameter $t > 0$ and edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`. The function also provides arc density standardized by the mean and asymptotic variance of the arc density of CS-PCD for uniform data in the triangle `tri` only when `M` is the center of mass. For the number of arcs, loops are not allowed.

is a logical argument (default is `FALSE`) for considering only the points inside the triangle or all the points as the vertices of the digraph. if `in.tri.only=TRUE`, arc density is computed only for the points inside the triangle (i.e., arc density of the subdigraph induced by the vertices in the triangle is computed), otherwise arc density of the entire digraph (i.e., digraph with all the vertices) is computed.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) for more on CS-PCDs.

Usage

```
CSarc.dens.tri(Xp, tri, t, M = c(1, 1, 1), in.tri.only = FALSE)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.
`in.tri.only`	A logical argument (default is `=FALSE`) for computing the arc density for only the points inside the triangle, `tri`. That is, if `=TRUE` arc density of the induced subdigraph with the vertices inside `tri` is computed, otherwise otherwise arc density of the entire digraph (i.e., digraph with all the vertices) is computed.

Value

A `list` with the elements

`arc.dens`	Arc density of CS-PCD whose vertices are the 2D numerical data set, `Xp`; CS proximity regions are defined with respect to the triangle `tri` and `M`-edge regions
`std.arc.dens`	Arc density standardized by the mean and asymptotic variance of the arc density of CS-PCD for uniform data in the triangle `tri`.This will only be returned if `M` is the center of mass.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). “Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering.” *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`ASarc.dens.tri`, `PEarc.dens.tri`, and `num.arcsCStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

CSarc.dens.tri(Xp,Tr,t=.5,M)
CSarc.dens.tri(Xp,Tr,t=.5,M, in.tri.only= FALSE)
#try also t=1 and t=1.5 above
```

`dimension` *The dimension of a* `vector` *or matrix or a data frame*

Description

Returns the dimension (i.e., number of columns) of `x`, which is a matrix or a `vector` or a data frame. This is different than the `dim` function in base `R`, in the sense that, `dimension` gives only the number of columns of the argument `x`, while `dim` gives the number of rows and columns of `x`. `dimension` also works for a scalar or a vector, while `dim` yields `NULL` for such arguments.

Usage

```
dimension(x)
```

Arguments

`x`	A `vector` or a matrix or a data frame whose dimension is to be determined.

Value

Dimension (i.e., number of columns) of `x`

Author(s)

Elvan Ceyhan

See Also

`is.point` and `dim` from the base distribution of `R`

Examples

```
dimension(3)
dim(3)

A<-c(1,2)
dimension(A)
dim(A)

B<-c(2,3)
dimension(rbind(A,B,A))
dimension(cbind(A,B,A))

M<-matrix(runif(20),ncol=5)
dimension(M)
dim(M)

dimension(c("a","b"))
```

`Dist` *The distance between two vectors, matrices, or data frames*

Description

Returns the Euclidean distance between `x` and `y` which can be vectors or matrices or data frames of any dimension (`x` and `y` should be of same dimension).

This function is different from the `dist` function in the `stats` package of the standard `R` distribution. `dist` requires its argument to be a data matrix and `dist` computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix (Becker et al. (1988)), while `Dist` needs two arguments to find the distances between. For two data matrices A and B, `dist(rbind(as.vector(A), as.vector(B)))` and `Dist(A,B)` yield the same result.

Usage

```
Dist(x, y)
```

Arguments

`x, y`	Vectors, matrices or data frames (both should be of the same type).

Value

Euclidean distance between `x` and `y`

Author(s)

Elvan Ceyhan

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`dist` from the base package `stats`

Examples

```
B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Dist(B,C);
dist(rbind(B,C))

x<-runif(10)
y<-runif(10)
Dist(x,y)

xm<-matrix(x,ncol=2)
ym<-matrix(y,ncol=2)
Dist(xm,ym)
dist(rbind(as.vector(xm),as.vector(ym)))

Dist(xm,xm)
```

`dist.point2line` *The distance from a point to a line defined by two points*

Description

Returns the distance from a point `p` to the line joining points `a` and `b` in 2D space.

Usage

```
dist.point2line(p, a, b)
```

Arguments

`p`	A 2D point, distance from `p` to the line passing through points `a` and `b` are to be computed.
`a, b`	2D points that determine the straight line (i.e., through which the straight line passes).

Value

A `list` with two elements

`dis`	Distance from point `p` to the line passing through `a` and `b`
`cl2p`	The closest point on the line passing through `a` and `b` to the point `p`

Author(s)

Elvan Ceyhan

See Also

`dist.point2plane`, `dist.point2set`, and `Dist`

Examples

```
A<-c(1,2); B<-c(2,3); P<-c(3,1.5)

dpl<-dist.point2line(P,A,B);
dpl
C<-dpl$cl2p
pts<-rbind(A,B,C,P)

xr<-range(pts[,1])
xf<-(xr[2]-xr[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100
lnAB<-Line(A,B,x)
y<-lnAB$y
int<-lnAB$intercept #intercept
sl<-lnAB$slope #slope

xsq<-seq(min(A[1],B[1],P[1])-xf,max(A[1],B[1],P[1])+xf,l=5)
#try also l=10, 20, or 100
pline<-(-1/sl)*(xsq-P[1])+P[2]
#line passing thru P and perpendicular to AB

Xlim<-range(pts[,1],x)
Ylim<-range(pts[,2],y)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(P),asp=1,pch=1,xlab="x",ylab="y",
main="Illustration of the distance from P \n to the Line Crossing Points A and B",
```

```
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(rbind(A,B),pch=1)
lines(x,y,lty=1,xlim=Xlim,ylim=Ylim)
int<-round(int,2); sl<-round(sl,2)
text(rbind((A+B)/2+xd*c(-.01,-.01)),ifelse(sl==0,paste("y=",int),
ifelse(sl==1,paste("y=x+",int),
ifelse(int==0,paste("y=",sl,"x"),paste("y=",sl,"x+",int)))))
text(rbind(A+xd*c(0,-.01),B+xd*c(.0,-.01),P+xd*c(.01,-.01)),c("A","B","P"))
lines(xsq,pline,lty=2)
segments(P[1],P[2], C[1], C[2], lty=1,col=2,lwd=2)
text(rbind(C+xd*c(-.01,-.01)),"C")
text(rbind((P+C)/2),col=2,paste("d=",round(dpl$dis,2)))
```

`dist.point2plane` *The distance from a point to a plane spanned by three 3D points*

Description

Returns the distance from a point `p` to the plane passing through points `a`, `b`, and `c` in 3D space.

Usage

```
dist.point2plane(p, a, b, c)
```

Arguments

`p`	A 3D point, distance from `p` to the plane passing through points `a`, `b`, and `c` are to be computed.
`a, b, c`	3D points that determine the plane (i.e., through which the plane is passing).

Value

A `list` with two elements

`dis`	Distance from point `p` to the plane spanned by 3D points `a`, `b`, and `c`
`cl2pl`	The closest point on the plane spanned by 3D points `a`, `b`, and `c` to the point `p`

Author(s)

Elvan Ceyhan

See Also

`dist.point2line`, `dist.point2set`, and `Dist`

Examples

```
P<-c(5,2,40)
P1<-c(1,2,3); P2<-c(3,9,12); P3<-c(1,1,3);

dis<-dist.point2plane(P,P1,P2,P3);
dis
Pr<-dis$prj #projection on the plane

xseq<-seq(0,10,l=5)  #try also l=10, 20, or 100
yseq<-seq(0,10,l=5)  #try also l=10, 20, or 100

pl.grid<-Plane(P1,P2,P3,xseq,yseq)$z

plot3D::persp3D(z = pl.grid, x = xseq, y = yseq, theta =225, phi = 30,
ticktype = "detailed",
expand = 0.7, facets = FALSE, scale = TRUE,
main="Point P and its Orthogonal Projection \n on the Plane Defined by P1, P2, P3")
#plane spanned by points P1, P2, P3
#add the vertices of the tetrahedron
plot3D::points3D(P[1],P[2],P[3], add=TRUE)
plot3D::points3D(Pr[1],Pr[2],Pr[3], add=TRUE)
plot3D::segments3D(P[1], P[2], P[3], Pr[1], Pr[2],Pr[3], add=TRUE,lwd=2)

plot3D::text3D(P[1]-.5,P[2],P[3]+1, c("P"),add=TRUE)
plot3D::text3D(Pr[1]-.5,Pr[2],Pr[3]+2, c("Pr"),add=TRUE)

persp(xseq,yseq,pl.grid, xlab="x",ylab="y",zlab="z",theta = -30,
phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.05, ticktype = "detailed")
```

`dist.point2set` *Distance from a point to a set of finite cardinality*

Description

Returns the Euclidean distance between a point `p` and set of points `Yp` and the closest point in set `Yp` to `p`. Distance between a point and a set is by definition the distance from the point to the closest point in the set. `p` should be of finite dimension and `Yp` should be of finite cardinality and `p` and elements of `Yp` must have the same dimension.

Usage

```
dist.point2set(p, Yp)
```

Arguments

`p`	A `vector` (i.e., a point in R^d).
`Yp`	A set of d-dimensional points.

Value

A `list` with the elements

`distance`	Distance from point `p` to set `Yp`
`ind.cl.point`	Index of the closest point in set `Yp` to the point `p`
`closest.point`	The closest point in set `Yp` to the point `p`

Author(s)

Elvan Ceyhan

See Also

`dist.point2line` and `dist.point2plane`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
dist.point2set(c(1,2),Te)

X2<-cbind(runif(10),runif(10))
dist.point2set(c(1,2),X2)

x<-runif(1)
y<-as.matrix(runif(10))
dist.point2set(x,y)
#this works, because x is a 1D point, and y is treated as a set of 10 1D points
#but will give an error message if y<-runif(10) is used above
```

`dom.num.exact` *Exact domination number (i.e., domination number by the exact algorithm)*

Description

Returns the (exact) domination number based on the incidence matrix `Inc.Mat` of a graph or a digraph and the indices (i.e., row numbers of `Inc.Mat`) for the corresponding (exact) minimum dominating set. Here the row number in the incidence matrix corresponds to the index of the vertex (i.e., index of the data point). The function works whether loops are allowed or not (i.e., whether the first diagonal is all 1 or all 0). It takes a rather long time for large number of vertices (i.e., large number of row numbers).

Usage

```
dom.num.exact(Inc.Mat)
```

Arguments

`Inc.Mat`	A square matrix consisting of 0's and 1's which represents the incidence matrix of a graph or digraph.

Value

A `list` with two elements

`dom.num`	The cardinality of the (exact) minimum dominating set, i.e., (exact) domination number of the graph or digraph whose incidence matrix `Inc.Mat` is given as input.
`ind.mds`	The vector of indices of the rows in the incidence matrix `Inc.Mat` for the (exact) minimum dominating set. The row numbers in the incidence matrix correspond to the indices of the vertices (i.e., indices of the data points).

Author(s)

Elvan Ceyhan

See Also

`dom.num.greedy`, `PEdom.num1D`, `PEdom.num.tri`, `PEdom.num.nondeg`, and `Idom.numCSup.bnd.tri`

Examples

```
n<-10
M<-matrix(sample(c(0,1),n^2,replace=TRUE),nrow=n)
diag(M)<-1

dom.num.greedy(M)
Idom.num.up.bnd(M,2)
dom.num.exact(M)
```

`dom.num.greedy` *Approximate domination number and approximate dominating set by the greedy algorithm*

Description

Returns the (approximate) domination number and the indices (i.e., row numbers) for the corresponding (approximate) minimum dominating set based on the incidence matrix `Inc.Mat` of a graph or a digraph by using the greedy algorithm (Chvatal (1979)). Here the row number in the incidence matrix corresponds to the index of the vertex (i.e., index of the data point). The function works whether loops are allowed or not (i.e., whether the first diagonal is all 1 or all 0). This function may yield the actual domination number or overestimates it.

Usage

```
dom.num.greedy(Inc.Mat)
```

Arguments

`Inc.Mat`	A square matrix consisting of 0's and 1's which represents the incidence matrix of a graph or digraph.

Value

A `list` with two elements

`dom.num`	The cardinality of the (approximate) minimum dominating set found by the greedy algorithm. i.e., (approximate) domination number of the graph or digraph whose incidence matrix `Inc.Mat` is given as input.
`ind.dom.set`	Indices of the rows in the incidence matrix `Inc.Mat` for the ((approximate) minimum dominating set). The row numbers in the incidence matrix correspond to the indices of the vertices (i.e., indices of the data points).

Author(s)

Elvan Ceyhan

References

Chvatal V (1979). "A greedy heuristic for the set-covering problem." *Mathematics of Operations Research*, **4**(3), 233 — 235.

Examples

```
n<-5
M<-matrix(sample(c(0,1),n^2,replace=TRUE),nrow=n)
diag(M)<-1

dom.num.greedy(M)
```

`edge.reg.triCM` *The vertices of the CM-edge region in a triangle that contains the point*

Description

Returns the edge whose region contains point, `p`, in the triangle `tri`$= T(A,B,C)$ with edge regions based on center of mass $CM = (A+B+C)/3$.

This function is related to `rel.edge.triCM`, but unlike `rel.edge.triCM` the related edges are given as vertices `ABC` for $re = 3$, as `BCA` for $re = 1$ and as `CAB` for $re = 2$ where edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. The vertices are given one vertex in each row in the output, e.g., ABC is printed as `rbind(A,B,C)`, where row 1 has the entries of vertex A, row 2 has the entries of vertex B, and row 3 has the entries of vertex C.

If the point, `p`, is not inside `tri`, then the function yields `NA` as output.

Edge region for BCA is the triangle $T(B,C,CM)$, edge region CAB is $T(A,C,CM)$, and edge region ABC is $T(A,B,CM)$.

See also (Ceyhan (2005, 2010)).

Usage

```
edge.reg.triCM(p, tri)
```

Arguments

`p`	A 2D point for which CM-edge region it resides in is to be determined in the triangle `tri`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

The CM-edge region that contains point, `p` in the triangle `tri`. The related edges are given as vertices `ABC` for $re = 3$, as `BCA` for $re = 1$ and as `CAB` for $re = 2$ where edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.edge.tri`, `rel.edge.triCM`, `rel.edge.basic.triCM`, `rel.edge.basic.tri`, `rel.edge.std.triCM`, and `edge.reg.triCM`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(.4,.2)  #try also P<-as.numeric(runif.tri(1,Tr)$g)
edge.reg.triCM(P,Tr)

P<-c(1.8,.5)
edge.reg.triCM(P,Tr)

CM<-(A+B+C)/3
p1<-(A+B+CM)/3
p2<-(B+C+CM)/3
p3<-(A+C+CM)/3

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-Tr; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Tr,CM,p1,p2,p3)
xc<-txt[,1]+c(-.02,.02,.02,-.05,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.02,0,0,0)
txt.str<-c("A","B","C","CM","re=T(A,B,CM)","re=T(B,C,CM)","re=T(A,C,CM)")
text(xc,yc,txt.str)
```

`fr2edgesCMedge.reg.std.tri` *The furthest points in a data set from edges in each CM-edge region in the standard equilateral triangle*

Description

An object of class `"Extrema"`. Returns the furthest data points among the data set, `Xp`, in each CM-edge region from the edge in the standard equilateral triangle $T_e = T(A = (0, 0), B = (1, 0), C = (1/2, \sqrt{3}/2))$.

`ch.all.intri` is for checking whether all data points are inside T_e (default is `FALSE`).

See also (Ceyhan (2005)).

Usage

```
fr2edgesCMedge.reg.std.tri(Xp, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points, some could be inside and some could be outside standard equilateral triangle T_e.
`ch.all.intri`	A logical argument used for checking whether all data points are inside T_e (default is `FALSE`).

Value

A `list` with the elements

`txt1`	Edge labels as $AB = 3$, $BC = 1$, and $AC = 2$ for T_e (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances to Edges"`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, furthest points from edges in each edge region.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is T_e.
`cent`	The center point used for construction of edge regions.
`ncent`	Name of the center, `cent`, it is center of mass `"CM"` for this function.
`regions`	Edge regions inside the triangle, T_e, provided as a list.
`region.names`	Names of the edge regions as `"er=1"`, `"er=2"`, and `"er=3"`.
`region.centers`	Centers of mass of the edge regions inside T_e.
`dist2ref`	Distances from furthest points in each edge region to the corresponding edge.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

See Also

`fr2vertsCCvert.reg.basic.tri`, `fr2vertsCCvert.reg`, `fr2vertsCCvert.reg.basic.tri`, `kfr2vertsCCvert.reg`, and `cl2edges.std.tri`

Examples

```
n<-20
Xp<-runif.std.tri(n)$gen.points

Ext<-fr2edgesCMedge.reg.std.tri(Xp)
Ext
summary(Ext)
plot(Ext,asp=1)

ed.far<-Ext

Xp2<-rbind(Xp,c(.8,.8))
fr2edgesCMedge.reg.std.tri(Xp2)
fr2edgesCMedge.reg.std.tri(Xp2,ch.all.intri = FALSE)
#gives error if ch.all.intri = TRUE

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3
p1<-(A+B)/2
p2<-(B+C)/2
p3<-(A+C)/2

Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",
main="Furthest Points in CM-Edge Regions \n of Std Equilateral Triangle from its Edges",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(Xp,xlab="",ylab="")
points(ed.far$ext,pty=2,pch=4,col="red")

txt<-rbind(Te,CM,p1,p2,p3)
xc<-txt[,1]+c(-.03,.03,.03,-.06,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.02,0,0,0)
txt.str<-c("A","B","C","CM","re=2","re=3","re=1")
```

```
text(xc,yc,txt.str)
```

`fr2vertsCCvert.reg` *The furthest points in a data set from vertices in each CC-vertex region in a triangle*

Description

An object of class `"Extrema"`. Returns the furthest data points among the data set, `Xp`, in each CC-vertex region from the vertex in the triangle, `tri`$= T(A, B, C)$. Vertex region labels/numbers correspond to the row number of the vertex in `tri`. `ch.all.intri` is for checking whether all data points are inside `tri` (default is `FALSE`).

If some of the data points are not inside `tri` and `ch.all.intri=TRUE`, then the function yields an error message. If some of the data points are not inside `tri` and `ch.all.intri=FALSE`, then the function yields the closest points to edges among the data points inside `tri` (yields `NA` if there are no data points inside `tri`).

See also (Ceyhan (2005, 2012)).

Usage

```
fr2vertsCCvert.reg(Xp, tri, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`ch.all.intri`	A logical argument (default=`FALSE`) to check whether all data points are inside the triangle `tri`. So, if it is `TRUE`, the function checks if all data points are inside the closure of the triangle (i.e., interior and boundary combined) else it does not.

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances from furthest points to ..."`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, furthest points from vertices in each CC-vertex region in the triangle `tri`.

`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is the triangle `tri` for this function.
`cent`	The center point used for construction of edge regions.
`ncent`	Name of the center, `cent`, it is circumcenter `"CC"` for this function
`regions`	CC-Vertex regions inside the triangle, `tri`, provided as a `list`
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside `tri`
`dist2ref`	Distances from furthest points in each vertex region to the corresponding vertex

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`fr2vertsCCvert.reg.basic.tri`, `fr2edgesCMedge.reg.std.tri`, `kfr2vertsCCvert.reg.basic.tri` and `kfr2vertsCCvert.reg`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.tri(n,Tr)$g

Ext<-fr2vertsCCvert.reg(Xp,Tr)
Ext
summary(Ext)
plot(Ext)

f2v<-Ext

CC<-circumcenter.tri(Tr)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
```

```
Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,xlab="",asp=1,ylab="",pch=".",
main="Furthest Points in CC-Vertex Regions \n from the Vertices",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(Xp)
points(rbind(f2v$ext),pch=4,col=2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.06,.08,.05,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.05,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

Xp2<-rbind(Xp,c(.2,.4))
fr2vertsCCvert.reg(Xp2,Tr,ch.all.intri = FALSE)
#gives an error message if ch.all.intri = TRUE
#since not all points in the data set are in the triangle
```

`fr2vertsCCvert.reg.basic.tri` *The furthest points from vertices in each CC-vertex region in a standard basic triangle*

Description

An object of class `"Extrema"`. Returns the furthest data points among the data set, `Xp`, in each CC-vertex region from the corresponding vertex in the standard basic triangle $T_b = T(A = (0, 0), B = (1, 0), C = (c_1, c_2))$.

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle is useful for simulation studies under the uniformity hypothesis.

`ch.all.intri` is for checking whether all data points are inside T_b (default is `FALSE`).

See also (Ceyhan (2005, 2012)).

Usage

```
fr2vertsCCvert.reg.basic.tri(Xp, c1, c2, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points.
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle. adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$
`ch.all.intri`	A logical argument for checking whether all data points are inside T_b (default is `FALSE`).

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances from furthest points to ..."`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, furthest points from vertices in each vertex region.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is T_b.
`cent`	The center point used for construction of edge regions.
`ncent`	Name of the center, `cent`, it is circumcenter `"CC"` for this function.
`regions`	Vertex regions inside the triangle, T_b, provided as a `list`.
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside T_b.
`dist2ref`	Distances from furthest points in each vertex region to the corresponding vertex.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

fr2vertsCCvert.reg, fr2edgesCMedge.reg.std.tri, and kfr2vertsCCvert.reg

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-20

set.seed(1)
Xp<-runif.basic.tri(n,c1,c2)$g

Ext<-fr2vertsCCvert.reg.basic.tri(Xp,c1,c2)
Ext
summary(Ext)
plot(Ext)

f2v<-Ext

CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",
main="Furthest Points in CC-Vertex Regions \n from the Vertices",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(Xp)
points(rbind(f2v$ext),pch=4,col=2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.03,0.02,.07,.06,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,.01,.02,.02,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
```

`funsAB2CMTe` *The lines joining two vertices to the center of mass in standard equilateral triangle*

Description

Two functions, `lineA2CMinTe` and `lineB2CMinTe` of class `"TriLines"`. Returns the `equation, slope, intercept`, and y-coordinates of the lines joining A and CM and also B and CM.

`lineA2CMinTe` is the line joining A to the center of mass, CM, and `lineB2CMinTe` is the line joining B to the center of mass, CM, in the standard equilateral triangle $T_e = (A, B, C)$ with $A = (0, 0)$, $B = (1, 0)$, $C = (1/2, \sqrt{3}/2)$; x-coordinates are provided in `vector x`.

Usage

```
lineA2CMinTe(x)

lineB2CMinTe(x)
```

Arguments

`x`	A single scalar or a `vector` of scalars which is the argument of the functions `lineA2CMinTe` and `lineB2CMinTe`.

Value

A `list` with the elements

`txt1`	Longer description of the line.
`txt2`	Shorter description of the line (to be inserted over the line in the plot).
`mtitle`	The `"main"` title for the plot of the line.
`cent`	The center chosen inside the standard equilateral triangle.
`cent.name`	The name of the center inside the standard equilateral triangle. It is `"CM"` for these two functions.
`tri`	The triangle (it is the standard equilateral triangle for this function).
`x`	The input vector, can be a scalar or a `vector` of scalars, which constitute the x-coordinates of the point(s) of interest on the line.
`y`	The output vector, will be a scalar if `x` is a scalar or a `vector` of scalars if `x` is a `vector` of scalar, constitutes the y-coordinates of the point(s) of interest on the line.
`slope`	Slope of the line.
`intercept`	Intercept of the line.
`equation`	Equation of the line.

Author(s)

Elvan Ceyhan

See Also

`lineA2MinTe`, `lineB2MinTe`, and `lineC2MinTe`

Examples

```
#Examples for lineA2CMinTe
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by = .1)  #try also by = .01

lnACM<-lineA2CMinTe(x)
lnACM
summary(lnACM)
plot(lnACM)

CM<-(A+B+C)/3;
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Te,CM,D1,D2,D3,c(.25,lineA2CMinTe(.25)$y),c(.75,lineB2CMinTe(.75)$y))
xc<-txt[,1]+c(-.02,.02,.02,.05,.05,-.03,.0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.02,0,.02,-.04,0,0)
txt.str<-c("A","B","C","CM","D1","D2","D3","lineA2CMinTe(x)","lineB2CMinTe(x)")
text(xc,yc,txt.str)

lineA2CMinTe(.25)$y



#Examples for lineB2CMinTe
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by = .1)  #try also by = .01

lnBCM<-lineB2CMinTe(x)
lnBCM
summary(lnBCM)
plot(lnBCM,xlab=" x",ylab="y")

lineB2CMinTe(.25)$y
```

`funsAB2MTe` *The lines joining the three vertices of the standard equilateral triangle to a center,* `M`*, of it*

Description

Three functions, `lineA2MinTe`, `lineB2MinTe` and `lineC2MinTe` of class `"TriLines"`. Returns the `equation, slope, intercept`, and y-coordinates of the lines joining A and `M`, B and `M`, and also C and `M`.

`lineA2MinTe` is the line joining A to the center, `M`, `lineB2MinTe` is the line joining B to `M`, and `lineC2MinTe` is the line joining C to `M`, in the standard equilateral triangle $T_e = (A, B, C)$ with $A = (0, 0)$, $B = (1, 0)$, $C = (1/2, \sqrt{3}/2)$; x-coordinates are provided in `vector x`

Usage

```
lineA2MinTe(x, M)

lineB2MinTe(x, M)

lineC2MinTe(x, M)
```

Arguments

`x`	A single scalar or a `vector` of scalars.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle.

Value

A `list` with the elements

`txt1`	Longer description of the line.
`txt2`	Shorter description of the line (to be inserted over the line in the plot).
`mtitle`	The `"main"` title for the plot of the line.
`cent`	The center chosen inside the standard equilateral triangle.
`cent.name`	The name of the center inside the standard equilateral triangle.
`tri`	The triangle (it is the standard equilateral triangle for this function).
`x`	The input vector, can be a scalar or a `vector` of scalars, which constitute the x-coordinates of the point(s) of interest on the line.
`y`	The output vector, will be a scalar if `x` is a scalar or a `vector` of scalars if `x` is a `vector` of scalar, constitutes the y-coordinates of the point(s) of interest on the line.
`slope`	Slope of the line.
`intercept`	Intercept of the line.
`equation`	Equation of the line.

See Also

lineA2CMinTe and lineB2CMinTe

Examples

```
#Examples for lineA2MinTe
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)

M<-c(.65,.2)  #try also M<-c(1,1,1)

xfence<-abs(A[1]-B[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by = .1)  #try also by = .01

lnAM<-lineA2MinTe(x,M)
lnAM
summary(lnAM)
plot(lnAM)

Ds<-prj.cent2edges(Te,M)
#finds the projections from a point M=(m1,m2) to the edges on the
#extension of the lines joining M to the vertices in the triangle Te

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,pch=".",xlab="",ylab="",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
L<-Ds; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty = 3,col=2)

txt<-rbind(Te,M,Ds,c(.25,lineA2MinTe(.25,M)$y),c(.4,lineB2MinTe(.4,M)$y),
c(.60,lineC2MinTe(.60,M)$y))
xc<-txt[,1]+c(-.02,.02,.02,.02,.04,-.03,.0,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.05,.02,.03,-.03,0,0,0)
txt.str<-c("A","B","C","M","D1","D2","D3","lineA2MinTe(x)","lineB2MinTe(x)","lineC2MinTe(x)")
text(xc,yc,txt.str)

lineA2MinTe(.25,M)



#Examples for lineB2MinTe
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
```

```
M<-c(.65,.2)  #try also M<-c(1,1,1)

xfence<-abs(A[1]-B[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by = .5)  #try also by = .1

lnBM<-lineB2MinTe(x,M)
lnBM
summary(lnBM)
plot(lnBM)



#Examples for lineC2MinTe
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)

M<-c(.65,.2)  #try also M<-c(1,1,1)

xfence<-abs(A[1]-B[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by = .5)
#try also by = .1

lnCM<-lineC2MinTe(x,M)
lnCM
summary(lnCM)
plot(lnCM)
```

`funsCartBary` *Converts of a point in Cartesian coordinates to Barycentric coordinates and vice versa*

Description

Two functions: `cart2bary` and `bary2cart`.

`cart2bary` converts Cartesian coordinates of a given point `P`$= (x, y)$ to barycentric coordinates (in the normalized form) with respect to the triangle `tri`$= (v_1, v_2, v_3)$ with vertex labeling done row-wise in `tri` (i.e., row i corresponds to vertex v_i for $i = 1, 2, 3$).

`bary2cart` converts barycentric coordinates of the point `P`$= (t_1, t_2, t_3)$ (not necessarily normalized) to Cartesian coordinates according to the coordinates of the triangle, `tri`. For information on barycentric coordinates, see (Weisstein (2019)).

Usage

```
cart2bary(P, tri)

bary2cart(P, tri)
```

Arguments

`P`	A 2D point for `cart2bary`, and a `vector` of three `numeric` entries for `bary2cart`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

`cart2bary` returns the barycentric coordinates of a given point `P`$= (x, y)$ and `bary2cart` returns the Cartesian coordinates of the point `P`$= (t_1, t_2, t_3)$ (not necessarily normalized).

Author(s)

Elvan Ceyhan

References

Weisstein EW (2019). “Barycentric Coordinates.” From MathWorld — A Wolfram Web Resource, `http://mathworld.wolfram.com/BarycentricCoordinates.html`.

Examples

```
#Examples for cart2bary
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tr<-rbind(A,B,C)

cart2bary(A,Tr)
cart2bary(c(.3,.2),Tr)



#Examples for bary2cart
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tr<-rbind(A,B,C)

bary2cart(c(.3,.2,.5),Tr)
bary2cart(c(6,2,4),Tr)
```

`funsCSEdgeRegs` *Each function is for the presence of an arc from a point in one of the edge regions to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Three indicator functions: `IarcCSstd.triRAB`, `IarcCSstd.triRBC` and `IarcCSstd.triRAC`.

The function `IarcCSstd.triRAB` returns I(`p2` is in $N_{CS}(p1, t)$ for `p1` in RAB (edge region for edge AB, i.e., edge 3) in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$;

`IarcCSstd.triRBC` returns I(`p2` is in $N_{CS}(p1, t)$ for `p1` in RBC (edge region for edge BC, i.e., edge 1) in T_e; and

`IarcCSstd.triRAC` returns I(`p2` is in $N_{CS}(p1, t)$ for `p1` in RAC (edge region for edge AC, i.e., edge 2) in T_e. That is, each function returns 1 if `p2` is in $N_{CS}(p1, t)$, returns 0 otherwise.

CS proximity region is defined with respect to T_e whose vertices are also labeled as $T_e = T(v = 1, v = 2, v = 3)$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e

If `p1` and `p2` are distinct and `p1` is outside the corresponding edge region and `p2` is outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their location (i.e., it allows loops).

See also (Ceyhan (2005, 2010)).

Usage

```
IarcCSstd.triRAB(p1, p2, t, M)

IarcCSstd.triRBC(p1, p2, t, M)

IarcCSstd.triRAC(p1, p2, t, M)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e.

Value

Each function returns $I($`p2` is in $N_{CS}(p1, t))$ for `p1`, that is, returns 1 if `p2` is in $N_{CS}(p1, t)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

See Also

`IarcCSt1.std.triRAB`, `IarcCSt1.std.triRBC` and `IarcCSt1.std.triRAC`

Examples

```
#Examples for IarcCSstd.triRAB
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T3<-rbind(A,B,CM);

set.seed(1)
Xp<-runif.std.tri(3)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

t<-1

IarcCSstd.triRAB(Xp[1,],Xp[2,],t,M)
IarcCSstd.triRAB(c(.2,.5),Xp[2,],t,M)



#Examples for IarcCSstd.triRBC
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T1<-rbind(B,C,CM);

set.seed(1)
Xp<-runif.std.tri(3)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

t<-1

IarcCSstd.triRBC(Xp[1,],Xp[2,],t,M)
IarcCSstd.triRBC(c(.2,.5),Xp[2,],t,M)



#Examples for IarcCSstd.triRAC
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T2<-rbind(A,C,CM);

set.seed(1)
Xp<-runif.std.tri(3)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

t<-1

IarcCSstd.triRAC(Xp[1,],Xp[2,],t,M)
IarcCSstd.triRAC(c(.2,.5),Xp[2,],t,M)
```

---

funsCSGamTe *The function* `gammakCSstd.tri` *is for k ($k = 2, 3, 4, 5$) points constituting a dominating set for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Four indicator functions: `Idom.num2CSstd.tri`, `Idom.num3CSstd.tri`, `Idom.num4CSstd.tri`, `Idom.num5CSstd.tri` and `Idom.num6CSstd.tri`.

The function `gammakCSstd.tri` returns I({p1,...,pk} is a dominating set of the CS-PCD) where vertices of CS-PCD are the 2D data set `Xp`, that is, returns 1 if {p1,...,pk} is a dominating set of CS-PCD, returns 0 otherwise for $k = 2, 3, 4, 5, 6$.

CS proximity region is constructed with respect to $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $t > 0$ and edge regions are based on center of mass $CM = (1/2, \sqrt{3}/6)$.

`ch.data.pnts` is for checking whether points p1,...,pk are data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points p1,...,pk would be a dominating set if they actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num2CSstd.tri(p1, p2, Xp, t, ch.data.pnts = FALSE)

Idom.num3CSstd.tri(p1, p2, p3, Xp, t, ch.data.pnts = FALSE)

Idom.num4CSstd.tri(p1, p2, p3, p4, Xp, t, ch.data.pnts = FALSE)

Idom.num5CSstd.tri(p1, p2, p3, p4, p5, Xp, t, ch.data.pnts = FALSE)

Idom.num6CSstd.tri(p1, p2, p3, p4, p5, p6, Xp, t, ch.data.pnts = FALSE)
```

Arguments

`p1, p2, p3, p4, p5, p6`	The points $\{p1, \ldots, pk\}$ are k 2D points (for $k = 2, 3, 4, 5, 6$) to be tested for constituting a dominating set of the CS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the CS-PCD.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`ch.data.pnts`	A logical argument for checking whether points $\{p1, \ldots, pk\}$ are data points in `Xp` or not (default is `FALSE`).

Value

The function `gammakCSstd.tri` returns {`p1,...,pk`} is a dominating set of the CS-PCD) where vertices of the CS-PCD are the 2D data set `Xp`), that is, returns 1 if {`p1,...,pk`} is a dominating set of CS-PCD, returns 0 otherwise.

Author(s)

Elvan Ceyhan

See Also

`Idom.num1CSstd.tri`, `Idom.num2PEtri` and `Idom.num2PEtetra`

Examples

```
set.seed(123)
#Examples for Idom.num2CSstd.tri
t<-1.5
n<-10 #try also 10, 20 (it may take longer for larger n)

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

Idom.num2CSstd.tri(Xp[1,],Xp[2,],Xp,t)
Idom.num2CSstd.tri(c(.2,.2),Xp[2,],Xp,t)

ind.gam2<-vector()
for (i in 1:(n-1))
 for (j in (i+1):n)
 {if (Idom.num2CSstd.tri(Xp[i,],Xp[j,],Xp,t)==1)
  ind.gam2<-rbind(ind.gam2,c(i,j))}

ind.gam2



#Examples for Idom.num3CSstd.tri
t<-1.5
n<-10 #try also 10, 20 (it may take longer for larger n)

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

Idom.num3CSstd.tri(Xp[1,],Xp[2,],Xp[3,],Xp,t)

ind.gam3<-vector()
for (i in 1:(n-2))
 for (j in (i+1):(n-1))
   for (k in (j+1):n)
   {if (Idom.num3CSstd.tri(Xp[i,],Xp[j,],Xp[k,],Xp,t)==1)
    ind.gam3<-rbind(ind.gam3,c(i,j,k))}
```

```
ind.gam3



#Examples for Idom.num4CSstd.tri
t<-1.5
n<-10 #try also 10, 20 (it may take longer for larger n)

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

Idom.num4CSstd.tri(Xp[1,],Xp[2,],Xp[3,],Xp[4,],Xp,t)

ind.gam4<-vector()
for (i in 1:(n-3))
 for (j in (i+1):(n-2))
   for (k in (j+1):(n-1))
     for (l in (k+1):n)
     {if (Idom.num4CSstd.tri(Xp[i,],Xp[j,],Xp[k,],Xp[l,],Xp,t)==1)
      ind.gam4<-rbind(ind.gam4,c(i,j,k,l))}

ind.gam4

Idom.num4CSstd.tri(c(.2,.2),Xp[2,],Xp[3,],Xp[4,],Xp,t,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE since not all points are data points in Xp



#Examples for Idom.num5CSstd.tri
t<-1.5
n<-10 #try also 10, 20 (it may take longer for larger n)

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

Idom.num5CSstd.tri(Xp[1,],Xp[2,],Xp[3,],Xp[4,],Xp[5,],Xp,t)

ind.gam5<-vector()
for (i1 in 1:(n-4))
 for (i2 in (i1+1):(n-3))
   for (i3 in (i2+1):(n-2))
     for (i4 in (i3+1):(n-1))
       for (i5 in (i4+1):n)
       {if (Idom.num5CSstd.tri(Xp[i1,],Xp[i2,],Xp[i3,],Xp[i4,],Xp[i5,],Xp,t)==1)
        ind.gam5<-rbind(ind.gam5,c(i1,i2,i3,i4,i5))}

ind.gam5

Idom.num5CSstd.tri(c(.2,.2),Xp[2,],Xp[3,],Xp[4,],Xp[5,],Xp,t,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE since not all points are data points in Xp
```

```
#Examples for Idom.num6CSstd.tri
t<-1.5
n<-10 #try also 10, 20 (it may take longer for larger n)

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

Idom.num6CSstd.tri(Xp[1,],Xp[2,],Xp[3,],Xp[4,],Xp[5,],Xp[6,],Xp,t)

ind.gam6<-vector()
for (i1 in 1:(n-5))
 for (i2 in (i1+1):(n-4))
   for (i3 in (i2+1):(n-3))
     for (i4 in (i3+1):(n-2))
       for (i5 in (i4+1):(n-1))
         for (i6 in (i5+1):n)
       {if (Idom.num6CSstd.tri(Xp[i1,],Xp[i2,],Xp[i3,],Xp[i4,],Xp[i5,],Xp[i6,],Xp,t)==1)
          ind.gam6<-rbind(ind.gam6,c(i1,i2,i3,i4,i5,i6))}

ind.gam6

Idom.num6CSstd.tri(c(.2,.2),Xp[2,],Xp[3,],Xp[4,],Xp[5,],Xp[6,],Xp,t,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE since not all points are data points in Xp
```

`funsCSt1EdgeRegs` *Each function is for the presence of an arc from a point in one of the edge regions to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case with $t = 1$*

Description

Three indicator functions: `IarcCSt1.std.triRAB`, `IarcCSt1.std.triRBC` and `IarcCSt1.std.triRAC`.

The function `IarcCSt1.std.triRAB` returns $I($p2 is in $N_{CS}(p1, t = 1)$ for p1 in RAB (edge region for edge AB, i.e., edge 3) in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$;

`IarcCSt1.std.triRBC` returns $I($p2 is in $N_{CS}(p1, t = 1)$ for p1 in RBC (edge region for edge BC, i.e., edge 1) in T_e; and

`IarcCSt1.std.triRAC` returns $I($p2 is in $N_{CS}(p1, t = 1)$ for p1 in RAC (edge region for edge AC, i.e., edge 2) in T_e.

That is, each function returns 1 if p2 is in $N_{CS}(p1, t = 1)$, returns 0 otherwise, where $N_{CS}(x, t)$ is the CS proximity region for point x with expansion parameter $t = 1$.

Usage

```
IarcCSt1.std.triRAB(p1, p2)

IarcCSt1.std.triRBC(p1, p2)
```

```
IarcCSt1.std.triRAC(p1, p2)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.

Value

Each function returns $I($`p2` is in $N_{CS}(p1, t = 1))$ for `p1`, that is, returns 1 if `p2` is in $N_{CS}(p1, t = 1)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

See Also

`IarcCSstd.triRAB`, `IarcCSstd.triRBC` and `IarcCSstd.triRAC`

Examples

```
#Examples for IarcCSt1.std.triRAB
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T3<-rbind(A,B,CM);

set.seed(1)
Xp<-runif.std.tri(10)$gen.points

IarcCSt1.std.triRAB(Xp[1,],Xp[2,])

IarcCSt1.std.triRAB(c(.2,.5),Xp[2,])



#Examples for IarcCSt1.std.triRBC
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T1<-rbind(B,C,CM);

set.seed(1)
Xp<-runif.std.tri(3)$gen.points

IarcCSt1.std.triRBC(Xp[1,],Xp[2,])

IarcCSt1.std.triRBC(c(.2,.5),Xp[2,])
```

```
#Examples for IarcCSt1.std.triRAC
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T2<-rbind(A,C,CM);

set.seed(1)
Xp<-runif.std.tri(3)$gen.points

IarcCSt1.std.triRAC(Xp[1,],Xp[2,])
IarcCSt1.std.triRAC(c(1,2),Xp[2,])
```

`funsIndDelTri` *Functions provide the indices of the Delaunay triangles where the points reside*

Description

Two functions: `index.delaunay.tri` and `indices.delaunay.tri`.

`index.delaunay.tri` finds the index of the Delaunay triangle in which the given point, `p`, resides.

`indices.delaunay.tri` finds the indices of triangles for all the points in data set, `Xp`, as a vector.

Delaunay triangulation is based on `Yp` and `DTmesh` are the Delaunay triangles with default `NULL`. The function returns `NA` for a point not inside the convex hull of `Yp`. Number of `Yp` points (i.e., size of `Yp`) should be at least three and the points should be in general position so that Delaunay triangulation is (uniquely) defined.

If the number of `Yp` points is 3, then there is only one Delaunay triangle and the indices of all the points inside this triangle are all 1.

See (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
index.delaunay.tri(p, Yp, DTmesh = NULL)

indices.delaunay.tri(Xp, Yp, DTmesh = NULL)
```

Arguments

`p`	A 2D point; the index of the Delaunay triangle in which `p` resides is to be determined. It is an argument for `index.delaunay.tri`.
`Yp`	A set of 2D points from which Delaunay triangulation is constructed.
`DTmesh`	Delaunay triangles based on `Yp`, default is `NULL`, which is computed via `tri.mesh` function in `interp` package. `triangles` function yields a triangulation data structure from the triangulation object created by `tri.mesh`.
`Xp`	A set of 2D points representing the set of data points for which the indices of the Delaunay triangles they reside is to be determined. It is an argument for `indices.delaunay.tri`.

Value

`index.delaunay.tri` returns the index of the Delaunay triangle in which the given point, `p`, resides and `indices.delaunay.tri` returns the `vector` of indices of the Delaunay triangles in which points in the data set, `Xp`, reside.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

Examples

```
#Examples for index.delaunay.tri
nx<-20 #number of X points (target)
ny<-5 #number of Y points (nontarget)
set.seed(1)
Yp<-cbind(runif(ny),runif(ny))

Xp<-runif.multi.tri(nx,Yp)$g #data under CSR in the convex hull of Ypoints
#try also Xp<-cbind(runif(nx),runif(nx))

index.delaunay.tri(Xp[10,],Yp)

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")
#Delaunay triangulation
TRY<-interp::triangles(DTY)[,1:3];
index.delaunay.tri(Xp[10,],Yp,DTY)

ind.DT<-vector()
for (i in 1:nx)
 ind.DT<-c(ind.DT,index.delaunay.tri(Xp[i,],Yp))
ind.DT

Xlim<-range(Yp[,1],Xp[,1])
Ylim<-range(Yp[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")
#Delaunay triangulation based on Y points
```

```
#plot of the data in the convex hull of Y points together with the Delaunay triangulation
plot(Xp,main=" ", xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),type="n")
interp::plot.triSht(DTY, add=TRUE, do.points = TRUE,pch=16,col="blue")
points(Xp,pch=".",cex=3)
text(Xp,labels = factor(ind.DT))



#Examples for indices.delaunay.tri
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Yp<-cbind(runif(ny),runif(ny))
Xp<-runif.multi.tri(nx,Yp)$g #data under CSR in the convex hull of Ypoints
#try also Xp<-cbind(runif(nx),runif(nx))

tr.ind<-indices.delaunay.tri(Xp,Yp)  #indices of the Delaunay triangles
tr.ind

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")
#Delaunay triangulation based on Y points
tr.ind<-indices.delaunay.tri(Xp,Yp,DTY)  #indices of the Delaunay triangles
tr.ind

Xlim<-range(Yp[,1],Xp[,1])
Ylim<-range(Yp[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot of the data in the convex hull of Y points together with the Delaunay triangulation

oldpar <- par(pty = "s")
plot(Xp,main=" ", xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),pch=".")
interp::plot.triSht(DTY, add=TRUE, do.points = TRUE,pch=16,col="blue")
text(Xp,labels = factor(tr.ind))
par(oldpar)
```

`funsMuVarCS1D` *Returning the mean and (asymptotic) variance of arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - middle interval case*

Description

Two functions: `muCS1D` and `asy.varCS1D`.

`muCS1D` returns the mean of the (arc) density of CS-PCD and `asy.varCS1D` returns the (asymptotic) variance of the arc density of CS-PCD for a given centrality parameter $c \in (0,1)$ and an expansion parameter $t > 0$ and 1D uniform data in a finite interval (a,b), i.e., data from $U(a,b)$ distribution.

See also (Ceyhan (2016)).

Usage

```
muCS1D(t, c)

asy.varCS1D(t, c)
```

Arguments

`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$= (a,b)$. For the interval, `int`$= (a,b)$, the parameterized center is $M_c = a + c(b-a)$.

Value

`muCS1D` returns the mean and `asy.varCS1D` returns the asymptotic variance of the arc density of CS-PCD for uniform data in an interval

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`muPE1D` and `asy.varPE1D`

Examples

```
#Examples for muCS1D
muCS1D(1.2,.4)
muCS1D(1.2,.6)

tseq<-seq(0.01,5,by=.05)
cseq<-seq(0.01,.99,by=.05)

ltseq<-length(tseq)
lcseq<-length(cseq)

mu.grid<-matrix(0,nrow=ltseq,ncol=lcseq)
for (i in 1:ltseq)
```

```
    for (j in 1:lcseq)
    {
      mu.grid[i,j]<-muCS1D(tseq[i],cseq[j])
    }

  persp(tseq,cseq,mu.grid, xlab="t", ylab="c", zlab="mu(t,c)",theta = -30,
  phi = 30, expand = 0.5, col = "lightblue", ltheta = 120,
  shade = 0.05, ticktype = "detailed")



  #Examples for asy.varCS1D
  asy.varCS1D(1.2,.8)

  tseq<-seq(0.01,5,by=.05)
  cseq<-seq(0.01,.99,by=.05)

  ltseq<-length(tseq)
  lcseq<-length(cseq)

  var.grid<-matrix(0,nrow=ltseq,ncol=lcseq)
  for (i in 1:ltseq)
    for (j in 1:lcseq)
    {
      var.grid[i,j]<-asy.varCS1D(tseq[i],cseq[j])
    }

  persp(tseq,cseq,var.grid, xlab="t", ylab="c", zlab="var(t,c)", theta = -30,
  phi = 30, expand = 0.5, col = "lightblue", ltheta = 120,
  shade = 0.05, ticktype = "detailed")
```

`funsMuVarCS2D` *Returns the mean and (asymptotic) variance of arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 2D uniform data in one triangle*

Description

Two functions: `muCS2D` and `asy.varCS2D`.

`muCS2D` returns the mean of the (arc) density of CS-PCD and `asy.varCS2D` returns the asymptotic variance of the arc density of CS-PCD with expansion parameter $t > 0$ for 2D uniform data in a triangle.

CS proximity regions are defined with respect to the triangle and vertex regions are based on center of mass, CM of the triangle.

See also (Ceyhan (2005); Ceyhan et al. (2007)).

Usage

```
muCS2D(t)

asy.varCS2D(t)
```

Arguments

`t`	A positive real number which serves as the expansion parameter in CS proximity region.

Value

`muCS2D` returns the mean and `asy.varCS2D` returns the (asymptotic) variance of the arc density of CS-PCD for uniform data in any triangle

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`muPE2D` and `asy.varPE2D`

Examples

```
#Examples for muCS2D
muCS2D(.5)

tseq<-seq(0.01,5,by=.1)
ltseq<-length(tseq)

mu<-vector()
for (i in 1:ltseq)
{
  mu<-c(mu,muCS2D(tseq[i]))
}

plot(tseq, mu,type="l",xlab="t",ylab=expression(mu(t)),lty=1,xlim=range(tseq))



#Examples for asy.varCS2D
```

```
asy.varCS2D(.5)

tseq<-seq(0.01,10,by=.1)
ltseq<-length(tseq)

asy.var<-vector()
for (i in 1:ltseq)
{
  asy.var<-c(asy.var,asy.varCS2D(tseq[i]))
}

oldpar <- par(mar=c(5,5,4,2))
plot(tseq, asy.var,type="l",xlab="t",
    ylab=expression(paste(sigma^2,"(t)")),lty=1,xlim=range(tseq))
par(oldpar)
```

`funsMuVarCSend.int` *Returns the mean and (asymptotic) variance of arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - end-interval case*

Description

Two functions: `muCSend.int` and `asy.varCSend.int`.

`muCSend.int` returns the mean of the arc density of CS-PCD and `asy.varCSend.int` returns the asymptotic variance of the arc density of CS-PCD for a given expansion parameter $t > 0$ for 1D uniform data in the left and right end-intervals for the interval (a, b).

See also (Ceyhan (2016)).

Usage

```
muCSend.int(t)

asy.varCSend.int(t)
```

Arguments

`t`	A positive real number which serves as the expansion parameter in CS proximity region.

Details

funsMuVarCSend.int

Value

`muCSend.int` returns the mean and `asy.varCSend.int` returns the asymptotic variance of the arc density of CS-PCD for uniform data in end-intervals

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`muPEend.int` and `asy.varPEend.int`

Examples

```
#Examples for muCSend.int
muCSend.int(1.2)

tseq<-seq(0.01,5,by=.05)
ltseq<-length(tseq)

mu.end<-vector()
for (i in 1:ltseq)
{
  mu.end<-c(mu.end,muCSend.int(tseq[i]))
}

oldpar <- par(no.readonly = TRUE)
par(mar = c(5,4,4,2) + 0.1)
plot(tseq, mu.end,type="l",
ylab=expression(paste(mu,"(t)")),xlab="t",lty=1,xlim=range(tseq),ylim=c(0,1))
par(oldpar)

#Examples for asy.varCSend.int
asy.varCSend.int(1.2)

tseq<-seq(.01,5,by=.05)
ltseq<-length(tseq)

var.end<-vector()
for (i in 1:ltseq)
{
  var.end<-c(var.end,asy.varCSend.int(tseq[i]))
}

oldpar <- par(no.readonly = TRUE)
par(mar=c(5,5,4,2))
plot(tseq, var.end,type="l",xlab="t",ylab=expression(paste(sigma^2,"(t)")),lty=1,xlim=range(tseq))
par(oldpar)
```

`funsMuVarPE1D` *Returns the mean and (asymptotic) variance of arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - middle interval case*

Description

The functions `muPE1D` and `asy.varPE1D` and their auxiliary functions.

`muPE1D` returns the mean of the (arc) density of PE-PCD and `asy.varPE1D` returns the (asymptotic) variance of the arc density of PE-PCD for a given centrality parameter $c \in (0,1)$ and an expansion parameter $r \geq 1$ and for 1D uniform data in a finite interval (a,b), i.e., data from $U(a,b)$ distribution.

`muPE1D` uses auxiliary (internal) function `mu1PE1D` which yields mean (i.e., expected value) of the arc density of PE-PCD for a given $c \in (0,1/2)$ and $r \geq 1$.

`asy.varPE1D` uses auxiliary (internal) functions `fvar1` which yields asymptotic variance of the arc density of PE-PCD for $c \in (1/4,1/2)$ and $r \geq 1$; and `fvar2` which yields asymptotic variance of the arc density of PE-PCD for $c \in (0,1/4)$ and $r \geq 1$.

See also (Ceyhan (2012)).

Usage

```
mu1PE1D(r, c)

muPE1D(r, c)

fvar1(r, c)

fvar2(r, c)

asy.varPE1D(r, c)
```

Arguments

`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$. For the interval, (a,b), the parameterized center is $M_c = a + c(b-a)$.

Value

`muPE1D` returns the mean and `asy.varPE1D` returns the asymptotic variance of the arc density of PE-PCD for $U(a,b)$ data

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`muCS1D` and `asy.varCS1D`

Examples

```
#Examples for muPE1D
muPE1D(1.2,.4)
muPE1D(1.2,.6)

rseq<-seq(1.01,5,by=.1)
cseq<-seq(0.01,.99,by=.1)

lrseq<-length(rseq)
lcseq<-length(cseq)

mu.grid<-matrix(0,nrow=lrseq,ncol=lcseq)
for (i in 1:lrseq)
  for (j in 1:lcseq)
  {
    mu.grid[i,j]<-muPE1D(rseq[i],cseq[j])
  }

persp(rseq,cseq,mu.grid, xlab="r", ylab="c", zlab="mu(r,c)", theta = -30, phi = 30,
expand = 0.5, col = "lightblue", ltheta = 120, shade = 0.05, ticktype = "detailed")



#Examples for asy.varPE1D
asy.varPE1D(1.2,.8)

rseq<-seq(1.01,5,by=.1)
cseq<-seq(0.01,.99,by=.1)

lrseq<-length(rseq)
lcseq<-length(cseq)

var.grid<-matrix(0,nrow=lrseq,ncol=lcseq)
for (i in 1:lrseq)
  for (j in 1:lcseq)
  {
    var.grid[i,j]<-asy.varPE1D(rseq[i],cseq[j])
  }

persp(rseq,cseq,var.grid, xlab="r", ylab="c", zlab="var(r,c)", theta = -30, phi = 30,
expand = 0.5, col = "lightblue", ltheta = 120, shade = 0.05, ticktype = "detailed")
```

`funsMuVarPE2D` *Returns the mean and (asymptotic) variance of arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D uniform data in one triangle*

Description

Two functions: `muPE2D` and `asy.varPE2D`.

`muPE2D` returns the mean of the (arc) density of PE-PCD and `asy.varPE2D` returns the asymptotic variance of the arc density of PE-PCD for 2D uniform data in a triangle.

PE proximity regions are defined with expansion parameter $r \geq 1$ with respect to the triangle in which the points reside and vertex regions are based on center of mass, CM of the triangle.

See also (Ceyhan et al. (2006)).

Usage

```
muPE2D(r)

asy.varPE2D(r)
```

Arguments

`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

Value

`muPE2D` returns the mean and `asy.varPE2D` returns the (asymptotic) variance of the arc density of PE-PCD for uniform data in any triangle.

Author(s)

Elvan Ceyhan

References

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`muCS2D` and `asy.varCS2D`

Examples

```
#Examples for muPE2D
muPE2D(1.2)

rseq<-seq(1.01,5,by=.05)
lrseq<-length(rseq)

mu<-vector()
for (i in 1:lrseq)
{
  mu<-c(mu,muPE2D(rseq[i]))
}

plot(rseq, mu,type="l",xlab="r",ylab=expression(mu(r)),lty=1,
xlim=range(rseq),ylim=c(0,1))



#Examples for asy.varPE2D
asy.varPE2D(1.2)

rseq<-seq(1.01,5,by=.05)
lrseq<-length(rseq)

avar<-vector()
for (i in 1:lrseq)
{
  avar<-c(avar,asy.varPE2D(rseq[i]))
}

oldpar <- par(mar=c(5,5,4,2))
plot(rseq, avar,type="l",xlab="r",
ylab=expression(paste(sigma^2,"(r)")),lty=1,xlim=range(rseq))
par(oldpar)
```

`funsMuVarPEend.int` *Returns the mean and (asymptotic) variance of arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - end-interval case*

Description

Two functions: `muPEend.int` and `asy.varPEend.int`.

`muPEend.int` returns the mean of the arc density of PE-PCD and `asy.varPEend.int` returns the asymptotic variance of the arc density of PE-PCD for a given expansion parameter $r \geq 1$ for 1D uniform data in the left and right end-intervals for the interval (a, b).

See also (Ceyhan (2012)).

Usage

```
muPEend.int(r)

asy.varPEend.int(r)
```

Arguments

`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

Value

`muPEend.int` returns the mean and `asy.varPEend.int` returns the asymptotic variance of the arc density of PE-PCD for uniform data in end-intervals

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`muCSend.int` and `asy.varCSend.int`

Examples

```
#Examples for muPEend.int
muPEend.int(1.2)

rseq<-seq(1.01,5,by=.1)
lrseq<-length(rseq)

mu.end<-vector()
for (i in 1:lrseq)
{
  mu.end<-c(mu.end,muPEend.int(rseq[i]))
}

plot(rseq, mu.end,type="l",
ylab=expression(paste(mu,"(r)")),xlab="r",lty=1,xlim=range(rseq),ylim=c(0,1))



#Examples for asy.varPEend.int
asy.varPEend.int(1.2)

rseq<-seq(1.01,5,by=.1)
```

```
lrseq<-length(rseq)

var.end<-vector()
for (i in 1:lrseq)
{
  var.end<-c(var.end,asy.varPEend.int(rseq[i]))
}

oldpar <- par(mar=c(5,5,4,2))
plot(rseq, var.end,type="l",
xlab="r",ylab=expression(paste(sigma^2,"(r)")),lty=1,xlim=range(rseq))
par(oldpar)
```

`funsPDomNum2PE1D` *The functions for probability of domination number $= 2$ for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - middle interval case*

Description

The function `Pdom.num2PE1D` and its auxiliary functions.

Returns $P(\gamma = 2)$ for PE-PCD whose vertices are a uniform data set of size `n` in a finite interval (a, b) where γ stands for the domination number.

The PE proximity region $N_{PE}(x, r, c)$ is defined with respect to (a, b) with centrality parameter $c \in (0, 1)$ and expansion parameter $r \geq 1$.

To compute the probability $P(\gamma = 2)$ for PE-PCD in the 1D case, we partition the domain $(r, c) = (1, \infty) \times (0, 1)$, and compute the probability for each partition set. The sample size (i.e., number of vertices or data points) is a positive integer, `n`.

Usage

```
Pdom.num2AI(r, c, n)

Pdom.num2AII(r, c, n)

Pdom.num2AIII(r, c, n)

Pdom.num2AIV(r, c, n)

Pdom.num2A(r, c, n)

Pdom.num2Asym(r, c, n)

Pdom.num2BIII(r, c, n)
```

```
Pdom.num2B(r, c, n)

Pdom.num2Bsym(r, c, n)

Pdom.num2CIV(r, c, n)

Pdom.num2C(r, c, n)

Pdom.num2Csym(r, c, n)

Pdom.num2PE1D(r, c, n)
```

Arguments

`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$. For the interval, (a,b), the parameterized center is $M_c = a + c(b-a)$.
`n`	A positive integer representing the size of the uniform data set.

Value

$P(\text{domination number} \leq 1)$ for PE-PCD whose vertices are a uniform data set of size `n` in a finite interval (a,b)

Auxiliary Functions for `Pdom.num2PE1D`

The auxiliary functions are `Pdom.num2AI`, `Pdom.num2AII`, `Pdom.num2AIII`, `Pdom.num2AIV`, `Pdom.num2A`, `Pdom.num2Asym`, `Pdom.num2BIII`, `Pdom.num2B`, `Pdom.num2B`,`Pdom.num2Bsym`, `Pdom.num2CIV`, `Pdom.num2C`, and `Pdom.num2Csym`, each corresponding to a partition of the domain of `r` and `c`. In particular, the domain partition is handled in 3 cases as

CASE A: $c \in ((3-\sqrt{5})/2, 1/2)$

CASE B: $c \in (1/4, (3-\sqrt{5})/2)$ and

CASE C: $c \in (0, 1/4)$.

Case A - $c \in ((3-\sqrt{5})/2, 1/2)$

In Case A, we compute $P(\gamma = 2)$ with

`Pdom.num2AIV(r,c,n)` if $1 < r < (1-c)/c$;

`Pdom.num2AIII(r,c,n)` if $(1-c)/c < r < 1/(1-c)$;

`Pdom.num2AII(r,c,n)` if $1/(1-c) < r < 1/c$;

and `Pdom.num2AI(r,c,n)` otherwise.

`Pdom.num2A(r,c,n)` combines these functions in Case A: $c \in ((3-\sqrt{5})/2, 1/2)$. Due to the symmetry in the PE proximity regions, we use `Pdom.num2Asym(r,c,n)` for c in $(1/2, (\sqrt{5}-1)/2)$ with the same auxiliary functions

`Pdom.num2AIV(r,1-c,n)` if $1 < r < c/(1-c)$;

`Pdom.num2AIII(r,1-c,n)` if $(c/(1-c) < r < 1/c$;

`Pdom.num2AII(r,1-c,n)` if $1/c < r < 1/(1-c)$;

and `Pdom.num2AI(r,1-c,n)` otherwise.

Case B - $c \in (1/4, (3-\sqrt{5})/2)$

In Case B, we compute $P(\gamma = 2)$ with

`Pdom.num2AIV(r,c,n)` if $1 < r < 1/(1-c)$;

`Pdom.num2BIII(r,c,n)` if $1/(1-c) < r < (1-c)/c$;

`Pdom.num2AII(r,c,n)` if $(1-c)/c < r < 1/c$;

and `Pdom.num2AI(r,c,n)` otherwise.

`Pdom.num2B(r,c,n)` combines these functions in Case B: $c \in (1/4, (3-\sqrt{5})/2)$. Due to the symmetry in the PE proximity regions, we use `Pdom.num2Bsym(r,c,n)` for `c` in $((\sqrt{5}-1)/2, 3/4)$ with the same auxiliary functions

`Pdom.num2AIV(r,1-c,n)` if $1 < r < 1/c$;

`Pdom.num2BIII(r,1-c,n)` if $1/c < r < c/(1-c)$;

`Pdom.num2AII(r,1-c,n)` if $c/(1-c) < r < 1/(1-c)$;

and `Pdom.num2AI(r,1-c,n)` otherwise.

Case C - $c \in (0, 1/4)$

In Case C, we compute $P(\gamma = 2)$ with

`Pdom.num2AIV(r,c,n)` if $1 < r < 1/(1-c)$;

`Pdom.num2BIII(r,c,n)` if $1/(1-c) < r < (1-\sqrt{1-4c})/(2c)$;

`Pdom.num2CIV(r,c,n)` if $(1-\sqrt{1-4c})/(2c) < r < (1+\sqrt{1-4c})/(2c)$;

`Pdom.num2BIII(r,c,n)` if $(1+\sqrt{1-4c})/(2c) < r < 1/(1-c)$;

`Pdom.num2AII(r,c,n)` if $1/(1-c) < r < 1/c$;

and `Pdom.num2AI(r,c,n)` otherwise.

`Pdom.num2C(r,c,n)` combines these functions in Case C: $c \in (0, 1/4)$. Due to the symmetry in the PE proximity regions, we use `Pdom.num2Csym(r,c,n)` for $c \in (3/4, 1)$ with the same auxiliary functions

`Pdom.num2AIV(r,1-c,n)` if $1 < r < 1/c$;

`Pdom.num2BIII(r,1-c,n)` if $1/c < r < (1-\sqrt{1-4(1-c)})/(2(1-c))$;

`Pdom.num2CIV(r,1-c,n)` if $(1-\sqrt{1-4(1-c)})/(2(1-c)) < r < (1+\sqrt{1-4(1-c)})/(2(1-c))$;

`Pdom.num2BIII(r,1-c,n)` if $(1+\sqrt{1-4(1-c)})/(2(1-c)) < r < c/(1-c)$;

`Pdom.num2AII(r,1-c,n)` if $c/(1-c) < r < 1/(1-c)$;

and `Pdom.num2AI(r,1-c,n)` otherwise.

Combining Cases A, B, and C, we get our main function `Pdom.num2PE1D` which computes $P(\gamma = 2)$ for any `(r,c)` in its domain.

Author(s)

Elvan Ceyhan

See Also

`Pdom.num2PEtri` and `Pdom.num2PE1Dasy`

Examples

```
#Examples for the main function Pdom.num2PE1D
r<-2
c<-.5

Pdom.num2PE1D(r,c,n=10)
Pdom.num2PE1D(r=1.5,c=1/1.5,n=100)
```

`funsRankOrderTe` *Returns the ranks and orders of points in decreasing distance to the edges of the triangle*

Description

Two functions: `rank.dist2edges.std.tri` and `order.dist2edges.std.tri`.

`rank.dist2edges.std.tri` finds the ranks of the distances of points in data, `Xp`, to the edges of the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$

`dec` is a logical argument, default is `TRUE`, so the ranks are for decreasing distances, if `FALSE` it will be in increasing distances.

`order.dist2edges.std.tri` finds the orders of the distances of points in data, `Xp`, to the edges of T_e. The arguments are as in `rank.dist2edges.std.tri`.

Usage

```
rank.dist2edges.std.tri(Xp, dec = TRUE)

order.dist2edges.std.tri(Xp, dec = TRUE)
```

Arguments

`Xp`	A set of 2D points representing the data set in which ranking in terms of the distance to the edges of T_e is performed.
`dec`	A logical argument indicating the how the ranking will be performed. If `TRUE`, the ranks are for decreasing distances, and if `FALSE` they will be in increasing distances, default is `TRUE`.

Value

A `list` with two elements

`distances`	Distances from data points to the edges of T_e
`dist.rank`	The ranks of the data points in decreasing distances to the edges of T_e

Author(s)

Elvan Ceyhan

Examples

```
#Examples for rank.dist2edges.std.tri
n<-10
set.seed(1)
Xp<-runif.std.tri(n)$gen.points

dec.dist<-rank.dist2edges.std.tri(Xp)
dec.dist
dec.dist.rank<-dec.dist[[2]]
#the rank of distances to the edges in decreasing order
dec.dist.rank

A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.0,.01),
ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(Xp,pch=".")
text(Xp,labels = factor(dec.dist.rank) )

inc.dist<-rank.dist2edges.std.tri(Xp,dec = FALSE)
inc.dist
inc.dist.rank<-inc.dist[[2]]
#the rank of distances to the edges in increasing order
inc.dist.rank
dist<-inc.dist[[1]] #distances to the edges of the std eq. triangle
dist

plot(A,pch=".",xlab="",ylab="",xlim=Xlim,ylim=Ylim)
polygon(Te)
points(Xp,pch=".",xlab="",ylab="", main="",xlim=Xlim+xd*c(-.05,.05),
ylim=Ylim+yd*c(-.05,.05))
text(Xp,labels = factor(inc.dist.rank))
```

```
#Examples for order.dist2edges.std.tri
n<-10
set.seed(1)
Xp<-runif.std.tri(n)$gen.points  #try also Xp<-cbind(runif(n),runif(n))

dec.dist<-order.dist2edges.std.tri(Xp)
dec.dist
dec.dist.order<-dec.dist[[2]]
#the order of distances to the edges in decreasing order
dec.dist.order

A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.01,.01),
ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(Xp,pch=".")
text(Xp[dec.dist.order,],labels = factor(1:n) )

inc.dist<-order.dist2edges.std.tri(Xp,dec = FALSE)
inc.dist
inc.dist.order<-inc.dist[[2]]
#the order of distances to the edges in increasing order
inc.dist.order
dist<-inc.dist[[1]] #distances to the edges of the std eq. triangle
dist
dist[inc.dist.order] #distances in increasing order

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),
ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(Xp,pch=".")
text(Xp[inc.dist.order,],labels = factor(1:n))
```

---

funsTbMid2CC — *Two functions* `lineD1CCinTb` *and* `lineD2CCinTb` *which are of class* `"TriLines"` *— The lines joining the midpoints of edges to the circumcenter (CC) in the standard basic triangle.*

Description

Returns the `equation`, `slope`, `intercept`, and y-coordinates of the lines joining D_1 and CC and also D_2 and CC, in the standard basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2))$ where c_1 is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$ and $D_1 = (B + C)/2$ and $D_2 = (A + C)/2$ are the midpoints of edges BC and AC.

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence standard basic triangle is useful for simulation studies under the uniformity hypothesis. x-coordinates are provided in `vector` `x`.

Usage

```
lineD1CCinTb(x, c1, c2)

lineD2CCinTb(x, c1, c2)
```

Arguments

`x`	A single scalar or a `vector` of scalars.
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Value

A `list` with the elements

`txt1`	Longer description of the line.
`txt2`	Shorter description of the line (to be inserted over the line in the plot).
`mtitle`	The `"main"` title for the plot of the line.
`cent`	The center chosen inside the standard equilateral triangle.
`cent.name`	The name of the center inside the standard basic triangle. It is `"CC"` for these two functions.
`tri`	The triangle (it is the standard basic triangle for this function).
`x`	The input vector, can be a scalar or a `vector` of scalars, which constitute the x-coordinates of the point(s) of interest on the line.
`y`	The output vector, will be a scalar if `x` is a scalar or a `vector` of scalars if `x` is a `vector` of scalar, constitutes the y-coordinates of the point(s) of interest on the line.
`slope`	Slope of the line.
`intercept`	Intercept of the line.
`equation`	Equation of the line.

Author(s)

Elvan Ceyhan

See Also

`lineA2CMinTe`, `lineB2CMinTe`, `lineA2MinTe`, `lineB2MinTe`, and `lineC2MinTe`

Examples

```
#Examples for lineD1CCinTb
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);  #the vertices of the standard basic triangle Tb

Tb<-rbind(A,B,C)

xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnD1CC<-lineD1CCinTb(x,c1,c2)
lnD1CC
summary(lnD1CC)
plot(lnD1CC)

CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter
CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

x1<-seq(0,1,by=.1)  #try also by=.01
y1<-lineD1CCinTb(x1,c1,c2)$y

Xlim<-range(Tb[,1],x1)
Ylim<-range(Tb[,2],y1)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.04,.03,.02,.09,-.08,0)
yc<-txt[,2]+c(.02,.02,.04,.08,.03,.03,-.05)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

lines(x1,y1,type="l",lty=2)
text(.8,.5,"lineD1CCinTb")

c1<-.4; c2<-.6;
x1<-seq(0,1,by=.1)  #try also by=.01
lineD1CCinTb(x1,c1,c2)
```

```
#Examples for lineD2CCinTb
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);  #the vertices of the standard basic triangle Tb

Tb<-rbind(A,B,C)

xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnD2CC<-lineD2CCinTb(x,c1,c2)
lnD2CC
summary(lnD2CC)
plot(lnD2CC)

CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter
CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

x2<-seq(0,1,by=.1)  #try also by=.01
y2<-lineD2CCinTb(x2,c1,c2)$y

Xlim<-range(Tb[,1],x1)
Ylim<-range(Tb[,2],y2)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.04,.03,.02,.09,-.08,0)
yc<-txt[,2]+c(.02,.02,.04,.08,.03,.03,-.05)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

lines(x2,y2,type="l",lty=2)
text(0,.5,"lineD2CCinTb")
```

`IarcASbasic.tri` *The indicator for the presence of an arc from a point to another for Arc Slice Proximity Catch Digraphs (AS-PCDs) - standard basic triangle case*

Description

Returns $I(p2 \in N_{AS}(p1))$ for points `p1` and `p2`, that is, returns 1 if $p2$ is in $N_{AS}(p1)$, returns 0 otherwise, where $N_{AS}(x)$ is the AS proximity region for point x.

AS proximity region is constructed in the standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle T_b or based on circumcenter of T_b; default is `M="CC"`, i.e., circumcenter of T_b. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside T_b, the function returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence standard basic triangle is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
IarcASbasic.tri(p1, p2, c1, c2, M = "CC", rv = NULL)
```

Arguments

`p1`	A 2D point whose AS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the AS proximity region of `p1` or not.
`c1, c2`	Positive real numbers representing the top vertex in standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	The center of the triangle. `"CC"` stands for circumcenter or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of T_b.
`rv`	The index of the `M`-vertex region in T_b containing the point, either `1,2,3` or `NULL` (default is `NULL`).

Value

$I(p2 \in N_{AS}(p1))$ for points `p1` and `p2`, that is, returns 1 if $p2$ is in $N_{AS}(p1)$ (i.e., if there is an arc from `p1` to `p2`), returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`IarcAStri` and `NAStri`

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.2)

P1<-as.numeric(runif.basic.tri(1,c1,c2)$g)
P2<-as.numeric(runif.basic.tri(1,c1,c2)$g)
IarcASbasic.tri(P1,P2,c1,c2,M)

P1<-c(.3,.2)
P2<-c(.6,.2)
IarcASbasic.tri(P1,P2,c1,c2,M)

#or try
Rv<-rel.vert.basic.triCC(P1,c1,c2)$rv
IarcASbasic.tri(P1,P2,c1,c2,M,Rv)

P1<-c(.3,.2)
P2<-c(.8,.2)
IarcASbasic.tri(P1,P2,c1,c2,M)

P3<-c(.5,.4)
IarcASbasic.tri(P1,P3,c1,c2,M)

P4<-c(1.5,.4)
IarcASbasic.tri(P1,P4,c1,c2,M)
IarcASbasic.tri(P4,P4,c1,c2,M)

c1<-.4; c2<-.6;
P1<-c(.3,.2)
P2<-c(.6,.2)
IarcASbasic.tri(P1,P2,c1,c2,M)
```

`IarcASset2pnt.tri` *The indicator for the presence of an arc from a point in set* `S` *to the point* `p` *for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns I($pt \in N_{AS}(x)$ for some $x \in S$), that is, returns 1 if p is in $\cup_{x \in S} N_{AS}(x)$, returns 0 otherwise, where $N_{AS}(x)$ is the AS proximity region for point x.

AS proximity regions are constructed with respect to the triangle, `tri`$= T(A,B,C) =$`(rv=1,rv=2,rv=3)`, and vertices of `tri` are also labeled as 1,2, and 3, respectively.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`.

If `p` is not in `S` and either `p` or all points in `S` are outside `tri`, it returns 0, but if `p` is in `S`, then it always returns 1 (i.e., loops are allowed).

See also (Ceyhan (2005, 2010)).

Usage

```
IarcASset2pnt.tri(S, p, tri, M = "CC")
```

Arguments

`S`	A set of 2D points whose AS proximity regions are considered.
`p`	A 2D point. The function determines whether `p` is inside the union of AS proximity regions of points in `S` or not.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M="CC"` i.e., the circumcenter of `tri`.

Value

$I(pt \in \cup_{xinS} N_{AS}(x,r))$, that is, returns 1 if `p` is in `S` or inside $N_{AS}(x)$ for at least one x in `S`, returns 0 otherwise, where AS proximity region is constructed in `tri`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`IarcAStri`, `IarcASset2pnt.tri`, and `IarcCSset2pnt.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$gen.points

S<-rbind(Xp[1,],Xp[2,])  #try also S<-c(1.5,1)

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

IarcASset2pnt.tri(S,Xp[3,],Tr,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,])
IarcASset2pnt.tri(S,Xp[3,],Tr,M)

IarcASset2pnt.tri(S,Xp[6,],Tr,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IarcASset2pnt.tri(S,Xp[3,],Tr,M)

IarcASset2pnt.tri(c(.2,.5),Xp[2,],Tr,M)
IarcASset2pnt.tri(Xp,c(.2,.5),Tr,M)
IarcASset2pnt.tri(Xp,Xp[2,],Tr,M)
IarcASset2pnt.tri(c(.2,.5),c(.2,.5),Tr,M)
IarcASset2pnt.tri(Xp[5,],Xp[2,],Tr,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,],c(.2,.5))
IarcASset2pnt.tri(S,Xp[3,],Tr,M)

P<-c(.4,.2)
S<-Xp[c(1,3,4),]
IarcASset2pnt.tri(Xp,P,Tr,M)
```

```
IarcASset2pnt.tri(S,P,Tr,M)

IarcASset2pnt.tri(rbind(S,S),P,Tr,M)
```

`IarcAStri` *The indicator for the presence of an arc from a point to another for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns $I(p2 \in N_{AS}(p1))$ for points `p1` and `p2`, that is, returns 1 if $p2$ is in $N_{AS}(p1)$, returns 0 otherwise, where $N_{AS}(x)$ is the AS proximity region for point x.

AS proximity regions are constructed with respect to the triangle, `tri`$= T(A,B,C) =$`(rv=1,rv=2,rv=3)`, and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside `tri`, the function returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010)).

Usage

```
IarcAStri(p1, p2, tri, M = "CC", rv = NULL)
```

Arguments

`p1`	A 2D point whose AS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the AS proximity region of `p1` or not.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M="CC"` i.e., the circumcenter of `tri`.
`rv`	The index of the `M`-vertex region in `tri` containing the point, either `1,2,3` or `NULL` (default is `NULL`).

Value

$I(p2 \in N_{AS}(p1))$ for `p1`, that is, returns 1 if $p2$ is in $N_{AS}(p1)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`IarcASbasic.tri`, `IarcPEtri`, and `IarcCStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

P1<-as.numeric(runif.tri(1,Tr)$g)
P2<-as.numeric(runif.tri(1,Tr)$g)
IarcAStri(P1,P2,Tr,M)

P1<-c(1.3,1.2)
P2<-c(1.8,.5)
IarcAStri(P1,P2,Tr,M)
IarcAStri(P1,P1,Tr,M)

#or try
Rv<-rel.vert.triCC(P1,Tr)$rv
IarcAStri(P1,P2,Tr,M,Rv)

P3<-c(1.6,1.4)
IarcAStri(P1,P3,Tr,M)

P4<-c(1.5,1.0)
IarcAStri(P1,P4,Tr,M)

P5<-c(.5,1.0)
IarcAStri(P1,P5,Tr,M)
IarcAStri(P5,P5,Tr,M)
```

```
#or try
Rv<-rel.vert.triCC(P5,Tr)$rv
IarcAStri(P5,P5,Tr,M,Rv)
```

`IarcCS.Te.onesixth` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - first one-sixth of the standard equilateral triangle case*

Description

Returns $I($`p2` is in $N_{CS}(p1, t = 1))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{CS}(p1, t = 1)$, returns 0 otherwise, where $N_{CS}(x, t = 1)$ is the CS proximity region for point x with expansion parameter $t = 1$.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ and edge regions are based on the center of mass $CM = (1/2, \sqrt{3}/6)$. Here `p1` must lie in the first one-sixth of T_e, which is the triangle with vertices $T(A, D_3, CM) = T((0, 0), (1/2, 0), CM)$. If `p1` and `p2` are distinct and `p1` is outside of $T(A, D_3, CM)$ or `p2` is outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Usage

```
IarcCS.Te.onesixth(p1, p2)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.

Value

$I($`p2` is in $N_{CS}(p1, t = 1))$ for `p1` in the first one-sixth of T_e, $T(A, D_3, CM)$, that is, returns 1 if `p2` is in $N_{CS}(p1, t = 1)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

See Also

`IarcCSstd.tri`

IarcCSbasic.tri

The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard basic triangle case

Description

Returns $I($`p2` is in $N_{CS}(p1,t))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{CS}(p1,t)$, returns 0 otherwise, where $N_{CS}(x,t)$ is the CS proximity region for point x with expansion parameter $r \geq 1$.

CS proximity region is defined with respect to the standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Edge regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle T_b; default is $M = (1,1,1)$ i.e., the center of mass of T_b. `re` is the index of the edge region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside T_b, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation, and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence standard basic triangle is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
IarcCSbasic.tri(p1, p2, t, c1, c2, M = c(1, 1, 1), re = NULL)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.
`t`	A positive real number which serves as the expansion parameter in CS proximity region; must be ≥ 1
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle adjacent to the shorter edges; c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle or circumcenter of T_b; default is $M = (1,1,1)$ i.e., the center of mass of T_b.
`re`	The index of the edge region in T_b containing the point, either `1,2,3` or `NULL` (default is `NULL`).

Value

$I($p2 is in $N_{CS}(p1,t))$ for points p1 and p2, that is, returns 1 if p2 is in $N_{CS}(p1,t)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`IarcCStri` and `IarcCSstd.tri`

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)

tau<-2

P1<-as.numeric(runif.basic.tri(1,c1,c2)$g)
P2<-as.numeric(runif.basic.tri(1,c1,c2)$g)
IarcCSbasic.tri(P1,P2,tau,c1,c2,M)

P1<-c(.4,.2)
P2<-c(.5,.26)
IarcCSbasic.tri(P1,P2,tau,c1,c2,M)
IarcCSbasic.tri(P1,P1,tau,c1,c2,M)

#or try
Re<-rel.edge.basic.tri(P1,c1,c2,M)$re
IarcCSbasic.tri(P1,P2,tau,c1,c2,M,Re)
IarcCSbasic.tri(P1,P1,tau,c1,c2,M,Re)
```

IarcCSedge.reg.std.tri

The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case

---

Description

Returns $I($`p2` is in $N_{CS}(p1,t))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{CS}(p1,t)$, returns 0 otherwise, where $N_{CS}(x,t)$ is the CS proximity region for point x with expansion parameter $t>0$. This function is equivalent to `IarcCSstd.tri`, except that it computes the indicator using the functions `IarcCSstd.triRAB`, `IarcCSstd.triRBC` and `IarcCSstd.triRAC` which are edge-region specific indicator functions. For example, `IarcCSstd.triRAB` computes $I($`p2` is in $N_{CS}(p1,t))$ for points `p1` and `p2` when `p1` resides in the edge region of edge AB.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(v=1,v=2,v=3) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ and edge regions are based on the center $M=(m_1,m_2)$ in Cartesian coordinates or $M=(\alpha,\beta,\gamma)$ in barycentric coordinates in the interior of T_e; default is $M=(1,1,1)$ i.e., the center of mass of T_e. `re` is the index of the edge region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
IarcCSedge.reg.std.tri(p1, p2, t, M = c(1, 1, 1), re = NULL)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M=(1,1,1)$ i.e. the center of mass of T_e.
`re`	The index of the edge region in T_e containing the point, either `1,2,3` or `NULL` (default is `NULL`).

Value

$I($`p2` is in $N_{CS}(p1,t))$ for `p1`, that is, returns 1 if `p2` is in $N_{CS}(p1,t)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). “Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering.” *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`IarcCStri` and `IarcPEstd.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-3

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

t<-1
IarcCSedge.reg.std.tri(Xp[1,],Xp[2,],t,M)
IarcCSstd.tri(Xp[1,],Xp[2,],t,M)

#or try
re<-rel.edge.std.triCM(Xp[1,])$re
IarcCSedge.reg.std.tri(Xp[1,],Xp[2,],t,M,re=re)
```

`IarcCSend.int` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - end-interval case*

Description

Returns $I(p_2 \text{ in } N_{CS}(p_1,t))$ for points p_1 and p_2, that is, returns 1 if p_2 is in $N_{CS}(p_1,t)$, returns 0 otherwise, where $N_{CS}(x,t)$ is the CS proximity region for point x with expansion parameter $t > 0$ for the region outside the interval (a,b).

`rv` is the index of the end vertex region p_1 resides, with default=`NULL`, and `rv=1` for left end-interval and `rv=2` for the right end-interval. If p_1 and p_2 are distinct and either of them are inside interval `int`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2016)).

Usage

```
IarcCSend.int(p1, p2, int, t, rv = NULL)
```

Arguments

`p1`	A 1D point for which the CS proximity region is constructed.
`p2`	A 1D point to check whether it is inside the proximity region or not.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`rv`	Index of the end-interval containing the point, either `1,2` or `NULL` (default=`NULL`).

Value

$I(p_2 \text{ in } N_{CS}(p_1,t))$ for points p_1 and p_2, that is, returns 1 if p_2 is in $N_{CS}(p_1,t)$ (i.e., if there is an arc from p_1 to p_2), returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`IarcCSmid.int`, `IarcPEmid.int`, and `IarcPEend.int`

Examples

```
a<-0; b<-10; int<-c(a,b)
t<-2

IarcCSend.int(15,17,int,t)
IarcCSend.int(15,15,int,t)
```

```
IarcCSend.int(1.5,17,int,t)
IarcCSend.int(1.5,1.5,int,t)

IarcCSend.int(-15,17,int,t)

IarcCSend.int(-15,-17,int,t)

a<-0; b<-10; int<-c(a,b)
t<-.5

IarcCSend.int(15,17,int,t)
```

`IarcCSint` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - one interval case*

Description

Returns $I(p_2 \text{ in } N_{CS}(p_1, t, c))$ for points p_1 and p_2, that is, returns 1 if p_2 is in $N_{CS}(p_1, t, c)$, returns 0 otherwise, where $N_{CS}(x, t, c)$ is the CS proximity region for point x with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$.

CS proximity region is constructed with respect to the interval (a, b). This function works whether p_1 and p_2 are inside or outside the interval `int`.

Vertex regions for middle intervals are based on the center associated with the centrality parameter $c \in (0, 1)$. If p_1 and p_2 are identical, then it returns 1 regardless of their locations (i.e., loops are allowed in the digraph).

See also (Ceyhan (2016)).

Usage

```
IarcCSint(p1, p2, int, t, c = 0.5)
```

Arguments

`p1`	A 1D point for which the proximity region is constructed.
`p2`	A 1D point for which it is checked whether it resides in the proximity region of p_1 or not.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

$I(p_2 \text{ in } N_{CS}(p_1,t,c))$ for p2, that is, returns 1 if p_2 in $N_{CS}(p_1,t,c)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`IarcCSmid.int`, `IarcCSend.int` and `IarcPEint`

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

IarcCSint(7,5,int,t,c)
IarcCSint(17,17,int,t,c)
IarcCSint(15,17,int,t,c)
IarcCSint(1,3,int,t,c)

IarcCSint(-17,17,int,t,c)

IarcCSint(3,5,int,t,c)
IarcCSint(3,3,int,t,c)
IarcCSint(4,5,int,t,c)
IarcCSint(a,5,int,t,c)

c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

IarcCSint(7,5,int,t,c)
```

`IarcCSmid.int` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - middle interval case*

Description

Returns $I(p_2 \text{ in } N_{CS}(p_1,t,c))$ for points p_1 and p_2, that is, returns 1 if p_2 is in $N_{CS}(p_1,t,c)$, returns 0 otherwise, where $N_{CS}(x,t,c)$ is the CS proximity region for point x and is constructed with expansion parameter $t > 0$ and centrality parameter $c \in (0,1)$ for the interval (a,b).

CS proximity regions are defined with respect to the middle interval `int` and vertex regions are based on the center associated with the centrality parameter $c \in (0,1)$. For the interval, `int`$= (a,b)$, the parameterized center is $M_c = a + c(b-a)$. `rv` is the index of the vertex region p_1 resides, with default=`NULL`.

If p_1 and p_2 are distinct and either of them are outside interval `int`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., loops are allowed in the digraph).

See also (Ceyhan (2016)).

Usage

```
IarcCSmid.int(p1, p2, int, t, c = 0.5, rv = NULL)
```

Arguments

`p1, p2`	1D points; p_1 is the point for which the proximity region, $N_{CS}(p_1,t,c)$ is constructed and p_2 is the point which the function is checking whether its inside $N_{CS}(p_1,t,c)$ or not.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$= (a,b)$ with the default `c=.5`. For the interval, `int`$= (a,b)$, the parameterized center is $M_c = a + c(b-a)$.
`rv`	Index of the end-interval containing the point, either `1,2` or `NULL` (default is `NULL`).

Value

$I(p_2 \text{ in } N_{CS}(p_1,t,c))$ for points p_1 and p_2 that is, returns 1 if p_2 is in $N_{CS}(p_1,t,c)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`IarcCSend.int`, `IarcPEmid.int`, and `IarcPEend.int`

Examples

```
c<-.5
t<-2
a<-0; b<-10; int<-c(a,b)

IarcCSmid.int(7,5,int,t,c)
IarcCSmid.int(7,7,int,t,c)
IarcCSmid.int(7,5,int,t,c=.4)

IarcCSmid.int(1,3,int,t,c)

IarcCSmid.int(9,11,int,t,c)

IarcCSmid.int(19,1,int,t,c)
IarcCSmid.int(19,19,int,t,c)

IarcCSmid.int(3,5,int,t,c)

#or try
Rv<-rel.vert.mid.int(3,int,c)$rv
IarcCSmid.int(3,5,int,t,c,rv=Rv)

IarcCSmid.int(7,5,int,t,c)
```

`IarcCSset2pnt.std.tri` *The indicator for the presence of an arc from a point in set* `S` *to the point* `p` *for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns $I($`p` in $N_{CS}(x,t)$ for some x in `S`), that is, returns 1 if `p` is in $\cup_{x in S} N_{CS}(x,t)$, returns 0 otherwise, CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A,B,C) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with the expansion parameter $t > 0$ and edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1,1,1)$ i.e., the center of mass of T_e (which is equivalent to circumcenter of T_e).

Edges of T_e, AB, BC, AC, are also labeled as edges 3, 1, and 2, respectively. If `p` is not in `S` and either `p` or all points in `S` are outside T_e, it returns 0, but if `p` is in `S`, then it always returns 1 regardless of its location (i.e., loops are allowed).

See also (Ceyhan (2012)).

Usage

```
IarcCSset2pnt.std.tri(S, p, t, M = c(1, 1, 1))
```

Arguments

S	A set of 2D points. Presence of an arc from a point in S to point p is checked by the function.
p	A 2D point. Presence of an arc from a point in S to point p is checked by the function.
t	A positive real number which serves as the expansion parameter in CS proximity region in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M=(1,1,1)$ i.e., the center of mass of T_e.

Value

$I($p is in $\cup_{xinS} N_{CS}(x,t))$, that is, returns 1 if p is in S or inside $N_{CS}(x,t)$ for at least one x in S, returns 0 otherwise. CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A,B,C) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with M-edge regions.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`IarcCSset2pnt.tri`, `IarcCSstd.tri`, `IarcCStri`, and `IarcPEset2pnt.std.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

t<-.5

S<-rbind(Xp[1,],Xp[2,])  #try also S<-c(.5,.5)
IarcCSset2pnt.std.tri(S,Xp[3,],t,M)
IarcCSset2pnt.std.tri(S,Xp[3,],t=1,M)
IarcCSset2pnt.std.tri(S,Xp[3,],t=1.5,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
```

```
IarcCSset2pnt.std.tri(S,Xp[3,],t,M)
```

`IarcCSset2pnt.tri` *The indicator for the presence of an arc from a point in set `S` to the point `p` for Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns I(`p` in $N_{CS}(x,t)$ for some x in `S`), that is, returns 1 if `p` in $\cup_{xinS} N_{CS}(x,t)$, returns 0 otherwise.

CS proximity region is constructed with respect to the triangle `tri` with the expansion parameter $t > 0$ and edge regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1,1,1)$ i.e., the center of mass of `tri`.

Edges of `tri`$= T(A,B,C)$, AB, BC, AC, are also labeled as edges 3, 1, and 2, respectively. If `p` is not in `S` and either `p` or all points in `S` are outside `tri`, it returns 0, but if `p` is in `S`, then it always returns 1 regardless of its location (i.e., loops are allowed).

Usage

```
IarcCSset2pnt.tri(S, p, tri, t, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points. Presence of an arc from a point in `S` to point `p` is checked by the function.
`p`	A 2D point. Presence of an arc from a point in `S` to point `p` is checked by the function.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region constructed in the triangle `tri`.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1,1,1)$ i.e., the center of mass of `tri`.

Value

I(`p` is in $\cup_{xinS} N_{CS}(x,t)$), that is, returns 1 if `p` is in `S` or inside $N_{CS}(x,t)$ for at least one x in `S`, returns 0 otherwise where CS proximity region is constructed with respect to the triangle `tri`

Author(s)

Elvan Ceyhan

See Also

`IarcCSset2pnt.std.tri`, `IarcCStri`, `IarcCSstd.tri`, `IarcASset2pnt.tri`, and `IarcPEset2pnt.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$gen.points

S<-rbind(Xp[1,],Xp[2,])  #try also S<-c(1.5,1)

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

tau<-.5

IarcCSset2pnt.tri(S,Xp[3,],Tr,tau,M)
IarcCSset2pnt.tri(S,Xp[3,],Tr,t=1,M)
IarcCSset2pnt.tri(S,Xp[3,],Tr,t=1.5,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IarcCSset2pnt.tri(S,Xp[3,],Tr,tau,M)
```

`IarcCSstd.tri` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns $I($`p2` is in $N_{CS}(p1,t))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{CS}(p1,t)$, returns 0 otherwise, where $N_{CS}(x,t)$ is the CS proximity region for point x with expansion parameter $t > 0$.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1,1,1)$ i.e., the center of mass of T_e. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
IarcCSstd.tri(p1, p2, t, M = c(1, 1, 1), re = NULL)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.
`re`	The index of the edge region in T_e containing the point, either `1,2,3` or `NULL` (default is `NULL`).

Value

$I($`p2` is in $N_{CS}(p1, t))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{CS}(p1, t)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`IarcCStri`, `IarcCSbasic.tri`, and `IarcPEstd.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
n<-3

set.seed(1)
```

```
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2) or M=(A+B+C)/3

IarcCSstd.tri(Xp[1,],Xp[3,],t=2,M)
IarcCSstd.tri(c(0,1),Xp[3,],t=2,M)

#or try
Re<-rel.edge.tri(Xp[1,],Te,M) $re
IarcCSstd.tri(Xp[1,],Xp[3,],t=2,M,Re)
```

`IarcCSt1.std.tri` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case with $t = 1$*

Description

Returns $I($`p2` is in $N_{CS}(p1, t = 1))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{CS}(p1, t = 1)$, returns 0 otherwise, where $N_{CS}(x, t = 1)$ is the CS proximity region for point x with expansion parameter $t = 1$.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ and edge regions are based on the center of mass $CM = (1/2, \sqrt{3}/6)$.

If `p1` and `p2` are distinct and either are outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Usage

```
IarcCSt1.std.tri(p1, p2)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.

Value

$I($`p2` is in $N_{CS}(p1, t = 1))$ for `p1` in T_e that is, returns 1 if `p2` is in $N_{CS}(p1, t = 1)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

See Also

IarcCSstd.tri

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-3

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

IarcCSt1.std.tri(Xp[1,],Xp[2,])
IarcCSt1.std.tri(c(.2,.5),Xp[2,])
```

`IarcCStri` *The indicator for the presence of an arc from one point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs)*

Description

Returns $I($p2 is in $N_{CS}(p1,t))$ for points p1 and p2, that is, returns 1 if p2 is in $NCS(p1,t)$, returns 0 otherwise, where $N_{CS}(x,t)$ is the CS proximity region for point x with the expansion parameter $t > 0$.

CS proximity region is constructed with respect to the triangle `tri` and edge regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of `tri` or based on the circumcenter of `tri`. `re` is the index of the edge region `p` resides, with default=`NULL`

If `p1` and `p2` are distinct and either of them are outside `tri`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
IarcCStri(p1, p2, tri, t, M, re = NULL)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.

`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.
`re`	Index of the `M`-edge region containing the point `p`, either `1,2,3` or `NULL` (default is `NULL`).

Value

I(`p2` is in $NCS(p1,t)$) for `p1`, that is, returns 1 if `p2` is in $NCS(p1,t)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`IarcAStri`, `IarcPEtri`, `IarcCStri`, and `IarcCSstd.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
tau<-1.5

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

n<-10
set.seed(1)
Xp<-runif.tri(n,Tr)$g

IarcCStri(Xp[1,],Xp[2,],Tr,tau,M)

P1<-as.numeric(runif.tri(1,Tr)$g)
P2<-as.numeric(runif.tri(1,Tr)$g)
IarcCStri(P1,P2,Tr,tau,M)

#or try
re<-rel.edges.tri(P1,Tr,M)$re
IarcCStri(P1,P2,Tr,tau,M,re)
```

`IarcCStri.alt` *An alternative to the function `IarcCStri` which yields the indicator for the presence of an arc from one point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs)*

Description

Returns $I($`p2` is in $N_{CS}(p1,t))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{CS}(p1,t)$, returns 0 otherwise, where $N_{CS}(x,t)$ is the CS proximity region for point x with the expansion parameter $t > 0$.

CS proximity region is constructed with respect to the triangle `tri` and edge regions are based on the center of mass, CM. `re` is the index of the CM-edge region `p` resides, with default=`NULL` but must be provided as vertices (y_1, y_2, y_3) for $re = 3$ as rbind(y2,y3,y1) for $re = 1$ and as rbind(y1,y3,y2) for $re = 2$ for triangle $T(y_1, y_2, y_3)$.

If `p1` and `p2` are distinct and either of them are outside `tri`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
IarcCStri.alt(p1, p2, tri, t, re = NULL)
```

Arguments

`p1`	A 2D point whose CS proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the CS proximity region of `p1` or not.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`re`	Index of the CM-edge region containing the point `p`, either `1,2,3` or `NULL`, default=`NULL` but must be provided (row-wise) as vertices (y_1, y_2, y_3) for `re=3` as (y_2, y_3, y_1) for `re=1` and as (y_1, y_3, y_2) for `re=2` for triangle $T(y_1, y_2, y_3)$.

Value

$I($`p2` is in $N_{CS}(p1,t))$ for `p1`, that is, returns 1 if `p2` is in $N_{CS}(p1,t)$, returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`IarcAStri`, `IarcPEtri`, `IarcCStri`, and `IarcCSstd.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.6,2);
Tr<-rbind(A,B,C);
t<-1.5

P1<-c(.4,.2)
P2<-c(1.8,.5)
IarcCStri(P1,P2,Tr,t,M=c(1,1,1))
IarcCStri.alt(P1,P2,Tr,t)

IarcCStri(P2,P1,Tr,t,M=c(1,1,1))
IarcCStri.alt(P2,P1,Tr,t)

#or try
re<-rel.edges.triCM(P1,Tr)$re
IarcCStri(P1,P2,Tr,t,M=c(1,1,1),re)
IarcCStri.alt(P1,P2,Tr,t,re)
```

`IarcPEbasic.tri` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard basic triangle case*

Description

Returns $I($`p2` is in $N_{PE}(p1,r))$ for points `p1` and `p2` in the standard basic triangle, that is, returns 1 if `p2` is in $N_{PE}(p1,r)$, and returns 0 otherwise, where $N_{PE}(x,r)$ is the PE proximity region for point x with expansion parameter $r \geq 1$.

PE proximity region is defined with respect to the standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle T_b or based on circumcenter of T_b; default is $M = (1, 1, 1)$, i.e., the center of mass of T_b. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside T_b, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2006)).

Usage

```
IarcPEbasic.tri(p1, p2, r, c1, c2, M = c(1, 1, 1), rv = NULL)
```

Arguments

`p1`	A 2D point whose PE proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the PE proximity region of `p1` or not.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2+c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle or circumcenter of T_b which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of T_b.
`rv`	The index of the vertex region in T_b containing the point, either `1,2,3` or `NULL` (default is `NULL`).

Value

$I($`p2` is in $N_{PE}(p1, r))$ for points `p1` and `p2` in the standard basic triangle, that is, returns 1 if `p2` is in $N_{PE}(p1, r)$, and returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`IarcPEtri` and `IarcPEstd.tri`

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)

r<-2

P1<-as.numeric(runif.basic.tri(1,c1,c2)$g)
P2<-as.numeric(runif.basic.tri(1,c1,c2)$g)
IarcPEbasic.tri(P1,P2,r,c1,c2,M)

P1<-c(.4,.2)
P2<-c(.5,.26)
IarcPEbasic.tri(P1,P2,r,c1,c2,M)
IarcPEbasic.tri(P2,P1,r,c1,c2,M)

#or try
Rv<-rel.vert.basic.tri(P1,c1,c2,M)$rv
IarcPEbasic.tri(P1,P2,r,c1,c2,M,Rv)
```

`IarcPEend.int` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - end-interval case*

Description

Returns $I(p_2 \in N_{PE}(p_1, r))$ for points p_1 and p_2, that is, returns 1 if p_2 is in $N_{PE}(p_1, r)$, returns 0 otherwise, where $N_{PE}(x, r)$ is the PE proximity region for point x with expansion parameter $r \geq 1$ for the region outside the interval (a, b).

`rv` is the index of the end vertex region p_1 resides, with default=`NULL`, and `rv=1` for left end-interval and `rv=2` for the right end-interval. If p_1 and p_2 are distinct and either of them are inside interval `int`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2012)).

Usage

```
IarcPEend.int(p1, p2, int, r, rv = NULL)
```

Arguments

`p1`	A 1D point whose PE proximity region is constructed.
`p2`	A 1D point. The function determines whether p_2 is inside the PE proximity region of p_1 or not.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`rv`	Index of the end-interval containing the point, either `1,2` or `NULL` (default is `NULL`).

Value

$I(p_2 \in N_{PE}(p_1, r))$ for points p_1 and p_2, that is, returns 1 if p_2 is in $N_{PE}(p_1, r)$ (i.e., if there is an arc from p_1 to p_2), returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`IarcPEmid.int`, `IarcCSmid.int`, and `IarcCSend.int`

Examples

```
a<-0; b<-10; int<-c(a,b)
r<-2

IarcPEend.int(15,17,int,r)
IarcPEend.int(1.5,17,int,r)
IarcPEend.int(-15,17,int,r)
```

`IarcPEint` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one interval case*

Description

Returns $I(p_2 \in N_{PE}(p_1, r, c))$ for points p_1 and p_2, that is, returns 1 if p_2 is in $N_{PE}(p_1, r, c)$, returns 0 otherwise, where $N_{PE}(x, r, c)$ is the PE proximity region for point x with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$.

PE proximity region is constructed with respect to the interval (a, b). This function works whether p_1 and p_2 are inside or outside the interval `int`.

Vertex regions for middle intervals are based on the center associated with the centrality parameter $c \in (0, 1)$. If p_1 and p_2 are identical, then it returns 1 regardless of their locations (i.e., loops are allowed in the digraph).

See also (Ceyhan (2012)).

Usage

```
IarcPEint(p1, p2, int, r, c = 0.5)
```

Arguments

`p1`	A 1D point for which the proximity region is constructed.
`p2`	A 1D point for which it is checked whether it resides in the proximity region of p_1 or not.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

$I(p_2 \in N_{PE}(p_1, r, c))$, that is, returns 1 if p_2 in $N_{PE}(p_1, r, c)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`IarcPEmid.int`, `IarcPEend.int` and `IarcCSint`

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

IarcPEint(7,5,int,r,c)
IarcPEint(15,17,int,r,c)
IarcPEint(1,3,int,r,c)
```

`IarcPEmid.int` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - middle interval case*

Description

Returns $I(p_2 \in N_{PE}(p_1, r, c))$ for points p_1 and p_2, that is, returns 1 if p_2 is in $N_{PE}(p_1, r, c)$, returns 0 otherwise, where $N_{PE}(x, r, c)$ is the PE proximity region for point x and is constructed with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$ for the interval (a, b).

PE proximity regions are defined with respect to the middle interval `int` and vertex regions are based on the center associated with the centrality parameter $c \in (0, 1)$. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$. `rv` is the index of the vertex region p_1 resides, with default=`NULL`. If p_1 and p_2 are distinct and either of them are outside interval `int`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., loops are allowed in the digraph).

See also (Ceyhan (2012, 2016)).

Usage

```
IarcPEmid.int(p1, x2, int, r, c = 0.5, rv = NULL)
```

Arguments

`p1, x2`	1D points; p_1 is the point for which the proximity region, $N_{PE}(p_1, r, c)$ is constructed and p_2 is the point which the function is checking whether its inside $N_{PE}(p_1, r, c)$ or not.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
`rv`	The index of the vertex region p_1 resides, with default=`NULL`.

Value

$I(p_2 \in N_{PE}(p_1, r, c))$ for points p_1 and p_2 that is, returns 1 if p_2 is in $N_{PE}(p_1, r, c)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`IarcPEend.int`, `IarcCSmid.int`, and `IarcCSend.int`

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

IarcPEmid.int(7,5,int,r,c)
IarcPEmid.int(1,3,int,r,c)
```

`IarcPEset2pnt.std.tri` *The indicator for the presence of an arc from a point in set* `S` *to the point* `p` *or Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case*

Description

Returns $I($`p` in $N_{PE}(x, r)$ for some x in `S`$)$ for `S`, in the standard equilateral triangle, that is, returns 1 if `p` is in $\cup_{x in S} N_{PE}(x, r)$, and returns 0 otherwise.

PE proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with the expansion parameter $r \geq 1$ and vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$, i.e., the center of mass of T_e (which is equivalent to the circumcenter for T_e).

Vertices of T_e are also labeled as 1, 2, and 3, respectively. If `p` is not in `S` and either `p` or all points in `S` are outside T_e, it returns 0, but if `p` is in `S`, then it always returns 1 regardless of its location (i.e., loops are allowed).

Usage

```
IarcPEset2pnt.std.tri(S, p, r, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points. Presence of an arc from a point in `S` to point `p` is checked by the function.
`p`	A 2D point. Presence of an arc from a point in `S` to point `p` is checked by the function.
`r`	A positive real number which serves as the expansion parameter in PE proximity region in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e., the center of mass of T_e.

Value

$I($p is in $U_{xinS} N_{PE}(x,r))$ for `S` in the standard equilateral triangle, that is, returns 1 if `p` is in `S` or inside $N_{PE}(x,r)$ for at least one x in `S`, and returns 0 otherwise. PE proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A,B,C) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with `M`-vertex regions

Author(s)

Elvan Ceyhan

See Also

`IarcPEset2pnt.tri`, `IarcPEstd.tri`, `IarcPEtri`, and `IarcCSset2pnt.std.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

r<-1.5

S<-rbind(Xp[1,],Xp[2,])  #try also S<-c(.5,.5)
IarcPEset2pnt.std.tri(S,Xp[3,],r,M)
IarcPEset2pnt.std.tri(S,Xp[3,],r=1,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,])
IarcPEset2pnt.std.tri(S,Xp[3,],r,M)
```

```
IarcPEset2pnt.std.tri(S,Xp[6,],r,M)
IarcPEset2pnt.std.tri(S,Xp[6,],r=1.25,M)

P<-c(.4,.2)
S<-Xp[c(1,3,4),]
IarcPEset2pnt.std.tri(Xp,P,r,M)
```

`IarcPEset2pnt.tri` *The indicator for the presence of an arc from a point in set* `S` *to the point* `p` *for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns $I($`p` in $N_{PE}(x,r)$ for some x in `S`), that is, returns 1 if `p` is in $\cup_{xinS} N_{PE}(x,r)$, and returns 0 otherwise.

PE proximity region is constructed with respect to the triangle `tri` with the expansion parameter $r \geq 1$ and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1,1,1)$, i.e., the center of mass of `tri`. Vertices of `tri` are also labeled as 1, 2, and 3, respectively.

If `p` is not in `S` and either `p` or all points in `S` are outside `tri`, it returns 0, but if `p` is in `S`, then it always returns 1 regardless of its location (i.e., loops are allowed).

Usage

```
IarcPEset2pnt.tri(S, p, tri, r, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points. Presence of an arc from a point in `S` to point `p` is checked by the function.
`p`	A 2D point. Presence of an arc from a point in `S` to point `p` is checked by the function.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region constructed in the triangle `tri`; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1,1,1)$, i.e., the center of mass of `tri`.

Value

$I($`p` is in $U_{xinS} N_{PE}(x,r))$, that is, returns 1 if `p` is in `S` or inside $N_{PE}(x,r)$ for at least one x in `S`, and returns 0 otherwise, where PE proximity region is constructed with respect to the triangle `tri`

Author(s)

Elvan Ceyhan

See Also

`IarcPEset2pnt.std.tri`, `IarcPEtri`, `IarcPEstd.tri`, `IarcASset2pnt.tri`, and `IarcCSset2pnt.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5

S<-rbind(Xp[1,],Xp[2,])  #try also S<-c(1.5,1)

IarcPEset2pnt.tri(S,Xp[3,],Tr,r,M)
IarcPEset2pnt.tri(S,Xp[3,],r=1,Tr,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,])
IarcPEset2pnt.tri(S,Xp[3,],Tr,r,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IarcPEset2pnt.tri(S,Xp[3,],Tr,r,M)

P<-c(.4,.2)
S<-Xp[c(1,3,4),]
IarcPEset2pnt.tri(Xp,P,Tr,r,M)
```

`IarcPEstd.tetra` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard regular tetrahedron case*

Description

Returns $I($`p2` is in $N_{PE}(p1, r))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{PE}(p1, r)$, returns 0 otherwise, where $N_{PE}(x, r)$ is the PE proximity region for point x with expansion parameter $r \geq 1$.

PE proximity region is defined with respect to the standard regular tetrahedron $T_h = T(v = 1, v = 2, v = 3, v = 4) = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3))$ and vertex regions are based on the circumcenter (which is equivalent to the center of mass for standard regular tetrahedron) of T_h. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside T_h, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010)).

Usage

```
IarcPEstd.tetra(p1, p2, r, rv = NULL)
```

Arguments

`p1`	A 3D point whose PE proximity region is constructed.
`p2`	A 3D point. The function determines whether `p2` is inside the PE proximity region of `p1` or not.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`rv`	Index of the vertex region containing the point, either `1,2,3,4` (default is `NULL`).

Value

$I($`p2` is in $N_{PE}(p1, r))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{PE}(p1, r)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`IarcPEtetra`, `IarcPEtri` and `IarcPEint`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-3  #try also n<-20
Xp<-runif.std.tetra(n)$g
r<-1.5
IarcPEstd.tetra(Xp[1,],Xp[3,],r)
IarcPEstd.tetra(c(.4,.4,.4),c(.5,.5,.5),r)

#or try
RV<-rel.vert.tetraCC(Xp[1,],tetra)$rv
IarcPEstd.tetra(Xp[1,],Xp[3,],r,rv=RV)

P1<-c(.1,.1,.1)
P2<-c(.5,.5,.5)
IarcPEstd.tetra(P1,P2,r)
```

`IarcPEstd.tri` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case*

Description

Returns $I($p2 is in $N_{PE}(p1,r))$ for points `p1` and `p2` in the standard equilateral triangle, that is, returns 1 if `p2` is in $N_{PE}(p1,r)$, and returns 0 otherwise, where $N_{PE}(x,r)$ is the PE proximity region for point x with expansion parameter $r \geq 1$.

PE proximity region is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha,\beta,\gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1,1,1)$, i.e., the center of mass of T_e. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
IarcPEstd.tri(p1, p2, r, M = c(1, 1, 1), rv = NULL)
```

Arguments

`p1`	A 2D point whose PE proximity region is constructed.

`p2`	A 2D point. The function determines whether `p2` is inside the PE proximity region of `p1` or not.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.
`rv`	The index of the vertex region in T_e containing the point, either `1,2,3` or `NULL` (default is `NULL`).

Value

$I($`p2` is in $N_{PE}(p1, r))$ for points `p1` and `p2` in the standard equilateral triangle, that is, returns 1 if `p2` is in $N_{PE}(p1, r)$, and returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`IarcPEtri`, `IarcPEbasic.tri`, and `IarcCSstd.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
n<-3

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

IarcPEstd.tri(Xp[1,],Xp[3,],r=1.5,M)
IarcPEstd.tri(Xp[1,],Xp[3,],r=2,M)

#or try
```

```
Rv<-rel.vert.std.triCM(Xp[1,])$rv
IarcPEstd.tri(Xp[1,],Xp[3,],r=2,rv=Rv)

P1<-c(.4,.2)
P2<-c(.5,.26)
r<-2
IarcPEstd.tri(P1,P2,r,M)
```

`IarcPEtetra` *The indicator for the presence of an arc from one 3D point to another 3D point for Proportional Edge Proximity Catch Digraphs (PE-PCDs)*

Description

Returns $I($`p2` is in $N_{PE}(p1,r))$ for 3D points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{PE}(p1,r)$, returns 0 otherwise, where $N_{PE}(x,r)$ is the PE proximity region for point x with the expansion parameter $r \geq 1$.

PE proximity region is constructed with respect to the tetrahedron `th` and vertex regions are based on the center `M` which is circumcenter (`"CC"`) or center of mass (`"CM"`) of `th` with default=`"CM"`. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside `th`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010)).

Usage

```
IarcPEtetra(p1, p2, th, r, M = "CM", rv = NULL)
```

Arguments

`p1`	A 3D point whose PE proximity region is constructed.
`p2`	A 3D point. The function determines whether `p2` is inside the PE proximity region of `p1` or not.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.
`rv`	Index of the `M`-vertex region containing the point, either `1,2,3,4` (default is `NULL`).

Value

$I($`p2` is in $N_{PE}(p1,r))$ for `p1`, that is, returns 1 if `p2` is in $N_{PE}(p1,r)$, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`IarcPEstd.tetra`, `IarcPEtri` and `IarcPEint`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-3  #try also n<-20

Xp<-runif.tetra(n,tetra)$g

M<-"CM"  #try also M<-"CC"
r<-1.5

IarcPEtetra(Xp[1,],Xp[2,],tetra,r)  #uses the default M="CM"
IarcPEtetra(Xp[1,],Xp[2,],tetra,r,M)

IarcPEtetra(c(.4,.4,.4),c(.5,.5,.5),tetra,r,M)

#or try
RV<-rel.vert.tetraCC(Xp[1,],tetra)$rv
IarcPEtetra(Xp[1,],Xp[3,],tetra,r,M,rv=RV)

P1<-c(.1,.1,.1)
P2<-c(.5,.5,.5)
IarcPEtetra(P1,P2,tetra,r,M)
```

`IarcPEtri` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns $I($`p2` is in $N_{PE}(p1, r))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{PE}(p1, r)$, and returns 0 otherwise, where $N_{PE}(x, r)$ is the PE proximity region for point x with the expansion parameter $r \geq 1$.

PE proximity region is constructed with respect to the triangle `tri` and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p1` and `p2` are distinct and either of them are outside `tri`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
IarcPEtri(p1, p2, tri, r, M = c(1, 1, 1), rv = NULL)
```

Arguments

`p1`	A 2D point whose PE proximity region is constructed.
`p2`	A 2D point. The function determines whether `p2` is inside the PE proximity region of `p1` or not.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.
`rv`	Index of the `M`-vertex region containing the point, either `1,2,3` or `NULL` (default is `NULL`).

Value

$I($`p2` is in $N_{PE}(p1, r))$ for points `p1` and `p2`, that is, returns 1 if `p2` is in $N_{PE}(p1, r)$, and returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`IarcPEbasic.tri`, `IarcPEstd.tri`, `IarcAStri`, and `IarcCStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0);

r<-1.5

n<-3
set.seed(1)
Xp<-runif.tri(n,Tr)$g

IarcPEtri(Xp[1,],Xp[2,],Tr,r,M)

P1<-as.numeric(runif.tri(1,Tr)$g)
P2<-as.numeric(runif.tri(1,Tr)$g)
IarcPEtri(P1,P2,Tr,r,M)

P1<-c(.4,.2)
P2<-c(1.8,.5)
IarcPEtri(P1,P2,Tr,r,M)
IarcPEtri(P2,P1,Tr,r,M)

M<-c(1.3,1.3)
r<-2

#or try
Rv<-rel.vert.tri(P1,Tr,M)$rv
IarcPEtri(P1,P2,Tr,r,M,Rv)
```

`Idom.num.up.bnd` *Indicator for an upper bound for the domination number by the exact algorithm*

Description

Returns 1 if the domination number is less than or equal to the prespecified value `k` and also the indices (i.e., row numbers) of a dominating set of size `k` based on the incidence matrix `Inc.Mat` of a graph or a digraph. Here the row number in the incidence matrix corresponds to the index of the vertex (i.e., index of the data point). The function works whether loops are allowed or not (i.e., whether the first diagonal is all 1 or all 0). It takes a rather long time for large number of vertices (i.e., large number of row numbers).

Usage

```
Idom.num.up.bnd(Inc.Mat, k)
```

Arguments

`Inc.Mat`	A square matrix consisting of 0's and 1's which represents the incidence matrix of a graph or digraph.
`k`	A positive integer for the upper bound (to be checked) for the domination number.

Value

A `list` with two elements

`dom.up.bnd`	The upper bound (to be checked) for the domination number. It is prespecified as `k` in the function arguments.
`Idom.num.up.bnd`	The indicator for the upper bound for domination number of the graph or digraph being the specified value `k` or not. It returns 1 if the upper bound is `k`, and 0 otherwise based on the incidence matrix `Inc.Mat` of the graph or digraph.
`ind.dom.set`	Indices of the rows in the incidence matrix `Inc.Mat` that correspond to the vertices in the dominating set of size `k` if it exists, otherwise it yields `NULL`.

Author(s)

Elvan Ceyhan

See Also

`dom.num.exact` and `dom.num.greedy`

Examples

```
n<-10
M<-matrix(sample(c(0,1),n^2,replace=TRUE),nrow=n)
diag(M)<-1

dom.num.greedy(M)
Idom.num.up.bnd(M,2)

for (k in 1:n)
print(c(k,Idom.num.up.bnd(M,k)))
```

`Idom.num1ASbasic.tri` *The indicator for a point being a dominating point for Arc Slice Proximity Catch Digraphs (AS-PCDs) - standard basic triangle case*

Description

Returns I(`p` is a dominating point of the AS-PCD) where the vertices of the AS-PCD are the 2D data set `Xp`, that is, returns 1 if `p` is a dominating point of AS-PCD, returns 0 otherwise. AS proximity regions are defined with respect to the standard basic triangle, T_b, c_1 is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence standard basic triangle is useful for simulation studies under the uniformity hypothesis.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle T_b or based on circumcenter of T_b; default is `M="CC"`, i.e., circumcenter of T_b. Point, `p`, is in the vertex region of vertex `rv` (default is `NULL`); vertices are labeled as $1, 2, 3$ in the order they are stacked row-wise.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num1ASbasic.tri(p, Xp, c1, c2, M = "CC", rv = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 2D point that is to be tested for being a dominating point or not of the AS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the AS-PCD.

`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2+c_2^2 \leq 1$.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle T_b or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of T_b.
`rv`	Index of the vertex whose region contains point `p`, `rv` takes the vertex labels as $1, 2, 3$ as in the row order of the vertices in T_b.
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

I(`p` is a dominating point of the AS-PCD) where the vertices of the AS-PCD are the 2D data set `Xp`, that is, returns 1 if `p` is a dominating point, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.num1AStri` and `Idom.num1PEbasic.tri`

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-10

set.seed(1)
Xp<-runif.basic.tri(n,c1,c2)$g

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.2)
```

```
Idom.num1ASbasic.tri(Xp[1,],Xp,c1,c2,M)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1ASbasic.tri(Xp[i,],Xp,c1,c2,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1

#or try
Rv<-rel.vert.basic.triCC(Xp[1,],c1,c2)$rv
Idom.num1ASbasic.tri(Xp[1,],Xp,c1,c2,M,Rv)

Idom.num1ASbasic.tri(c(.2,.4),Xp,c1,c2,M)
Idom.num1ASbasic.tri(c(.2,.4),c(.2,.4),c1,c2,M)

Xp2<-rbind(Xp,c(.2,.4))
Idom.num1ASbasic.tri(Xp[1,],Xp2,c1,c2,M)

CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter

if (dimension(M)==3) {M<-bary2cart(M,Tb)}
#need to run this when M is given in barycentric coordinates

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-prj.cent2edges.basic.tri(c1,c2,M)
}

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(Xp)
points(rbind(Xp[ind.gam1,]),pch=4,col=2)

txt<-rbind(Tb,cent,Ds)
xc<-txt[,1]+c(-.03,.03,.02,.06,.06,-0.05,.01)
yc<-txt[,2]+c(.02,.02,.03,.0,.03,.03,-.03)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
```

```
text(xc,yc,txt.str)

Idom.num1ASbasic.tri(c(.4,.2),Xp,c1,c2,M)

Idom.num1ASbasic.tri(c(.5,.11),Xp,c1,c2,M)

Idom.num1ASbasic.tri(c(.5,.11),Xp,c1,c2,M,ch.data.pnt=FALSE)
#gives an error message if ch.data.pnt=TRUE since the point is not in the standard basic triangle
```

`Idom.num1AStri` *The indicator for a point being a dominating point for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns I(`p` is a dominating point of the AS-PCD whose vertices are the 2D data set `Xp`), that is, returns 1 if `p` is a dominating point of AS-PCD, returns 0 otherwise. Point, `p`, is in the region of vertex `rv` (default is `NULL`); vertices are labeled as $1, 2, 3$ in the order they are stacked row-wise in `tri`.

AS proximity regions are defined with respect to the triangle `tri` and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num1AStri(p, Xp, tri, M = "CC", rv = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 2D point that is to be tested for being a dominating point or not of the AS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of `tri`.

`rv`	Index of the vertex whose region contains point `p`, `rv` takes the vertex labels as $1, 2, 3$ as in the row order of the vertices in `tri`.
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

I(`p` is a dominating point of the AS-PCD whose vertices are the 2D data set `Xp`), that is, returns 1 if `p` is a dominating point of the AS-PCD, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.num1ASbasic.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

Idom.num1AStri(Xp[1,],Xp,Tr,M)
Idom.num1AStri(Xp[1,],Xp[1,],Tr,M)
Idom.num1AStri(c(1.5,1.5),c(1.6,1),Tr,M)
Idom.num1AStri(c(1.6,1),c(1.5,1.5),Tr,M)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1AStri(Xp[i,],Xp,Tr,M))}
```

```
ind.gam1<-which(gam.vec==1)
ind.gam1

#or try
Rv<-rel.vert.triCC(Xp[1,],Tr)$rv
Idom.num1AStri(Xp[1,],Xp,Tr,M,Rv)

Idom.num1AStri(c(.2,.4),Xp,Tr,M)
Idom.num1AStri(c(.2,.4),c(.2,.4),Tr,M)

Xp2<-rbind(Xp,c(.2,.4))
Idom.num1AStri(Xp[1,],Xp2,Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

CC<-circumcenter.tri(Tr)  #the circumcenter

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-prj.cent2edges(Tr,M)
}

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp)
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(Xp[ind.gam1,]),pch=4,col=2)

txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)

Idom.num1AStri(c(1.5,1.1),Xp,Tr,M)

Idom.num1AStri(c(1.5,1.1),Xp,Tr,M)

Idom.num1AStri(c(1.5,1.1),Xp,Tr,M,ch.data.pnt=FALSE)
#gives an error message if ch.data.pnt=TRUE since point p is not a data point in Xp
```

`Idom.num1CS.Te.onesixth`

The indicator for a point being a dominating point for Central Similarity Proximity Catch Digraphs (CS-PCDs) - first one-sixth of the standard equilateral triangle case

---

Description

Returns $I($`p` is a dominating point of the 2D data set `Xp` of CS-PCD$)$ in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$, that is, returns 1 if `p` is a dominating point of CS-PCD, returns 0 otherwise.

Point, `p`, must lie in the first one-sixth of T_e, which is the triangle with vertices $T(A, D_3, CM) = T((0, 0), (1/2, 0), CM)$.

CS proximity region is constructed with respect to T_e with expansion parameter $t = 1$.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005)).

Usage

```
Idom.num1CS.Te.onesixth(p, Xp, ch.data.pnt = FALSE)
```

Arguments

`p`	A 2D point that is to be tested for being a dominating point or not of the CS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the CS-PCD.
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

$I($`p` is a dominating point of the CS-PCD$)$ where the vertices of the CS-PCD are the 2D data set `Xp`, that is, returns 1 if `p` is a dominating point, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

See Also

`Idom.num1CSstd.tri` and `Idom.num1CSt1std.tri`

`Idom.num1CSint` *The indicator for a point being a dominating point for Central Similarity Proximity Catch Digraphs (CS-PCDs) for an interval*

Description

Returns $I($`p` is a dominating point of CS-PCD$)$ where the vertices of the CS-PCD are the 1D data set `Xp`).

CS proximity region is defined with respect to the interval `int` with an expansion parameter, $t > 0$, and a centrality parameter, $c \in (0, 1)$, so arcs may exist for `Xp` points inside the interval `int`$= (a, b)$.

Vertex regions are based on the center associated with the centrality parameter $c \in (0, 1)$. `rv` is the index of the vertex region `p` resides, with default=`NULL`.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

Usage

```
Idom.num1CSint(p, Xp, int, t, c = 0.5, rv = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 1D point that is to be tested for being a dominating point or not of the CS-PCD.
`Xp`	A set of 1D points which constitutes the vertices of the CS-PCD.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
`rv`	Index of the vertex region in which the point resides, either `1,2` or `NULL` (default is `NULL`).
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

I(p is a dominating point of CS-PCD) where the vertices of the CS-PCD are the 1D data set `Xp`), that is, returns 1 if `p` is a dominating point, returns 0 otherwise

Author(s)

Elvan Ceyhan

See Also

`Idom.num1PEint`

Examples

```
t<-2
c<-.4
a<-0; b<-10; int<-c(a,b)

Mc<-centerMc(int,c)
n<-10

set.seed(1)
Xp<-runif(n,a,b)

Idom.num1CSint(Xp[5],Xp,int,t,c)

Idom.num1CSint(2,Xp,int,t,c,ch.data.pnt = FALSE)
#gives an error if ch.data.pnt = TRUE since p is not a data point in Xp

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1CSint(Xp[i],Xp,int,t,c))}

ind.gam1<-which(gam.vec==1)
ind.gam1

domset<-Xp[ind.gam1]
if (length(ind.gam1)==0)
{domset<-NA}

#or try
Rv<-rel.vert.mid.int(Xp[5],int,c)$rv
Idom.num1CSint(Xp[5],Xp,int,t,c,Rv)

Xlim<-range(a,b,Xp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),xlab="",pch=".",xlim=Xlim+xd*c(-.05,.05))
abline(h=0)
abline(v=c(a,b,Mc),col=c(1,1,2),lty=2)
points(cbind(Xp,0))
points(cbind(domset,0),pch=4,col=2)
```

```
text(cbind(c(a,b,Mc),-0.1),c("a","b","Mc"))

Idom.num1CSint(Xp[5],Xp,int,t,c)

n<-10
Xp2<-runif(n,a+b,b+10)
Idom.num1CSint(5,Xp2,int,t,c)
```

`Idom.num1CSstd.tri` *The indicator for a point being a dominating point for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns $I($`p` is a dominating point of the CS-PCD$)$ where the vertices of the CS-PCD are the 2D data set `Xp` in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$, that is, returns 1 if `p` is a dominating point of CS-PCD, returns 0 otherwise.

CS proximity region is constructed with respect to T_e with expansion parameter $t > 0$ and edge regions are based on center of mass $CM = (1/2, \sqrt{3}/6)$.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num1CSstd.tri(p, Xp, t, ch.data.pnt = FALSE)
```

Arguments

`p`	A 2D point that is to be tested for being a dominating point or not of the CS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the CS-PCD.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

$I($`p` is a dominating point of the CS-PCD$)$ where the vertices of the CS-PCD are the 2D data set `Xp`, that is, returns 1 if `p` is a dominating point, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.num1CSt1std.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
Te<-rbind(A,B,C);
t<-1.5
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

Idom.num1CSstd.tri(Xp[3,],Xp,t)
Idom.num1CSstd.tri(c(1,2),c(1,2),t)
Idom.num1CSstd.tri(c(1,2),c(1,2),t,ch.data.pnt = TRUE)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1CSstd.tri(Xp[i,],Xp,t))}

ind.gam1<-which(gam.vec==1)
ind.gam1

Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(Xp)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE);
```

```
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(Xp[ind.gam1,]),pch=4,col=2)
#rbind is to insert the points correctly if there is only one dominating point

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.02,.02,.01,.05)
yc<-txt[,2]+c(.02,.02,.03,.02)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

Idom.num1CSstd.tri(c(1,2),Xp,t,ch.data.pnt = FALSE)
#gives an error if ch.data.pnt = TRUE message since p is not a data point
```

`Idom.num1CSt1std.tri` *The indicator for a point being a dominating point for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case with $t = 1$*

Description

Returns $I($p is a dominating point of the CS-PCD$)$ where the vertices of the CS-PCD are the 2D data set `Xp` in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$, that is, returns 1 if `p` is a dominating point of CS-PCD, returns 0 otherwise.

Point, `p`, is in the edge region of edge re (default is `NULL`); vertices are labeled as $1, 2, 3$ in the order they are stacked row-wise in T_e, and the opposite edges are labeled with label of the vertices (that is, edge numbering is `1,2`, and `3` for edges AB, BC, and AC).

CS proximity region is constructed with respect to T_e with expansion parameter $t = 1$ and edge regions are based on center of mass $CM = (1/2, \sqrt{3}/6)$.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num1CSt1std.tri(p, Xp, re = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 2D point that is to be tested for being a dominating point or not of the CS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the CS-PCD.
`re`	The index of the edge region in T_e containing the point, either `1,2,3` or `NULL` (default is `NULL`).
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

$I($p is a dominating point of the CS-PCD$)$ where the vertices of the CS-PCD are the 2D data set `Xp`, that is, returns 1 if `p` is a dominating point, returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.num1CSstd.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
Te<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

Idom.num1CSt1std.tri(Xp[3,],Xp)

Idom.num1CSt1std.tri(c(1,2),c(1,2))
Idom.num1CSt1std.tri(c(1,2),c(1,2),ch.data.pnt = TRUE)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1CSt1std.tri(Xp[i,],Xp))}

ind.gam1<-which(gam.vec==1)
ind.gam1

Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
plot(Te,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(Xp)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE);
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(Xp[ind.gam1,]),pch=4,col=2)
#rbind is to insert the points correctly if there is only one dominating point

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.02,.02,.01,.05)
yc<-txt[,2]+c(.02,.02,.03,.02)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)
```

`Idom.num1PEbasic.tri` *The indicator for a point being a dominating point or not for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard basic triangle case*

Description

Returns $I($`p` is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp` for data in the standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, that is, returns 1 if `p` is a dominating point of PE-PCD, and returns 0 otherwise.

PE proximity regions are defined with respect to the standard basic triangle T_b. In the standard basic triangle, T_b, c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle is useful for simulation studies under the uniformity hypothesis.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a standard basic triangle to the edges on the extension of the lines joining `M` to the vertices or based on the circumcenter of T_b; default is $M = (1,1,1)$, i.e., the center of mass of T_b. Point, `p`, is in the vertex region of vertex `rv` (default is `NULL`); vertices are labeled as $1,2,3$ in the order they are stacked row-wise.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2011)).

Usage

```
Idom.num1PEbasic.tri(
  p,
  Xp,
  r,
  c1,
  c2,
  M = c(1, 1, 1),
  rv = NULL,
  ch.data.pnt = FALSE
)
```

Arguments

`p`	A 2D point that is to be tested for being a dominating point or not of the PE-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2+c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle T_b or the circumcenter of T_b which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of T_b.
`rv`	Index of the vertex whose region contains point `p`, `rv` takes the vertex labels as $1, 2, 3$ as in the row order of the vertices in T_b.
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

$I($`p` is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp`, that is, returns 1 if `p` is a dominating point, and returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

`Idom.num1ASbasic.tri` and `Idom.num1AStri`

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.basic.tri(n,c1,c2)$g

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.3)
r<-2

P<-c(.4,.2)
Idom.num1PEbasic.tri(P,Xp,r,c1,c2,M)
Idom.num1PEbasic.tri(Xp[1,],Xp,r,c1,c2,M)

Idom.num1PEbasic.tri(c(1,1),Xp,r,c1,c2,M,ch.data.pnt = FALSE)
#gives an error message if ch.data.pnt = TRUE since point p=c(1,1) is not a data point in Xp

#or try
Rv<-rel.vert.basic.tri(Xp[1,],c1,c2,M)$rv
Idom.num1PEbasic.tri(Xp[1,],Xp,r,c1,c2,M,Rv)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1PEbasic.tri(Xp[i,],Xp,r,c1,c2,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tb)}
#need to run this when M is given in barycentric coordinates

if (identical(M,circumcenter.tri(Tb)))
{
  plot(Tb,pch=".",asp=1,xlab="",ylab="",axes=TRUE,
  xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
  polygon(Tb)
  points(Xp,pch=1,col=1)
  Ds<-rbind((B+C)/2,(A+C)/2,(A+B)/2)
} else
{plot(Tb,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
```

```
  polygon(Tb)
  points(Xp,pch=1,col=1)
  Ds<-prj.cent2edges.basic.tri(c1,c2,M)}
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(Xp[ind.gam1,]),pch=4,col=2)

txt<-rbind(Tb,M,Ds)
xc<-txt[,1]+c(-.02,.02,.02,-.02,.03,-.03,.01)
yc<-txt[,2]+c(.02,.02,.02,-.02,.02,.02,-.03)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)

Idom.num1PEbasic.tri(c(.2,.1),Xp,r,c1,c2,M,ch.data.pnt=FALSE)
#gives an error message if ch.data.pnt=TRUE since point p is not a data point in Xp
```

`Idom.num1PEint` *The indicator for a point being a dominating point for Proportional Edge Proximity Catch Digraphs (PE-PCDs) for an interval*

Description

Returns $I(\texttt{p} \text{ is a dominating point of the PE-PCD})$ where the vertices of the PE-PCD are the 1D data set `Xp`.

PE proximity region is defined with respect to the interval `int` with an expansion parameter, $r \geq 1$, and a centrality parameter, $c \in (0,1)$, so arcs may exist for `Xp` points inside the interval `int`$= (a,b)$.

Vertex regions are based on the center associated with the centrality parameter $c \in (0,1)$. `rv` is the index of the vertex region `p` resides, with default=`NULL`.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

Usage

```
Idom.num1PEint(p, Xp, int, r, c = 0.5, rv = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 1D point that is to be tested for being a dominating point or not of the PE-PCD.
`Xp`	A set of 1D points which constitutes the vertices of the PE-PCD.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$. For the interval, `int`$=(a,b)$, the parameterized center is $M_c=a+c(b-a)$; default `c=.5`.
`rv`	Index of the vertex region in which the point resides, either `1,2` or `NULL` (default is `NULL`).
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

$I($`p` is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 1D data set `Xp`, that is, returns 1 if `p` is a dominating point, returns 0 otherwise

Author(s)

Elvan Ceyhan

See Also

`Idom.num1PEtri`

Examples

```
r<-2
c<-.4
a<-0; b<-10
int=c(a,b)

Mc<-centerMc(int,c)

n<-10

set.seed(1)
Xp<-runif(n,a,b)

Idom.num1PEint(Xp[5],Xp,int,r,c)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1PEint(Xp[i],Xp,int,r,c))}

ind.gam1<-which(gam.vec==1)
ind.gam1

domset<-Xp[ind.gam1]
if (length(ind.gam1)==0)
{domset<-NA}

#or try
Rv<-rel.vert.mid.int(Xp[5],int,c)$rv
Idom.num1PEint(Xp[5],Xp,int,r,c,Rv)
```

```
Xlim<-range(a,b,Xp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),xlab="",pch=".",xlim=Xlim+xd*c(-.05,.05))
abline(h=0)
points(cbind(Xp,0))
abline(v=c(a,b,Mc),col=c(1,1,2),lty=2)
points(cbind(domset,0),pch=4,col=2)
text(cbind(c(a,b,Mc),-0.1),c("a","b","Mc"))

Idom.num1PEint(2,Xp,int,r,c,ch.data.pnt = FALSE)
#gives an error message if ch.data.pnt = TRUE since point p is not a data point in Xp
```

`Idom.num1PEstd.tetra` *The indicator for a 3D point being a dominating point for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard regular tetrahedron case*

Description

Returns $I($`p` is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp` in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3))$, that is, returns 1 if `p` is a dominating point of PE-PCD, returns 0 otherwise.

Point, `p`, is in the vertex region of vertex `rv` (default is `NULL`); vertices are labeled as `1,2,3,4` in the order they are stacked row-wise in T_h.

PE proximity region is constructed with respect to the tetrahedron T_h with expansion parameter $r \geq 1$ and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num1PEstd.tetra(p, Xp, r, rv = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 3D point that is to be tested for being a dominating point or not of the PE-PCD.
`Xp`	A set of 3D points which constitutes the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

`rv`	Index of the vertex whose region contains point `p`, `rv` takes the vertex labels as `1,2,3,4` as in the row order of the vertices in standard regular tetrahedron, default is `NULL`.
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

$I($p is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp`, that is, returns 1 if `p` is a dominating point, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`Idom.num1PEtetra`, `Idom.num1PEtri` and `Idom.num1PEbasic.tri`

Examples

```
set.seed(123)
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-5 #try also n<-20
Xp<-runif.std.tetra(n)$g  #try also Xp<-cbind(runif(n),runif(n),runif(n))
r<-1.5

P<-c(.4,.1,.2)
Idom.num1PEstd.tetra(Xp[1,],Xp,r)
Idom.num1PEstd.tetra(P,Xp,r)

Idom.num1PEstd.tetra(Xp[1,],Xp,r)
Idom.num1PEstd.tetra(Xp[1,],Xp[1,],r)

#or try
RV<-rel.vert.tetraCC(Xp[1,],tetra)$rv
Idom.num1PEstd.tetra(Xp[1,],Xp,r,rv=RV)

Idom.num1PEstd.tetra(c(-1,-1,-1),Xp,r)
Idom.num1PEstd.tetra(c(-1,-1,-1),c(-1,-1,-1),r)
```

```
gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1PEstd.tetra(Xp[i,],Xp,r))}

ind.gam1<-which(gam.vec==1)
ind.gam1
g1.pts<-Xp[ind.gam1,]

Xlim<-range(tetra[,1],Xp[,1])
Ylim<-range(tetra[,2],Xp[,2])
Zlim<-range(tetra[,3],Xp[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Xp[,1],Xp[,2],Xp[,3], phi =0,theta=40, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
         pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
if (length(g1.pts)!=0)
{
  if (length(g1.pts)==3) g1.pts<-matrix(g1.pts,nrow=1)
  plot3D::points3D(g1.pts[,1],g1.pts[,2],g1.pts[,3], pch=4,col="red", add=TRUE)}

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)

CM<-apply(tetra,2,mean)
D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-matrix(rep(CM,6),ncol=3,byrow=TRUE)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)

P<-c(.4,.1,.2)
Idom.num1PEstd.tetra(P,Xp,r)

Idom.num1PEstd.tetra(c(-1,-1,-1),Xp,r,ch.data.pnt = FALSE)
#gives an error message if ch.data.pnt = TRUE
```

`Idom.num1PEtetra` *The indicator for a 3D point being a dominating point for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns $I($`p` is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp` in the tetrahedron `th`, that is, returns 1 if `p` is a dominating point of PE-PCD, returns 0 otherwise.

Point, `p`, is in the vertex region of vertex `rv` (default is `NULL`); vertices are labeled as `1,2,3,4` in the order they are stacked row-wise in `th`.

PE proximity region is constructed with respect to the tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions are based on center of mass (`M="CM"`) or circumcenter (`M="CC"`) only. and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num1PEtetra(p, Xp, th, r, M = "CM", rv = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 3D point that is to be tested for being a dominating point or not of the PE-PCD.
`Xp`	A set of 3D points which constitutes the vertices of the PE-PCD.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.
`rv`	Index of the vertex whose region contains point `p`, `rv` takes the vertex labels as `1,2,3,4` as in the row order of the vertices in standard tetrahedron, default is `NULL`.
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

$I($`p` is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp`, that is, returns 1 if `p` is a dominating point, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`Idom.num1PEstd.tetra`, `Idom.num1PEtri` and `Idom.num1PEbasic.tri`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-5 #try also n<-20

Xp<-runif.tetra(n,tetra)$g  #try also Xp<-cbind(runif(n),runif(n),runif(n))

M<-"CM"; cent<-apply(tetra,2,mean)  #center of mass
#try also M<-"CC"; cent<-circumcenter.tetra(tetra)  #circumcenter

r<-2

P<-c(.4,.1,.2)
Idom.num1PEtetra(Xp[1,],Xp,tetra,r,M)
Idom.num1PEtetra(P,Xp,tetra,r,M)

#or try
RV<-rel.vert.tetraCC(Xp[1,],tetra)$rv
Idom.num1PEtetra(Xp[1,],Xp,tetra,r,M,rv=RV)

Idom.num1PEtetra(c(-1,-1,-1),Xp,tetra,r,M)
Idom.num1PEtetra(c(-1,-1,-1),c(-1,-1,-1),tetra,r,M)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1PEtetra(Xp[i,],Xp,tetra,r,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1
g1.pts<-Xp[ind.gam1,]

Xlim<-range(tetra[,1],Xp[,1],cent[1])
Ylim<-range(tetra[,2],Xp[,2],cent[2])
Zlim<-range(tetra[,3],Xp[,3],cent[3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Xp[,1],Xp[,2],Xp[,3], phi =0,theta=40, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
         pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
```

```
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
if (length(g1.pts)!=0)
{plot3D::points3D(g1.pts[,1],g1.pts[,2],g1.pts[,3], pch=4,col="red", add=TRUE)}

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)

D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-rbind(cent,cent,cent,cent,cent,cent)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)

P<-c(.4,.1,.2)
Idom.num1PEtetra(P,Xp,tetra,r,M)

Idom.num1PEtetra(c(-1,-1,-1),Xp,tetra,r,M,ch.data.pnt = FALSE)
#gives an error message if ch.data.pnt = TRUE since p is not a data point
```

`Idom.num1PEtri` *The indicator for a point being a dominating point for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns $I($p is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp` in the triangle `tri`, that is, returns 1 if `p` is a dominating point of PE-PCD, and returns 0 otherwise.

Point, `p`, is in the vertex region of vertex `rv` (default is `NULL`); vertices are labeled as $1, 2, 3$ in the order they are stacked row-wise in `tri`.

PE proximity region is constructed with respect to the triangle `tri` with expansion parameter $r \geq 1$ and vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.

`ch.data.pnt` is for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
Idom.num1PEtri(p, Xp, tri, r, M = c(1, 1, 1), rv = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 2D point that is to be tested for being a dominating point or not of the PE-PCD.

`Xp`	A set of 2D points which constitutes the vertices of the PE-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.
`rv`	Index of the vertex whose region contains point `p`, `rv` takes the vertex labels as $1, 2, 3$ as in the row order of the vertices in `tri`.
`ch.data.pnt`	A logical argument for checking whether point `p` is a data point in `Xp` or not (default is `FALSE`).

Value

$I($`p` is a dominating point of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp`, that is, returns 1 if `p` is a dominating point, and returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE (2007). “On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs.” *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`Idom.num1PEbasic.tri` and `Idom.num1AStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20
```

```
set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5  #try also r<-2

Idom.num1PEtri(Xp[1,],Xp,Tr,r,M)
Idom.num1PEtri(c(1,2),c(1,2),Tr,r,M)
Idom.num1PEtri(c(1,2),c(1,2),Tr,r,M,ch.data.pnt = TRUE)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Idom.num1PEtri(Xp[i,],Xp,Tr,r,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1

#or try
Rv<-rel.vert.tri(Xp[1,],Tr,M)$rv
Idom.num1PEtri(Xp[1,],Xp,Tr,r,M,Rv)

Ds<-prj.cent2edges(Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

Xlim<-range(Tr[,1],Xp[,1],M[1])
Ylim<-range(Tr[,2],Xp[,2],M[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(Xp[ind.gam1,]),pch=4,col=2)
#rbind is to insert the points correctly if there is only one dominating point

txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.03,.02,-.02,.04,-.03,.0)
yc<-txt[,2]+c(.02,.02,.05,-.03,.04,.06,-.07)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)

P<-c(1.4,1)
Idom.num1PEtri(P,P,Tr,r,M)
Idom.num1PEtri(Xp[1,],Xp,Tr,r,M)

Idom.num1PEtri(c(1,2),Xp,Tr,r,M,ch.data.pnt = FALSE)
#gives an error message if ch.data.pnt = TRUE since p is not a data point
```

`Idom.num2ASbasic.tri` *The indicator for two points being a dominating set for Arc Slice Proximity Catch Digraphs (AS-PCDs) - standard basic triangle case*

Description

Returns $I(\{$`p1,p2`$\}$ is a dominating set of AS-PCD$)$ where vertices of AS-PCD are the 2D data set `Xp`), that is, returns 1 if $\{$`p1,p2`$\}$ is a dominating set of AS-PCD, returns 0 otherwise.

AS proximity regions are defined with respect to the standard basic triangle $T_b = T(c(0,0), c(1,0), c(c1, c2))$, In the standard basic triangle, T_b, c_1 is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence standard basic triangle is useful for simulation studies under the uniformity hypothesis.

Point, `p1`, is in the vertex region of vertex `rv1` (default is `NULL`) and point, `p2`, is in the vertex region of vertex `rv2` (default is `NULL`); vertices are labeled as $1, 2, 3$ in the order they are stacked row-wise.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle T_b or based on circumcenter of T_b; default is `M="CC"`, i.e., circumcenter of T_b.

`ch.data.pnts` is for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1` and `p2` would be a dominating set if they actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num2ASbasic.tri(
  p1,
  p2,
  Xp,
  c1,
  c2,
  M = "CC",
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

`p1, p2`	Two 2D points to be tested for constituting a dominating set of the AS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the AS-PCD.
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2+c_2^2 \leq 1$.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle T_b or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of T_b.
`rv1, rv2`	The indices of the vertices whose regions contains `p1` and `p2`, respectively. They take the vertex labels as $1, 2, 3$ as in the row order of the vertices in T_b (default is `NULL` for both).
`ch.data.pnts`	A logical argument for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

$I($`{p1,p2}` is a dominating set of the AS-PCD$)$ where the vertices of AS-PCD are the 2D data set `Xp`), that is, returns 1 if `{p1,p2}` is a dominating set of AS-PCD, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.num2AStri`

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-10
```

```
set.seed(1)
Xp<-runif.basic.tri(n,c1,c2)$g

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.2)

Idom.num2ASbasic.tri(Xp[1,],Xp[2,],Xp,c1,c2,M)
Idom.num2ASbasic.tri(Xp[1,],Xp[1,],Xp,c1,c2,M)  #one point can not a dominating set of size two

Idom.num2ASbasic.tri(c(.2,.4),c(.2,.5),rbind(c(.2,.4),c(.2,.5)),c1,c2,M)

ind.gam2<-vector()
for (i in 1:(n-1))
  for (j in (i+1):n)
  {if (Idom.num2ASbasic.tri(Xp[i,],Xp[j,],Xp,c1,c2,M)==1)
   ind.gam2<-rbind(ind.gam2,c(i,j))}
ind.gam2

#or try
rv1<-rel.vert.basic.triCC(Xp[1,],c1,c2)$rv
rv2<-rel.vert.basic.triCC(Xp[2,],c1,c2)$rv
Idom.num2ASbasic.tri(Xp[1,],Xp[2,],Xp,c1,c2,M,rv1,rv2)
Idom.num2ASbasic.tri(c(.2,.4),Xp[2,],Xp,c1,c2,M,rv1,rv2)

#or try
rv1<-rel.vert.basic.triCC(Xp[1,],c1,c2)$rv
Idom.num2ASbasic.tri(Xp[1,],Xp[2,],Xp,c1,c2,M,rv1)

#or try
Rv2<-rel.vert.basic.triCC(Xp[2,],c1,c2)$rv
Idom.num2ASbasic.tri(Xp[1,],Xp[2,],Xp,c1,c2,M,rv2=Rv2)

Idom.num2ASbasic.tri(c(.3,.2),c(.35,.25),Xp,c1,c2,M)
```

`Idom.num2AStri` *The indicator for two points constituting a dominating set for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns $I(\{$`p1,p2`$\}$ is a dominating set of the AS-PCD$)$ where vertices of the AS-PCD are the 2D data set `Xp`), that is, returns 1 if {`p1,p2`} is a dominating set of AS-PCD, returns 0 otherwise.

AS proximity regions are defined with respect to the triangle `tri`. Point, `p1`, is in the region of vertex `rv1` (default is `NULL`) and point, `p2`, is in the region of vertex `rv2` (default is `NULL`); vertices (and hence `rv1` and `rv2`) are labeled as $1, 2, 3$ in the order they are stacked row-wise in `tri`.

Vertex regions are based on the center `M="CC"` for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M="CC"` the circumcenter of `tri`.

`ch.data.pnts` is for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1` and `p2` would constitute dominating set if they actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num2AStri(
  p1,
  p2,
  Xp,
  tri,
  M = "CC",
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

`p1, p2`	Two 2D points to be tested for constituting a dominating set of the AS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of `tri`.
`rv1, rv2`	The indices of the vertices whose regions contains `p1` and `p2`, respectively. They take the vertex labels as $1, 2, 3$ as in the row order of the vertices in `tri` (default is `NULL` for both).
`ch.data.pnts`	A logical argument for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

I(`{p1,p2}` is a dominating set of the AS-PCD) where vertices of the AS-PCD are the 2D data set `Xp`), that is, returns 1 if `{p1,p2}` is a dominating set of AS-PCD, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.num2ASbasic.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

Idom.num2AStri(Xp[1,],Xp[2,],Xp,Tr,M)
Idom.num2AStri(Xp[1,],Xp[1,],Xp,Tr,M)  #same two points cannot be a dominating set of size 2

Idom.num2AStri(c(.2,.4),Xp[2,],Xp,Tr,M)
Idom.num2AStri(c(.2,.4),c(.2,.5),Xp,Tr,M)
Idom.num2AStri(c(.2,.4),c(.2,.5),rbind(c(.2,.4),c(.2,.5)),Tr,M)

#or try
rv1<-rel.vert.triCC(c(.2,.4),Tr)$rv
rv2<-rel.vert.triCC(c(.2,.5),Tr)$rv
Idom.num2AStri(c(.2,.4),c(.2,.5),rbind(c(.2,.4),c(.2,.5)),Tr,M,rv1,rv2)

ind.gam2<-vector()
for (i in 1:(n-1))
  for (j in (i+1):n)
  {if (Idom.num2AStri(Xp[i,],Xp[j,],Xp,Tr,M)==1)
   ind.gam2<-rbind(ind.gam2,c(i,j))}
ind.gam2

#or try
rv1<-rel.vert.triCC(Xp[1,],Tr)$rv
rv2<-rel.vert.triCC(Xp[2,],Tr)$rv
Idom.num2AStri(Xp[1,],Xp[2,],Xp,Tr,M,rv1,rv2)

#or try
rv1<-rel.vert.triCC(Xp[1,],Tr)$rv
Idom.num2AStri(Xp[1,],Xp[2,],Xp,Tr,M,rv1)

#or try
```

```
Rv2<-rel.vert.triCC(Xp[2,],Tr)$rv
Idom.num2AStri(Xp[1,],Xp[2,],Xp,Tr,M,rv2=Rv2)

Idom.num2AStri(c(1.3,1.2),c(1.35,1.25),Xp,Tr,M)
```

`Idom.num2CS.Te.onesixth` *The indicator for two points constituting a dominating set for Central Similarity Proximity Catch Digraphs (CS-PCDs) - first one-sixth of the standard equilateral triangle case*

Description

Returns $I(\{$`p1,p2`$\}$ is a dominating set of the CS-PCD$)$ where the vertices of the CS-PCD are the 2D data set `Xp`), that is, returns 1 if `p` is a dominating point of CS-PCD, returns 0 otherwise.

CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ and with expansion parameter $t = 1$. Point, `p1`, must lie in the first one-sixth of T_e, which is the triangle with vertices $T(A, D_3, CM) = T((0,0),(1/2,0),CM)$.

`ch.data.pnts` is for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1` and `p2` would be a dominating set if they actually were in the data set.

See also (Ceyhan (2005)).

Usage

```
Idom.num2CS.Te.onesixth(p1, p2, Xp, ch.data.pnts = FALSE)
```

Arguments

`p1, p2`	Two 2D points to be tested for constituting a dominating set of the CS-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the CS-PCD.
`ch.data.pnts`	A logical argument for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

$I(\{$`p1,p2`$\}$ is a dominating set of the CS-PCD$)$ where the vertices of the CS-PCD are the 2D data set `Xp`), that is, returns 1 if `{p1,p2}` is a dominating set of CS-PCD, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

See Also

`Idom.num2CSstd.tri`

`Idom.num2PEbasic.tri` *The indicator for two points being a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard basic triangle case*

Description

Returns $I($`{p1,p2}` is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp` in the standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, that is, returns 1 if `{p1,p2}` is a dominating set of PE-PCD, and returns 0 otherwise.

PE proximity regions are defined with respect to T_b. In the standard basic triangle, T_b, c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle is useful for simulation studies under the uniformity hypothesis.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a standard basic triangle T_b; default is $M = (1,1,1)$, i.e., the center of mass of T_b. Point, `p1`, is in the vertex region of vertex `rv1` (default is `NULL`); and point, `p2`, is in the vertex region of vertex `rv2` (default is `NULL`); vertices are labeled as $1,2,3$ in the order they are stacked row-wise.

`ch.data.pnts` is for checking whether points `p1` and `p2` are both data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1` and `p2` would constitute a dominating set if they both were actually in the data set.

See also (Ceyhan (2005, 2011)).

Usage

```
Idom.num2PEbasic.tri(
  p1,
  p2,
  Xp,
  r,
  c1,
  c2,
  M = c(1, 1, 1),
```

```
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

`p1, p2`	Two 2D points to be tested for constituting a dominating set of the PE-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle. adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2+c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle T_b or the circumcenter of T_b which may be entered as "CC" as well; default is $M = (1,1,1)$, i.e., the center of mass of T_b.
`rv1, rv2`	The indices of the vertices whose regions contains `p1` and `p2`, respectively. They take the vertex labels as $1, 2, 3$ as in the row order of the vertices in T_b (default is `NULL` for both).
`ch.data.pnts`	A logical argument for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

$I($`{p1,p2}` is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp`, that is, returns 1 if `{p1,p2}` is a dominating set of PE-PCD, and returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

`Idom.num2PEtri`, `Idom.num2ASbasic.tri`, and `Idom.num2AStri`

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.basic.tri(n,c1,c2)$g

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.3)

r<-2

Idom.num2PEbasic.tri(Xp[1,],Xp[2,],Xp,r,c1,c2,M)

Idom.num2PEbasic.tri(c(1,2),c(1,3),rbind(c(1,2),c(1,3)),r,c1,c2,M)
Idom.num2PEbasic.tri(c(1,2),c(1,3),rbind(c(1,2),c(1,3)),r,c1,c2,M,
ch.data.pnts = TRUE)

ind.gam2<-vector()
for (i in 1:(n-1))
  for (j in (i+1):n)
  {if (Idom.num2PEbasic.tri(Xp[i,],Xp[j,],Xp,r,c1,c2,M)==1)
   ind.gam2<-rbind(ind.gam2,c(i,j))}
ind.gam2

#or try
rv1<-rel.vert.basic.tri(Xp[1,],c1,c2,M)$rv;
rv2<-rel.vert.basic.tri(Xp[2,],c1,c2,M)$rv;
Idom.num2PEbasic.tri(Xp[1,],Xp[2,],Xp,r,c1,c2,M,rv1,rv2)

#or try
rv1<-rel.vert.basic.tri(Xp[1,],c1,c2,M)$rv;
Idom.num2PEbasic.tri(Xp[1,],Xp[2,],Xp,r,c1,c2,M,rv1)

#or try
rv2<-rel.vert.basic.tri(Xp[2,],c1,c2,M)$rv;
Idom.num2PEbasic.tri(Xp[1,],Xp[2,],Xp,r,c1,c2,M,rv2=rv2)

Idom.num2PEbasic.tri(c(1,2),Xp[2,],Xp,r,c1,c2,M,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE since not both points are data points in Xp
```

`Idom.num2PEstd.tetra` *The indicator for two 3D points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard regular tetrahedron case*

Description

Returns $I($`{p1,p2}` is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp` in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3))$, that is, returns 1 if `{p1,p2}` is a dominating set of PE-PCD, returns 0 otherwise.

Point, `p1`, is in the region of vertex `rv1` (default is `NULL`) and point, `p2`, is in the region of vertex `rv2` (default is `NULL`); vertices (and hence `rv1` and `rv2`) are labeled as `1,2,3,4` in the order they are stacked row-wise in T_h.

PE proximity region is constructed with respect to the tetrahedron T_h with expansion parameter $r \geq 1$ and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

`ch.data.pnts` is for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1` and `p2` would constitute a dominating set if they actually were both in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num2PEstd.tetra(
  p1,
  p2,
  Xp,
  r,
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

`p1, p2`	Two 3D points to be tested for constituting a dominating set of the PE-PCD.
`Xp`	A set of 3D points which constitutes the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`rv1, rv2`	The indices of the vertices whose regions contains `p1` and `p2`, respectively. They take the vertex labels as `1,2,3,4` as in the row order of the vertices in T_h (default is `NULL` for both).
`ch.data.pnts`	A logical argument for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

$I($`{p1,p2}` is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp`), that is, returns 1 if `{p1,p2}` is a dominating set of PE-PCD, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`Idom.num2PEtetra`, `Idom.num2PEtri` and `Idom.num2PEbasic.tri`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-5 #try also n<-20
Xp<-runif.std.tetra(n)$g  #try also Xp<-cbind(runif(n),runif(n),runif(n))
r<-1.5

Idom.num2PEstd.tetra(Xp[1,],Xp[2,],Xp,r)

ind.gam2<-vector()
for (i in 1:(n-1))
 for (j in (i+1):n)
 {if (Idom.num2PEstd.tetra(Xp[i,],Xp[j,],Xp,r)==1)
  ind.gam2<-rbind(ind.gam2,c(i,j))}

ind.gam2

#or try
rv1<-rel.vert.tetraCC(Xp[1,],tetra)$rv;rv2<-rel.vert.tetraCC(Xp[2,],tetra)$rv
Idom.num2PEstd.tetra(Xp[1,],Xp[2,],Xp,r,rv1,rv2)

#or try
rv1<-rel.vert.tetraCC(Xp[1,],tetra)$rv;
Idom.num2PEstd.tetra(Xp[1,],Xp[2,],Xp,r,rv1)

#or try
rv2<-rel.vert.tetraCC(Xp[2,],tetra)$rv
Idom.num2PEstd.tetra(Xp[1,],Xp[2,],Xp,r,rv2=rv2)

P1<-c(.1,.1,.1)
P2<-c(.4,.1,.2)
Idom.num2PEstd.tetra(P1,P2,Xp,r)

Idom.num2PEstd.tetra(c(-1,-1,-1),Xp[2,],Xp,r,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE
#since not both points, p1 and p2, are data points in Xp
```

`Idom.num2PEtetra` *The indicator for two 3D points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns $I(${`p1,p2`} is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp` in the tetrahedron `th`, that is, returns 1 if {`p1,p2`} is a dominating set of PE-PCD, returns 0 otherwise.

Point, `p1`, is in the region of vertex `rv1` (default is `NULL`) and point, `p2`, is in the region of vertex `rv2` (default is `NULL`); vertices (and hence `rv1` and `rv2`) are labeled as `1,2,3,4` in the order they are stacked row-wise in `th`.

PE proximity region is constructed with respect to the tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions are based on center of mass (`M="CM"`) or circumcenter (`M="CC"`) only.

`ch.data.pnts` is for checking whether points `p1` and `p2` are both data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1` and `p2` would constitute a dominating set if they actually were both in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num2PEtetra(
  p1,
  p2,
  Xp,
  th,
  r,
  M = "CM",
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

`p1, p2`	Two 3D points to be tested for constituting a dominating set of the PE-PCD.
`Xp`	A set of 3D points which constitutes the vertices of the PE-PCD.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.

`rv1, rv2`	The indices of the vertices whose regions contains `p1` and `p2`, respectively. They take the vertex labels as `1,2,3,4` as in the row order of the vertices in `th` (default is `NULL` for both).
`ch.data.pnts`	A logical argument for checking whether both points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

$I(\{$`p1,p2`$\}$ is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp`), that is, returns 1 if $\{$`p1,p2`$\}$ is a dominating set of PE-PCD, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`Idom.num2PEstd.tetra`, `Idom.num2PEtri` and `Idom.num2PEbasic.tri`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-5

set.seed(1)
Xp<-runif.tetra(n,tetra)$g  #try also Xp<-cbind(runif(n),runif(n),runif(n))

M<-"CM";  #try also M<-"CC";
r<-1.5

Idom.num2PEtetra(Xp[1,],Xp[2,],Xp,tetra,r,M)
Idom.num2PEtetra(c(-1,-1,-1),Xp[2,],Xp,tetra,r,M)

ind.gam2<-ind.gamn2<-vector()
for (i in 1:(n-1))
 for (j in (i+1):n)
 {if (Idom.num2PEtetra(Xp[i,],Xp[j,],Xp,tetra,r,M)==1)
 {ind.gam2<-rbind(ind.gam2,c(i,j))
 }
 }
ind.gam2
```

```
#or try
rv1<-rel.vert.tetraCC(Xp[1,],tetra)$rv;rv2<-rel.vert.tetraCC(Xp[2,],tetra)$rv
Idom.num2PEtetra(Xp[1,],Xp[2,],Xp,tetra,r,M,rv1,rv2)

#or try
rv1<-rel.vert.tetraCC(Xp[1,],tetra)$rv;
Idom.num2PEtetra(Xp[1,],Xp[2,],Xp,tetra,r,M,rv1)

#or try
rv2<-rel.vert.tetraCC(Xp[2,],tetra)$rv
Idom.num2PEtetra(Xp[1,],Xp[2,],Xp,tetra,r,M,rv2=rv2)

P1<-c(.1,.1,.1)
P2<-c(.4,.1,.2)
Idom.num2PEtetra(P1,P2,Xp,tetra,r,M)

Idom.num2PEtetra(c(-1,-1,-1),Xp[2,],Xp,tetra,r,M,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE
#since not both points, p1 and p2, are data points in Xp
```

`Idom.num2PEtri` *The indicator for two points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns $I(\{$`p1,p2`$\}$ is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp`, that is, returns 1 if $\{$`p1,p2`$\}$ is a dominating set of PE-PCD, and returns 0 otherwise.

Point, `p1`, is in the region of vertex `rv1` (default is `NULL`) and point, `p2`, is in the region of vertex `rv2` (default is `NULL`); vertices (and hence `rv1` and `rv2`) are labeled as $1, 2, 3$ in the order they are stacked row-wise in `tri`.

PE proximity regions are defined with respect to the triangle `tri` and vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.

`ch.data.pnts` is for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1` and `p2` would be a dominating set if they actually were in the data set.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
Idom.num2PEtri(
  p1,
```

```
  p2,
  Xp,
  tri,
  r,
  M = c(1, 1, 1),
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

`p1, p2`	Two 2D points to be tested for constituting a dominating set of the PE-PCD.
`Xp`	A set of 2D points which constitutes the vertices of the PE-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.
`rv1, rv2`	The indices of the vertices whose regions contains `p1` and `p2`, respectively. They take the vertex labels as $1, 2, 3$ as in the row order of the vertices in `tri` (default is `NULL` for both).
`ch.data.pnts`	A logical argument for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

$I($`{p1,p2}` is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 2D data set `Xp`, that is, returns 1 if `{p1,p2}` is a dominating set of PE-PCD, and returns 0 otherwise.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`Idom.num2PEbasic.tri`, `Idom.num2AStri`, and `Idom.num2PEtetra`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5  #try also r<-2

Idom.num2PEtri(Xp[1,],Xp[2,],Xp,Tr,r,M)

ind.gam2<-vector()
for (i in 1:(n-1))
  for (j in (i+1):n)
  {if (Idom.num2PEtri(Xp[i,],Xp[j,],Xp,Tr,r,M)==1)
   ind.gam2<-rbind(ind.gam2,c(i,j))}
ind.gam2

#or try
rv1<-rel.vert.tri(Xp[1,],Tr,M)$rv;
rv2<-rel.vert.tri(Xp[2,],Tr,M)$rv
Idom.num2PEtri(Xp[1,],Xp[2,],Xp,Tr,r,M,rv1,rv2)

Idom.num2PEtri(Xp[1,],c(1,2),Xp,Tr,r,M,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE
#since not both points, p1 and p2, are data points in Xp
```

`Idom.num3PEstd.tetra` *The indicator for three 3D points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard regular tetrahedron case*

Description

Returns $I(\{$`p1,p2,pt3`$\}$ is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp` in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3))$, that is, returns 1 if $\{$`p1,p2,pt3`$\}$ is a dominating set of PE-PCD, returns 0 otherwise.

Point, `p1`, is in the region of vertex `rv1` (default is `NULL`), point, `p2`, is in the region of vertex `rv2` (default is `NULL`); point, `pt3`), is in the region of vertex `rv3`) (default is `NULL`); vertices (and hence `rv1, rv2` and `rv3`) are labeled as `1,2,3,4` in the order they are stacked row-wise in T_h.

PE proximity region is constructed with respect to the tetrahedron T_h with expansion parameter $r \geq 1$ and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

`ch.data.pnts` is for checking whether points `p1`, `p2` and `pt3` are all data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1`, `p2` and `pt3` would constitute a dominating set if they actually were all in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num3PEstd.tetra(
  p1,
  p2,
  pt3,
  Xp,
  r,
  rv1 = NULL,
  rv2 = NULL,
  rv3 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

`p1, p2, pt3`	Three 3D points to be tested for constituting a dominating set of the PE-PCD.
`Xp`	A set of 3D points which constitutes the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`rv1, rv2, rv3`	The indices of the vertices whose regions contains `p1`, `p2` and `pt3`, respectively. They take the vertex labels as `1,2,3,4` as in the row order of the vertices in T_h (default is `NULL` for all).
`ch.data.pnts`	A logical argument for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

$I(\{$`p1,p2,pt3`$\}$ is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp`), that is, returns 1 if $\{$`p1,p2,pt3`$\}$ is a dominating set of PE-PCD, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`Idom.num3PEtetra`

Examples

```
set.seed(123)
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-5 #try 20, 40, 100 (larger n may take a long time)
Xp<-runif.std.tetra(n)$g  #try also Xp<-cbind(runif(n),runif(n),runif(n))
r<-1.25

Idom.num3PEstd.tetra(Xp[1,],Xp[2,],Xp[3,],Xp,r)

ind.gam3<-vector()
for (i in 1:(n-2))
 for (j in (i+1):(n-1))
   for (k in (j+1):n)
 {if (Idom.num3PEstd.tetra(Xp[i,],Xp[j,],Xp[k,],Xp,r)==1)
  ind.gam3<-rbind(ind.gam3,c(i,j,k))}

ind.gam3

#or try
rv1<-rel.vert.tetraCC(Xp[1,],tetra)$rv; rv2<-rel.vert.tetraCC(Xp[2,],tetra)$rv;
rv3<-rel.vert.tetraCC(Xp[3,],tetra)$rv
Idom.num3PEstd.tetra(Xp[1,],Xp[2,],Xp[3,],Xp,r,rv1,rv2,rv3)

#or try
rv1<-rel.vert.tetraCC(Xp[1,],tetra)$rv;
Idom.num3PEstd.tetra(Xp[1,],Xp[2,],Xp[3,],Xp,r,rv1)

#or try
rv2<-rel.vert.tetraCC(Xp[2,],tetra)$rv
Idom.num3PEstd.tetra(Xp[1,],Xp[2,],Xp[3,],Xp,r,rv2=rv2)

P1<-c(.1,.1,.1)
P2<-c(.3,.3,.3)
```

```
P3<-c(.4,.1,.2)
Idom.num3PEstd.tetra(P1,P2,P3,Xp,r)

Idom.num3PEstd.tetra(Xp[1,],c(1,1,1),Xp[3,],Xp,r,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE since not all points are data points in Xp
```

`Idom.num3PEtetra` *The indicator for three 3D points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns $I(\{$`p1,p2,pt3`$\}$ is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp` in the tetrahedron `th`, that is, returns 1 if $\{$`p1,p2,pt3`$\}$ is a dominating set of PE-PCD, returns 0 otherwise.

Point, `p1`, is in the region of vertex `rv1` (default is `NULL`), point, `p2`, is in the region of vertex `rv2` (default is `NULL`); point, `pt3`), is in the region of vertex `rv3`) (default is `NULL`); vertices (and hence `rv1`, `rv2` and `rv3`) are labeled as `1,2,3,4` in the order they are stacked row-wise in `th`.

PE proximity region is constructed with respect to the tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

`ch.data.pnts` is for checking whether points `p1`, `p2` and `pt3` are all data points in `Xp` or not (default is `FALSE`), so by default this function checks whether the points `p1`, `p2` and `pt3` would constitute a dominating set if they actually were all in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.num3PEtetra(
  p1,
  p2,
  pt3,
  Xp,
  th,
  r,
  M = "CM",
  rv1 = NULL,
  rv2 = NULL,
  rv3 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

`p1`, `p2`, `pt3`	Three 3D points to be tested for constituting a dominating set of the PE-PCD.
`Xp`	A set of 3D points which constitutes the vertices of the PE-PCD.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.
`rv1`, `rv2`, `rv3`	The indices of the vertices whose regions contains `p1`, `p2` and `pt3`, respectively. They take the vertex labels as `1,2,3,4` as in the row order of the vertices in `th` (default is `NULL` for all).
`ch.data.pnts`	A logical argument for checking whether points `p1` and `p2` are data points in `Xp` or not (default is `FALSE`).

Value

$I(\{$`p1,p2,pt3`$\}$ is a dominating set of the PE-PCD$)$ where the vertices of the PE-PCD are the 3D data set `Xp`), that is, returns 1 if $\{$`p1,p2,pt3`$\}$ is a dominating set of PE-PCD, returns 0 otherwise

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`Idom.num3PEstd.tetra`

Examples

```
set.seed(123)
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-5 #try 20, 40, 100 (larger n may take a long time)
Xp<-runif.tetra(n,tetra)$g

M<-"CM";  #try also M<-"CC";
r<-1.25
```

```
Idom.num3PEtetra(Xp[1,],Xp[2,],Xp[3,],Xp,tetra,r,M)

ind.gam3<-vector()
for (i in 1:(n-2))
 for (j in (i+1):(n-1))
   for (k in (j+1):n)
   {if (Idom.num3PEtetra(Xp[i,],Xp[j,],Xp[k,],Xp,tetra,r,M)==1)
    ind.gam3<-rbind(ind.gam3,c(i,j,k))}

ind.gam3

#or try
rv1<-rel.vert.tetraCC(Xp[1,],tetra)$rv; rv2<-rel.vert.tetraCC(Xp[2,],tetra)$rv;
rv3<-rel.vert.tetraCC(Xp[3,],tetra)$rv
Idom.num3PEtetra(Xp[1,],Xp[2,],Xp[3,],Xp,tetra,r,M,rv1,rv2,rv3)

#or try
rv1<-rel.vert.tetraCC(Xp[1,],tetra)$rv;
Idom.num3PEtetra(Xp[1,],Xp[2,],Xp[3,],Xp,tetra,r,M,rv1)

#or try
rv2<-rel.vert.tetraCC(Xp[2,],tetra)$rv
Idom.num3PEtetra(Xp[1,],Xp[2,],Xp[3,],Xp,tetra,r,M,rv2=rv2)

P1<-c(.1,.1,.1)
P2<-c(.3,.3,.3)
P3<-c(.4,.1,.2)
Idom.num3PEtetra(P1,P2,P3,Xp,tetra,r,M)

Idom.num3PEtetra(Xp[1,],c(1,1,1),Xp[3,],Xp,tetra,r,M,ch.data.pnts = FALSE)
#gives an error message if ch.data.pnts = TRUE since not all points are data points in Xp
```

`Idom.numASup.bnd.tri` *Indicator for an upper bound for the domination number of Arc Slice Proximity Catch Digraph (AS-PCD) by the exact algorithm - one triangle case*

Description

Returns I(domination number of AS-PCD whose vertices are the data points `Xp` is less than or equal to `k`), that is, returns 1 if the domination number of AS-PCD is less than the prespecified value `k`, returns 0 otherwise. It also provides the vertices (i.e., data points) in a dominating set of size `k` of AS-PCD.

AS proximity regions are constructed with respect to the triangle `tri` and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`.

The vertices of `triangle`, `tri`, are labeled as $1, 2, 3$ according to the row number the vertex is recorded in `tri`. Loops are allowed in the digraph. It takes a long time for large number of vertices (i.e., large number of row numbers).

Usage

```
Idom.numASup.bnd.tri(Xp, k, tri, M = "CC")
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`k`	A positive integer to be tested for an upper bound for the domination number of AS-PCDs.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M="CC"` i.e., the circumcenter of `tri`.

Value

A `list` with the elements

`domUB`	The suggested upper bound (to be checked) for the domination number of AS-PCD. It is prespecified as `k` in the function arguments.
`Idom.num.up.bnd`	The indicator for the upper bound for domination number of AS-PCD being the specified value `k` or not. It returns 1 if the upper bound is `k`, and 0 otherwise.
`ind.dom.set`	The vertices (i.e., data points) in the dominating set of size `k` if it exists, otherwise it yields `NULL`.

Author(s)

Elvan Ceyhan

See Also

`Idom.numCSup.bnd.tri`, `Idom.numCSup.bnd.std.tri`, `Idom.num.up.bnd`, and `dom.num.exact`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$gen.points
```

```
M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

Idom.numASup.bnd.tri(Xp,1,Tr)

for (k in 1:n)
  print(c(k,Idom.numASup.bnd.tri(Xp,k,Tr,M)))

Idom.numASup.bnd.tri(Xp,k=4,Tr,M)

P<-c(.4,.2)
Idom.numASup.bnd.tri(P,1,Tr,M)

Idom.numASup.bnd.tri(rbind(Xp,Xp),k=2,Tr,M)
```

`Idom.numCSup.bnd.std.tri`

The indicator for `k` *being an upper bound for the domination number of Central Similarity Proximity Catch Digraph (CS-PCD) by the exact algorithm - standard equilateral triangle case*

Description

Returns $I(\text{domination number of CS-PCD is less than or equal to } \texttt{k})$ where the vertices of the CS-PCD are the data points `Xp`, that is, returns 1 if the domination number of CS-PCD is less than the prespecified value `k`, returns 0 otherwise. It also provides the vertices (i.e., data points) in a dominating set of size `k` of CS-PCD.

CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e., the center of mass of T_e (which is equivalent to the circumcenter of T_e).

Edges of T_e, AB, BC, AC, are also labeled as 3, 1, and 2, respectively. Loops are allowed in the digraph. It takes a long time for large number of vertices (i.e., large number of row numbers).

See also (Ceyhan (2012)).

Usage

```
Idom.numCSup.bnd.std.tri(Xp, k, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of CS-PCD.
`k`	A positive integer representing an upper bound for the domination number of CS-PCD.

`t`	A positive real number which serves as the expansion parameter in CS proximity region in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e.

Value

A `list` with two elements

`domUB`	The upper bound `k` (to be checked) for the domination number of CS-PCD. It is prespecified as `k` in the function arguments.
`Idom.num.up.bnd`	The indicator for the upper bound for domination number of CS-PCD being the specified value `k` or not. It returns 1 if the upper bound is `k`, and 0 otherwise.
`ind.domset`	The vertices (i.e., data points) in the dominating set of size `k` if it exists, otherwise it is `NULL`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.numCSup.bnd.tri`, `Idom.num.up.bnd`, `Idom.numASup.bnd.tri`, and `dom.num.exact`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

t<-.5

Idom.numCSup.bnd.std.tri(Xp,1,t,M)

for (k in 1:n)
  print(c(k,Idom.numCSup.bnd.std.tri(Xp,k,t,M)$Idom.num.up.bnd))
  print(c(k,Idom.numCSup.bnd.std.tri(Xp,k,t,M)$domUB))
```

`Idom.numCSup.bnd.tri` *Indicator for an upper bound for the domination number of Central Similarity Proximity Catch Digraph (CS-PCD) by the exact algorithm - one triangle case*

Description

Returns $I($domination number of CS-PCD is less than or equal to `k`$)$ where the vertices of the CS-PCD are the data points `Xp`, that is, returns 1 if the domination number of CS-PCD is less than the prespecified value `k`, returns 0 otherwise. It also provides the vertices (i.e., data points) in a dominating set of size `k` of CS-PCD.

CS proximity region is constructed with respect to the triangle `tri`$= T(A,B,C)$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1,m_2)$ in Cartesian coordinates or $M = (\alpha,\beta,\gamma)$ in barycentric coordinates in the interior of `tri`; default is $M = (1,1,1)$ i.e., the center of mass of `tri`.

Edges of `tri`, AB, BC, AC, are also labeled as 3, 1, and 2, respectively. Loops are allowed in the digraph.

See also (Ceyhan (2012)).

Caveat: It takes a long time for large number of vertices (i.e., large number of row numbers).

Usage

```
Idom.numCSup.bnd.tri(Xp, k, tri, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of CS-PCD.
`k`	A positive integer to be tested for an upper bound for the domination number of CS-PCDs.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region in the triangle `tri`.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1,1,1)$, i.e. the center of mass of `tri`.

Value

A `list` with two elements

`domUB`	The upper bound `k` (to be checked) for the domination number of CS-PCD. It is prespecified as `k` in the function arguments.

`Idom.num.up.bnd`	The indicator for the upper bound for domination number of CS-PCD being the specified value `k` or not. It returns 1 if the upper bound is `k`, and 0 otherwise.
`ind.domset`	The vertices (i.e., data points) in the dominating set of size `k` if it exists, otherwise it is `NULL`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.numCSup.bnd.std.tri`, `Idom.num.up.bnd`, `Idom.numASup.bnd.tri`, and `dom.num.exact`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

t<-.5

Idom.numCSup.bnd.tri(Xp,1,Tr,t,M)

for (k in 1:n)
  print(c(k,Idom.numCSup.bnd.tri(Xp,k,Tr,t,M)))
```

`Idom.setAStri`	*The indicator for the set of points* `S` *being a dominating set or not for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns $I($`S` a dominating set of AS-PCD$)$, that is, returns 1 if `S` is a dominating set of AS-PCD, returns 0 otherwise.

AS-PCD has vertex set `Xp` and AS proximity region is constructed with vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri` whose vertices are also labeled as edges 1, 2, and 3, respectively.

See also (Ceyhan (2005, 2010)).

Usage

```
Idom.setAStri(S, Xp, tri, M = "CC")
```

Arguments

`S`	A set of 2D points which is to be tested for being a dominating set for the AS-PCDs.
`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M="CC"` i.e., the circumcenter of `tri`.

Value

$I($`S` a dominating set of AS-PCD$)$, that is, returns 1 if `S` is a dominating set of AS-PCD whose vertices are the data points in `Xp`; returns 0 otherwise, where AS proximity region is constructed in the triangle `tri`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IarcASset2pnt.tri, Idom.setPEtri and Idom.setCStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

S<-rbind(Xp[1,],Xp[2,])
Idom.setAStri(S,Xp,Tr,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,])
Idom.setAStri(S,Xp,Tr,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
Idom.setAStri(S,Xp,Tr,M)

Idom.setAStri(c(.2,.5),Xp,Tr,M)
Idom.setAStri(c(.2,.5),c(.2,.5),Tr,M)
Idom.setAStri(Xp[5,],Xp[2,],Tr,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,],c(.2,.5))
Idom.setAStri(S,Xp[3,],Tr,M)

Idom.setAStri(Xp,Xp,Tr,M)

P<-c(.4,.2)
S<-Xp[c(1,3,4),]
Idom.setAStri(Xp,P,Tr,M)
Idom.setAStri(S,P,Tr,M)
Idom.setAStri(S,Xp,Tr,M)

Idom.setAStri(rbind(S,S),Xp,Tr,M)
```

`Idom.setCSstd.tri` *The indicator for the set of points* `S` *being a dominating set or not for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns $I($S a dominating set of the CS-PCD$)$ where the vertices of the CS-PCD are the data set Xp), that is, returns 1 if S is a dominating set of CS-PCD, returns 0 otherwise.

CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e., the center of mass of T_e (which is equivalent to the circumcenter of T_e).

Edges of T_e, AB, BC, AC, are also labeled as 3, 1, and 2, respectively.

See also (Ceyhan (2012)).

Usage

```
Idom.setCSstd.tri(S, Xp, t, M = c(1, 1, 1))
```

Arguments

S	A set of 2D points which is to be tested for being a dominating set for the CS-PCDs.
Xp	A set of 2D points which constitute the vertices of the CS-PCD.
t	A positive real number which serves as the expansion parameter in CS proximity region in the standard equilateral triangle $T_e = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.

Value

$I($S a dominating set of the CS-PCD$)$, that is, returns 1 if S is a dominating set of CS-PCD, returns 0 otherwise, where CS proximity region is constructed in the standard equilateral triangle T_e

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

Idom.setCStri and Idom.setPEstd.tri

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

t<-.5

S<-rbind(Xp[1,],Xp[2,])
Idom.setCSstd.tri(S,Xp,t,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,])
Idom.setCSstd.tri(S,Xp,t,M)
```

`Idom.setCStri` *The indicator for the set of points `S` being a dominating set or not for Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns $I($`S` a dominating set of CS-PCD whose vertices are the data set `Xp`$)$, that is, returns 1 if `S` is a dominating set of CS-PCD, returns 0 otherwise.

CS proximity region is constructed with respect to the triangle `tri` with the expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

The triangle `tri`$= T(A, B, C)$ has edges AB, BC, AC which are also labeled as edges 3, 1, and 2, respectively.

See also (Ceyhan (2012)).

Usage

```
Idom.setCStri(S, Xp, tri, t, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points which is to be tested for being a dominating set for the CS-PCDs.
`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

`t`	A positive real number which serves as the expansion parameter in CS proximity region constructed in the triangle `tri`.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

Value

$I($`S` a dominating set of the CS-PCD$)$, that is, returns 1 if `S` is a dominating set of CS-PCD whose vertices are the data points in `Xp`; returns 0 otherwise, where CS proximity region is constructed in the triangle `tri`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`Idom.setCSstd.tri`, `Idom.setPEtri` and `Idom.setAStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

tau<-.5
S<-rbind(Xp[1,],Xp[2,])
Idom.setCStri(S,Xp,Tr,tau,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,])
Idom.setCStri(S,Xp,Tr,tau,M)
```

`Idom.setPEstd.tri` *The indicator for the set of points* `S` *being a dominating set or not for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case*

Description

Returns $I($`S` a dominating set of PE-PCD whose vertices are the data points `Xp`$)$ for `S` in the standard equilateral triangle, that is, returns 1 if `S` is a dominating set of PE-PCD, and returns 0 otherwise.

PE proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$, i.e., the center of mass of T_e (which is also equivalent to the circumcenter of T_e). Vertices of T_e are also labeled as 1, 2, and 3, respectively.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
Idom.setPEstd.tri(S, Xp, r, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points whose PE proximity regions are considered.
`Xp`	A set of 2D points which constitutes the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region in the standard equilateral triangle $T_e = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.

Value

$I($`S` a dominating set of PE-PCD$)$ for `S` in the standard equilateral triangle, that is, returns 1 if `S` is a dominating set of PE-PCD, and returns 0 otherwise, where PE proximity region is constructed in the standard equilateral triangle T_e.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`Idom.setPEtri` and `Idom.setCSstd.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

r<-1.5

S<-rbind(Xp[1,],Xp[2,])
Idom.setPEstd.tri(S,Xp,r,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,],c(.2,.5))
Idom.setPEstd.tri(S,Xp[3,],r,M)
```

`Idom.setPEtri` *The indicator for the set of points* `S` *being a dominating set or not for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns $I($`S` a dominating set of PE-PCD whose vertices are the data set `Xp`$)$, that is, returns 1 if `S` is a dominating set of PE-PCD, and returns 0 otherwise.

PE proximity region is constructed with respect to the triangle `tri` with the expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`. The triangle `tri`$= T(A, B, C)$ has edges AB, BC, AC which are also labeled as edges 3, 1, and 2, respectively.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
Idom.setPEtri(S, Xp, tri, r, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points which is to be tested for being a dominating set for the PE-PCDs.
`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region constructed in the triangle `tri`; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.

Value

$I($`S` a dominating set of PE-PCD$)$, that is, returns 1 if `S` is a dominating set of PE-PCD whose vertices are the data points in `Xp`; and returns 0 otherwise, where PE proximity region is constructed in the triangle `tri`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`Idom.setPEstd.tri`, `IarcPEset2pnt.tri`, `Idom.setCStri`, and `Idom.setAStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5

S<-rbind(Xp[1,],Xp[2,])
Idom.setPEtri(S,Xp,Tr,r,M)

S<-rbind(Xp[1,],Xp[2,],Xp[3,],Xp[5,])
Idom.setPEtri(S,Xp,Tr,r,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
Idom.setPEtri(S,Xp,Tr,r,M)
```

`in.circle` *Check whether a point is inside a circle*

Description

Checks if the point `p` lies in the circle with center `cent` and radius `rad`, denoted as `C(cent,rad)`. So, it returns 1 or `TRUE` if `p` is inside the circle, and 0 otherwise.

`boundary` is a logical argument (default=`FALSE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `p`, lies in the closure of the circle (i.e., interior and boundary combined) else it checks if `p` lies in the interior of the circle.

Usage

```
in.circle(p, cent, rad, boundary = TRUE)
```

Arguments

`p`	A 2D point to be checked whether it is inside the circle or not.
`cent`	A 2D point in Cartesian coordinates which serves as the center of the circle.
`rad`	A positive real number which serves as the radius of the circle.
`boundary`	A logical parameter (default=`TRUE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `p`, lies in the closure of the circle (i.e., interior and boundary combined); else, it checks if `p` lies in the interior of the circle.

Value

Indicator for the point `p` being inside the circle or not, i.e., returns 1 or `TRUE` if `p` is inside the circle, and 0 otherwise.

Author(s)

Elvan Ceyhan

See Also

`in.triangle`, `in.tetrahedron`, and `on.convex.hull` from the `interp` package for documentation for `in.convex.hull`

Examples

```
cent<-c(1,1); rad<-1; p<-c(1.4,1.2)
#try also cent<-runif(2); rad<-runif(1); p<-runif(2);

in.circle(p,cent,rad)

p<-c(.4,-.2)
in.circle(p,cent,rad)

p<-c(1,0)
in.circle(p,cent,rad)
in.circle(p,cent,rad,boundary=FALSE)
```

`in.tetrahedron` *Check whether a point is inside a tetrahedron*

Description

Checks if the point `p` lies in the tetrahedron, `th`, using the barycentric coordinates, generally denoted as (α, β, γ). If all (normalized or non-normalized) barycentric coordinates are positive then the point `p` is inside the tetrahedron, if all are nonnegative with one or more are zero, then `p` falls on the boundary. If some of the barycentric coordinates are negative, then `p` falls outside the tetrahedron.

`boundary` is a logical argument (default=`FALSE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `p`, lies in the closure of the tetrahedron (i.e., interior and boundary combined) else it checks if `p` lies in the interior of the tetrahedron.

Usage

```
in.tetrahedron(p, th, boundary = TRUE)
```

Arguments

`p`	A 3D point to be checked whether it is inside the tetrahedron or not.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`boundary`	A logical parameter (default=`TRUE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `p`, lies in the closure of the tetrahedron (i.e., interior and boundary combined); else, it checks if `p` lies in the interior of the tetrahedron.

Value

A `list` with two elements

`in.tetra`	A logical output, if the point, `p`, is inside the tetrahedron, `th`, it is `TRUE`, else it is `FALSE`.
`barycentric`	The barycentric coordinates of the point `p` with respect to the tetrahedron, `th`.

Author(s)

Elvan Ceyhan

See Also

`in.triangle`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0);
D<-c(1/2,sqrt(3)/6,sqrt(6)/3); P<-c(.1,.1,.1)
tetra<-rbind(A,B,C,D)

in.tetrahedron(P,tetra,boundary = FALSE)

in.tetrahedron(C,tetra)
in.tetrahedron(C,tetra,boundary = FALSE)
```

```
n1<-5; n2<-5; n<-n1+n2
Xp<-rbind(cbind(runif(n1),runif(n1,0,sqrt(3)/2),runif(n1,0,sqrt(6)/3)),
          runif.tetra(n2,tetra)$g)

in.tetra<-vector()
for (i in 1:n)
{in.tetra<-c(in.tetra,in.tetrahedron(Xp[i,],tetra,boundary = TRUE)$in.tetra) }

in.tetra
dat.tet<-Xp[in.tetra,]
if (is.vector(dat.tet)) {dat.tet<-matrix(dat.tet,nrow=1)}

Xlim<-range(tetra[,1],Xp[,1])
Ylim<-range(tetra[,2],Xp[,2])
Zlim<-range(tetra[,3],Xp[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Xp[,1],Xp[,2],Xp[,3], phi=40,theta=40,
bty = "g", pch = 20, cex = 1,
ticktype="detailed",xlim=Xlim+xd*c(-.05,.05),
ylim=Ylim+yd*c(-.05,.05),zlim=Zlim+zd*c(-.05,.05))
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
plot3D::points3D(dat.tet[,1],dat.tet[,2],dat.tet[,3],pch=4, add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3],
labels=c("A","B","C","D"), add=TRUE)

in.tetrahedron(P,tetra) #this works fine
```

`in.tri.all` *Check whether all points in a data set are inside the triangle*

Description

Checks if all the data points in the 2D data set, `Xp`, lie in the triangle, `tri`, using the barycentric coordinates, generally denoted as (α, β, γ).

If all (normalized or non-normalized) barycentric coordinates of a point are positive then the point is inside the triangle, if all are nonnegative with one or more are zero, then the point falls in the boundary. If some of the barycentric coordinates are negative, then the point falls outside the triangle.

`boundary` is a logical argument (default=`TRUE`) to include boundary or not, so if it is `TRUE`, the function checks if a point lies in the closure of the triangle (i.e., interior and boundary combined); else, it checks if the point lies in the interior of the triangle.

Usage

```
in.tri.all(Xp, tri, boundary = TRUE)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`boundary`	A logical parameter (default=`FALSE`) to include boundary or not, so if it is `TRUE`, the function checks if a point lies in the closure of the triangle (i.e., interior and boundary combined) else it checks if the point lies in the interior of the triangle.

Value

A logical output, if all data points in `Xp` are inside the triangle, `tri`, the output is `TRUE`, else it is `FALSE`.

Author(s)

Elvan Ceyhan

See Also

`in.triangle` and `on.convex.hull` from the `interp` package for documentation for `in.convex.hull`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2); p<-c(1.4,1.2)

Tr<-rbind(A,B,C)

in.tri.all(p,Tr)

#for the vertex A
in.tri.all(A,Tr)
in.tri.all(A,Tr,boundary = FALSE)

#for a point on the edge AB
D3<-(A+B)/2
in.tri.all(D3,Tr)
in.tri.all(D3,Tr,boundary = FALSE)

#data set
n<-10
Xp<-cbind(runif(n),runif(n))
in.tri.all(Xp,Tr,boundary = TRUE)

Xp<-runif.std.tri(n)$gen.points
in.tri.all(Xp,Tr)
in.tri.all(Xp,Tr,boundary = FALSE)
```

```
Xp<-runif.tri(n,Tr)$g
in.tri.all(Xp,Tr)
in.tri.all(Xp,Tr,boundary = FALSE)
```

`in.triangle` *Check whether a point is inside a triangle*

Description

Checks if the point `p` lies in the triangle, `tri`, using the barycentric coordinates, generally denoted as (α,β,γ).

If all (normalized or non-normalized) barycentric coordinates are positive then the point `p` is inside the triangle, if all are nonnegative with one or more are zero, then `p` falls in the boundary. If some of the barycentric coordinates are negative, then `p` falls outside the triangle.

`boundary` is a logical argument (default=`TRUE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `p`, lies in the closure of the triangle (i.e., interior and boundary combined); else, it checks if `p` lies in the interior of the triangle.

Usage

```
in.triangle(p, tri, boundary = TRUE)
```

Arguments

`p`	A 2D point to be checked whether it is inside the triangle or not.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`boundary`	A logical parameter (default=`TRUE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `p`, lies in the closure of the triangle (i.e., interior and boundary combined); else, it checks if `p` lies in the interior of the triangle.

Value

A `list` with two elements

`in.tri`	A logical output, it is `TRUE`, if the point, `p`, is inside the triangle, `tri`, else it is `FALSE`.
`barycentric`	The barycentric coordinates (α,β,γ) of the point `p` with respect to the triangle, `tri`.

Author(s)

Elvan Ceyhan

See Also

`in.tri.all` and `on.convex.hull` from the `interp` package for documentation for `in.convex.hull`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2); p<-c(1.4,1.2)
Tr<-rbind(A,B,C)
in.triangle(p,Tr)

p<-c(.4,-.2)
in.triangle(p,Tr)

#for the vertex A
in.triangle(A,Tr)
in.triangle(A,Tr,boundary = FALSE)

#for a point on the edge AB
D3<-(A+B)/2
in.triangle(D3,Tr)
in.triangle(D3,Tr,boundary = FALSE)

#for a NA entry point
p<-c(NA,.2)
in.triangle(p,Tr)
```

`inci.matAS` *Incidence matrix for Arc Slice Proximity Catch Digraphs (AS-PCDs) - multiple triangle case*

Description

Returns the incidence matrix for the AS-PCD whose vertices are a given 2D numerical data set, `Xp`, in the convex hull of `Yp` which is partitioned by the Delaunay triangles based on `Yp` points.

AS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points and vertex regions are based on the center `M="CC"` for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is `M="CC"` i.e., circumcenter of each triangle.

Each Delaunay triangle is first converted to an (nonscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the incidence matrix loops are allowed, so the diagonal entries are all equal to 1.

See (Ceyhan (2005, 2010)) for more on AS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
inci.matAS(Xp, Yp, M = "CC")
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`M`	The center of the triangle. `"CC"` stands for circumcenter of each Delaunay triangle or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is `M="CC"` i.e., the circumcenter of each triangle.

Value

Incidence matrix for the AS-PCD whose vertices are the 2D data set, `Xp`, and AS proximity regions are defined in the Delaunay triangles based on `Yp` points.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`inci.matAStri`, `inci.matPE`, and `inci.matCS`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))
```

```
M<-"CC"  #try also M<-c(1,1,1)

IM<-inci.matAS(Xp,Yp,M)
IM
dom.num.greedy(IM)  #try also dom.num.exact(IM)  #this might take a long time for large  nx

IM<-inci.matAS(Xp,Yp[1:3,],M)

inci.matAS(Xp,rbind(Yp,Yp))
```

`inci.matAStri` *Incidence matrix for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns the incidence matrix of the AS-PCD whose vertices are the given 2D numerical data set, `Xp`, in the triangle `tri`$= T(v = 1, v = 2, v = 3)$.

AS proximity regions are defined with respect to the triangle `tri`$= T(v = 1, v = 2, v = 3)$ and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005, 2010)).

Usage

```
inci.matAStri(Xp, tri, M = "CC")
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M="CC"` i.e., the circumcenter of `tri`.

Value

Incidence matrix for the AS-PCD whose vertices are the 2D data set, `Xp`, and AS proximity regions are defined with respect to the triangle `tri` and vertex regions based on the center `M`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`inci.matAS`, `inci.matPEtri`, and `inci.matCStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

IM<-inci.matAStri(Xp,Tr,M)
IM

dom.num.greedy(IM)
dom.num.exact(IM)
```

`inci.matCS` *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) - multiple triangle case*

Description

Returns the incidence matrix of Central Similarity Proximity Catch Digraph (CS-PCD) whose vertices are the data points in `Xp` in the multiple triangle case.

CS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $t > 0$ and edge regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (nonscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the incidence matrix loops are allowed, so the diagonal entries are all equal to 1.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) for more on CS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
inci.matCS(Xp, Yp, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle, default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

Incidence matrix for the CS-PCD with vertices being 2D data set, `Xp`. CS proximity regions are constructed with respect to the Delaunay triangles and `M`-edge regions.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`inci.matCStri`, `inci.matCSstd.tri`, `inci.matAS`, and `inci.matPE`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

t<-1.5  #try also t<-2

IM<-inci.matCS(Xp,Yp,t,M)
IM
dom.num.greedy(IM) #try also dom.num.exact(IM)  #takes a very long time for large nx, try smaller nx
Idom.num.up.bnd(IM,3)  #takes a very long time for large nx, try smaller nx
```

---

inci.matCS1D *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) for 1D data - multiple interval case*

Description

Returns the incidence matrix for the CS-PCD for a given 1D numerical data set, `Xp`, as the vertices of the digraph and `Yp` determines the end points of the intervals (in the multi-interval case). If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed. Loops are allowed, so the diagonal entries are all equal to 1.

CS proximity region is constructed with an expansion parameter $t > 0$ and a centrality parameter $c \in (0, 1)$.

See also (Ceyhan (2016)).

Usage

```
inci.matCS1D(Xp, Yp, t, c = 0.5)
```

Arguments

`Xp`	a set of 1D points which constitutes the vertices of the digraph.
`Yp`	a set of 1D points which constitutes the end points of the intervals that partition the real line.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

Incidence matrix for the CS-PCD with vertices being 1D data set, `Xp`, and `Yp` determines the end points of the intervals (the multi-interval case)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`inci.matCS1D`, `inci.matPEtri`, and `inci.matPE`

Examples

```
t<-2
c<-.4
a<-0; b<-10;
nx<-10; ny<-4

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

IM<-inci.matCS1D(Xp,Yp,t,c)
IM
dom.num.greedy(IM)

dom.num.exact(IM)  #might take a long time depending on nx

Idom.num.up.bnd(IM,5)

Arcs<-arcsCS1D(Xp,Yp,t,c)
Arcs
summary(Arcs)
```

```
plot(Arcs)

inci.matCS1D(Xp,Yp+10,t,c)

t<-2
c<-.4
a<-0; b<-10;
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

inci.matCS1D(Xp,Yp,t,c)
```

`inci.matCSint` *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) for 1D data - one interval case*

Description

Returns the incidence matrix for the CS-PCD for a given 1D numerical data set, `Xp`, as the vertices of the digraph and `int` determines the end points of the interval (in the one interval case). Loops are allowed, so the diagonal entries are all equal to 1.

CS proximity region is constructed with an expansion parameter $t > 0$ and a centrality parameter $c \in (0, 1)$.

See also (Ceyhan (2016)).

Usage

```
inci.matCSint(Xp, int, t, c = 0.5)
```

Arguments

`Xp`	a set of 1D points which constitutes the vertices of the digraph.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

Incidence matrix for the CS-PCD with vertices being 1D data set, `Xp`, and `int` determines the end points of the intervals (in the one interval case)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`inci.matCS1D`, `inci.matPE1D`, `inci.matPEtri`, and `inci.matPE`

Examples

```
c<-.4
t<-1
a<-0; b<-10; int<-c(a,b)

xf<-(int[2]-int[1])*.1

set.seed(123)

n<-10
Xp<-runif(n,a-xf,b+xf)

IM<-inci.matCSint(Xp,int,t,c)
IM

dom.num.greedy(IM)
Idom.num.up.bnd(IM,3)
dom.num.exact(IM)

inci.matCSint(Xp,int+10,t,c)
```

`inci.matCSstd.tri` *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns the incidence matrix for the CS-PCD whose vertices are the given 2D numerical data set, `Xp`, in the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is

$M = (1, 1, 1)$ i.e., the center of mass of T_e. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
inci.matCSstd.tri(Xp, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates. which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.

Value

Incidence matrix for the CS-PCD with vertices being 2D data set, `Xp` and CS proximity regions are defined in the standard equilateral triangle T_e with `M`-edge regions.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`inci.matCStri`, `inci.matCS` and `inci.matPEstd.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.std.tri(n)$gen.points
```

```
M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

inc.mat<-inci.matCSstd.tri(Xp,t=1.25,M)
inc.mat
sum(inc.mat)-n
num.arcsCSstd.tri(Xp,t=1.25)

dom.num.greedy(inc.mat) #try also dom.num.exact(inc.mat)  #might take a long time for large n
Idom.num.up.bnd(inc.mat,1)
```

`inci.matCStri` *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns the incidence matrix for the CS-PCD whose vertices are the given 2D numerical data set, `Xp`, in the triangle `tri`$= T(v = 1, v = 2, v = 3)$.

CS proximity regions are constructed with respect to triangle `tri` with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
inci.matCStri(Xp, tri, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of CS-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

Value

Incidence matrix for the CS-PCD with vertices being 2D data set, `Xp`, in the triangle `tri` with edge regions based on center `M`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`inci.matCS`, `inci.matPEtri`, and `inci.matAStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

IM<-inci.matCStri(Xp,Tr,t=1.25,M)
IM
dom.num.greedy(IM) #try also dom.num.exact(IM)
Idom.num.up.bnd(IM,3)

inci.matCStri(Xp,Tr,t=1.5,M)
```

`inci.matPE` *Incidence matrix for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - multiple triangle case*

Description

Returns the incidence matrix of Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in `Xp` in the multiple triangle case.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the incidence matrix loops are allowed, so the diagonal entries are all equal to 1.

See (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)) for more on the PE-PCDs. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
inci.matPE(Xp, Yp, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this, argument should be set as `M="CC"`), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

Incidence matrix for the PE-PCD with vertices being 2D data set, `Xp`. PE proximity regions are constructed with respect to the Delaunay triangles and `M`-vertex regions.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`inci.matPEtri`, `inci.matPEstd.tri`, `inci.matAS`, and `inci.matCS`

Examples

```
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

r<-1.5  #try also r<-2

IM<-inci.matPE(Xp,Yp,r,M)
IM
dom.num.greedy(IM)
#try also dom.num.exact(IM)
#might take a long time in this brute-force fashion ignoring the
#disconnected nature of the digraph inherent by the geometric construction of it
```

`inci.matPE1D` *Incidence matrix for Proportional-Edge Proximity Catch Digraphs (PE-PCDs) for 1D data - multiple interval case*

Description

Returns the incidence matrix for the PE-PCD for a given 1D numerical data set, `Xp`, as the vertices of the digraph and `Yp` determines the end points of the intervals (in the multi-interval case). If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed. Loops are allowed, so the diagonal entries are all equal to 1.

PE proximity region is constructed with an expansion parameter $r \geq 1$ and a centrality parameter $c \in (0, 1)$.

See also (Ceyhan (2012)).

Usage

```
inci.matPE1D(Xp, Yp, r, c = 0.5)
```

Arguments

`Xp`	a set of 1D points which constitutes the vertices of the digraph.
`Yp`	a set of 1D points which constitutes the end points of the intervals that partition the real line.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

Incidence matrix for the PE-PCD with vertices being 1D data set, `Xp`, and `Yp` determines the end points of the intervals (in the multi-interval case)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`inci.matCS1D`, `inci.matPEtri`, and `inci.matPE`

Examples

```
r<-2
c<-.4
a<-0; b<-10;
nx<-10; ny<-4

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

IM<-inci.matPE1D(Xp,Yp,r,c)
IM
```

```
dom.num.greedy(IM)
Idom.num.up.bnd(IM,6)
dom.num.exact(IM)
```

`inci.matPEint` *Incidence matrix for Proportional-Edge Proximity Catch Digraphs (PE-PCDs) for 1D data - one interval case*

Description

Returns the incidence matrix for the PE-PCD for a given 1D numerical data set, `Xp`, as the vertices of the digraph and `int` determines the end points of the interval (in the one interval case). Loops are allowed, so the diagonal entries are all equal to 1.

PE proximity region is constructed with an expansion parameter $r \geq 1$ and a centrality parameter $c \in (0, 1)$.

See also (Ceyhan (2012)).

Usage

```
inci.matPEint(Xp, int, r, c = 0.5)
```

Arguments

`Xp`	a set of 1D points which constitutes the vertices of the digraph.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

Incidence matrix for the PE-PCD with vertices being 1D data set, `Xp`, and `int` determines the end points of the intervals (in the one interval case)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`inci.matCSint`, `inci.matPE1D`, `inci.matPEtri`, and `inci.matPE`

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

xf<-(int[2]-int[1])*.1

set.seed(123)

n<-10
Xp<-runif(n,a-xf,b+xf)

IM<-inci.matPEint(Xp,int,r,c)
IM

dom.num.greedy(IM)
Idom.num.up.bnd(IM,6)
dom.num.exact(IM)

inci.matPEint(Xp,int+10,r,c)
```

`inci.matPEstd.tri` *Incidence matrix for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case*

Description

Returns the incidence matrix for the PE-PCD whose vertices are the given 2D numerical data set, `Xp`, in the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$.

PE proximity region is constructed with respect to the standard equilateral triangle T_e with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$, i.e., the center of mass of T_e. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005, 2010)).

Usage

```
inci.matPEstd.tri(Xp, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.

Value

Incidence matrix for the PE-PCD with vertices being 2D data set, `Xp` in the standard equilateral triangle where PE proximity regions are defined with `M`-vertex regions.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

`inci.matPEtri`, `inci.matPE`, and `inci.matCSstd.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
n<-10

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

inc.mat<-inci.matPEstd.tri(Xp,r=1.25,M)
inc.mat
sum(inc.mat)-n
num.arcsPEstd.tri(Xp,r=1.25)

dom.num.greedy(inc.mat)
```

```
Idom.num.up.bnd(inc.mat,2) #try also dom.num.exact(inc.mat)
```

`inci.matPEtetra` *Incidence matrix for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns the incidence matrix for the PE-PCD whose vertices are the given 3D numerical data set, `Xp`, in the tetrahedron $th = T(v = 1, v = 2, v = 3, v = 4)$.

PE proximity regions are constructed with respect to tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions are based on the center `M` which is circumcenter (`"CC"`) or center of mass (`"CM"`) of `th` with default=`"CM"`. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005, 2010)).

Usage

```
inci.matPEtetra(Xp, th, r, M = "CM")
```

Arguments

`Xp`	A set of 3D points which constitute the vertices of PE-PCD.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.

Value

Incidence matrix for the PE-PCD with vertices being 3D data set, `Xp`, in the tetrahedron `th` with vertex regions based on circumcenter or center of mass

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`inci.matPEtri`, `inci.matPE1D`, and `inci.matPE`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-5

Xp<-runif.tetra(n,tetra)$g  #try also Xp<-c(.5,.5,.5)

M<-"CM"  #try also M<-"CC"
r<-1.5

IM<-inci.matPEtetra(Xp,tetra,r=1.25)  #uses the default M="CM"
IM<-inci.matPEtetra(Xp,tetra,r=1.25,M)
IM
dom.num.greedy(IM)
Idom.num.up.bnd(IM,3) #try also dom.num.exact(IM)  #this might take a long time for large n
```

`inci.matPEtri` *Incidence matrix for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns the incidence matrix for the PE-PCD whose vertices are the given 2D numerical data set, `Xp`, in the triangle `tri`$= T(v=1,v=2,v=3)$.

PE proximity regions are constructed with respect to triangle `tri` with expansion parameter $r \geq 1$ and vertex regions are based on the center $M=(m_1,m_2)$ in Cartesian coordinates or $M=(\alpha,\beta,\gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M=(1,1,1)$, i.e., the center of mass of `tri`. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
inci.matPEtri(Xp, tri, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of PE-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.

Value

Incidence matrix for the PE-PCD with vertices being 2D data set, `Xp`, in the triangle `tri` with vertex regions based on center `M`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`inci.matPE`, `inci.matCStri`, and `inci.matAStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)
IM<-inci.matPEtri(Xp,Tr,r=1.25,M)

IM
dom.num.greedy(IM) #try also dom.num.exact(IM)
Idom.num.up.bnd(IM,3)
```

`index.six.Te` *Region index inside the Gamma-1 region*

Description

Returns the region index of the point `p` for the 6 regions in standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$, starting with 1 on the first one-sixth of the triangle, and numbering follows the counter-clockwise direction (see the plot in the examples). These regions are in the inner hexagon which is the Gamma-1 region for CS-PCD with $t = 1$ if `p` is not in any of the 6 regions the function returns `NA`.

Usage

```
index.six.Te(p)
```

Arguments

`p`	A 2D point whose index for the 6 regions in standard equilateral triangle T_e is determined.

Value

rel An integer between 1-6 (inclusive) or `NA`

Author(s)

Elvan Ceyhan

See Also

runif.std.tri.onesixth

Examples

```
P<-c(.4,.2)
index.six.Te(P)

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

h1<-c(1/2,sqrt(3)/18); h2<-c(2/3, sqrt(3)/9); h3<-c(2/3, 2*sqrt(3)/9);
h4<-c(1/2, 5*sqrt(3)/18); h5<-c(1/3, 2*sqrt(3)/9); h6<-c(1/3, sqrt(3)/9);

r1<-(h1+h6+CM)/3;r2<-(h1+h2+CM)/3;r3<-(h2+h3+CM)/3;
r4<-(h3+h4+CM)/3;r5<-(h4+h5+CM)/3;r6<-(h5+h6+CM)/3;
```

```
Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
polygon(rbind(h1,h2,h3,h4,h5,h6))

txt<-rbind(h1,h2,h3,h4,h5,h6)
xc<-txt[,1]+c(-.02,.02,.02,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,0,0,0)
txt.str<-c("h1","h2","h3","h4","h5","h6")
text(xc,yc,txt.str)

txt<-rbind(Te,CM,r1,r2,r3,r4,r5,r6)
xc<-txt[,1]+c(-.02,.02,.02,0,0,0,0,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,0,0,0,0,0,0,0)
txt.str<-c("A","B","C","CM","1","2","3","4","5","6")
text(xc,yc,txt.str)

n<-10  #try also n<-40
Xp<-runif.std.tri(n)$gen.points

Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

rsix<-vector()
for (i in 1:n)
  rsix<-c(rsix,index.six.Te(Xp[i,]))
rsix

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(Xp,pch=".")
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
polygon(rbind(h1,h2,h3,h4,h5,h6))
text(Xp,labels=factor(rsix))

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.02,.02,.02,0)
yc<-txt[,2]+c(.02,.02,.02,-.05)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)
```

`intersect.line.circle` *The points of intersection of a line and a circle*

Description

Returns the intersection point(s) of a line and a circle. The line is determined by the two points `p1` and `p2` and the circle is centered at point `cent` and has radius `rad`. If the circle does not intersect the line, the function yields `NULL`; if the circle intersects at only one point, it yields only that point; otherwise it yields both intersection points as output. When there are two intersection points, they are listed in the order of the x-coordinates of `p1` and `p2`; and if the x-coordinates of `p1` and `p2` are equal, intersection points are listed in the order of y-coordinates of `p1` and `p2`.

Usage

```
intersect.line.circle(p1, p2, cent, rad)
```

Arguments

`p1, p2`	2D points that determine the straight line (i.e., through which the straight line passes).
`cent`	A 2D point representing the center of the circle.
`rad`	A positive real number representing the radius of the circle.

Value

point(s) of intersection between the circle and the line (if they do not intersect, the function yields `NULL` as the output)

Author(s)

Elvan Ceyhan

See Also

`intersect2lines`

Examples

```
P1<-c(.3,.2)*100
P2<-c(.6,.3)*100
cent<-c(1.1,1.1)*100
rad<-2*100

intersect.line.circle(P1,P2,cent,rad)
intersect.line.circle(P2,P1,cent,rad)
intersect.line.circle(P1,P1+c(0,1),cent,rad)
intersect.line.circle(P1+c(0,1),P1,cent,rad)
```

```
dist.point2line(cent,P1,P2)
rad2<-dist.point2line(cent,P1,P2)$d
intersect.line.circle(P1,P2,cent,rad2)
intersect.line.circle(P1,P2,cent,rad=.8)
intersect.line.circle(P1,P2,cent,rad=.78)

#plot of the line and the circle
A<-c(.3,.2); B<-c(.6,.3); cent<-c(1,1); rad<-2 #check dist.point2line(cent,A,B)$dis, .3

IPs<-intersect.line.circle(A,B,cent,rad)

xr<-range(A[1],B[1],cent[1])
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-rad-xf,xr[2]+rad+xf,l=20)  #try also l=100
lnAB<-Line(A,B,x)
y<-lnAB$y

Xlim<-range(x,cent[1])
Ylim<-range(y,A[2],B[2],cent[2]-rad,cent[2]+rad)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(A,B,cent),pch=1,asp=1,xlab="x",ylab="y",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
lines(x,y,lty=1)
interp::circles(cent[1],cent[2],rad)
IP.txt<-c()
if (!is.null(IPs))
{
  for (i in 1:(length(IPs)/2))
    IP.txt<-c(IP.txt,paste("I",i, sep = ""))
}
txt<-rbind(A,B,cent,IPs)
text(txt+cbind(rep(xd*.03,nrow(txt)),rep(-yd*.03,nrow(txt))),c("A","B","M",IP.txt))
```

`intersect.line.plane` *The point of intersection of a line and a plane*

Description

Returns the point of the intersection of the line determined by the 3D points p_1 and p_2 and the plane spanned by 3D points p3, p4, and p5.

Usage

```
intersect.line.plane(p1, p2, p3, p4, p5)
```

Arguments

p1, p2	3D points that determine the straight line (i.e., through which the straight line passes).
p3, p4, p5	3D points that determine the plane (i.e., through which the plane passes).

Value

The coordinates of the point of intersection the line determined by the 3D points p_1 and p_2 and the plane determined by 3D points p3, p4, and p5.

Author(s)

Elvan Ceyhan

See Also

`intersect2lines` and `intersect.line.circle`

Examples

```
L1<-c(2,4,6); L2<-c(1,3,5);
A<-c(1,10,3); B<-c(1,1,3); C<-c(3,9,12)

Pint<-intersect.line.plane(L1,L2,A,B,C)
Pint
pts<-rbind(L1,L2,A,B,C,Pint)

tr<-max(Dist(L1,L2),Dist(L1,Pint),Dist(L2,Pint))
tf<-tr*1.1 #how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf,tf,l=5)  #try also l=10, 20, or 100

lnAB3D<-Line3D(L1,L2,tsq)
xl<-lnAB3D$x
yl<-lnAB3D$y
zl<-lnAB3D$z

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1
#how far to go at the lower and upper ends in the y-coordinate
xp<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100
yp<-seq(yr[1]-yf,yr[2]+yf,l=5)  #try also l=10, 20, or 100

plABC<-Plane(A,B,C,xp,yp)
z.grid<-plABC$z

res<-persp(xp,yp,z.grid, xlab="x",ylab="y",zlab="z",theta = -30,
phi = 30, expand = 0.5,
col = "lightblue", ltheta = 120, shade = 0.05, ticktype = "detailed")
lines (trans3d(xl, yl, zl, pmat = res), col = 3)
```

```
Xlim<-range(xl,pts[,1])
Ylim<-range(yl,pts[,2])
Zlim<-range(zl,pts[,3])

xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::persp3D(z = z.grid, x = xp, y = yp, theta =225, phi = 30,
ticktype = "detailed"
,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),zlim=Zlim+zd*c(-.1,.1),
expand = 0.7, facets = FALSE, scale = TRUE)
        #plane spanned by points A, B, C
#add the defining points
plot3D::points3D(pts[,1],pts[,2],pts[,3], pch = ".", col = "black",
bty = "f", cex = 5,add=TRUE)
plot3D::points3D(Pint[1],Pint[2],Pint[3], pch = "*", col = "red",
bty = "f", cex = 5,add=TRUE)
plot3D::lines3D(xl, yl, zl, bty = "g", cex = 2,
ticktype = "detailed",add=TRUE)
```

`intersect2lines` *The point of intersection of two lines defined by two pairs of points*

Description

Returns the intersection of two lines, first line passing through points `p1` and `q1` and second line passing through points `p2` and `q2`. The points are chosen so that lines are well defined.

Usage

```
intersect2lines(p1, q1, p2, q2)
```

Arguments

`p1, q1`	2D points that determine the first straight line (i.e., through which the first straight line passes).
`p2, q2`	2D points that determine the second straight line (i.e., through which the second straight line passes).

Value

The coordinates of the point of intersection of the two lines, first passing through points `p1` and `q1` and second passing through points `p2` and `q2`.

Author(s)

Elvan Ceyhan

See Also

`intersect.line.circle` and `dist.point2line`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75); C<-c(0,6); D<-c(3,-2)

ip<-intersect2lines(A,B,C,D)
ip
pts<-rbind(A,B,C,D,ip)
xr<-range(pts[,1])
xf<-abs(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100
lnAB<-Line(A,B,x)
lnCD<-Line(C,D,x)

y1<-lnAB$y
y2<-lnCD$y
Xlim<-range(x,pts)
Ylim<-range(y1,y2,pts)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
pf<-c(xd,-yd)*.025

#plot of the line joining A and B
plot(rbind(A,B,C,D),pch=1,xlab="x",ylab="y",
main="Point of Intersection of Two Lines",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
lines(x,y1,lty=1,col=1)
lines(x,y2,lty=1,col=2)
text(rbind(A+pf,B+pf),c("A","B"))
text(rbind(C+pf,D+pf),c("C","D"))
text(rbind(ip+pf),c("intersection\n point"))
```

`interval.indices.set` *Indices of the intervals where the 1D point(s) reside*

Description

Returns the indices of intervals for all the points in 1D data set, `Xp`, as a vector.

Intervals are based on `Yp` and left end interval is labeled as 1, the next interval as 2, and so on. If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

Usage

```
interval.indices.set(Xp, Yp)
```

Arguments

`Xp`	A set of 1D points for which the indices of intervals are to be determined.
`Yp`	A set of 1D points from which intervals are constructed.

Value

The `vector` of indices of the intervals in which points in the 1D data set, `Xp`, reside

Author(s)

Elvan Ceyhan

Examples

```
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.1
Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)  #try also Yp<-runif(ny,a+1,b-1)

ind<-interval.indices.set(Xp,Yp)
ind

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0), xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
text(Xp,yjit,labels=factor(ind))
```

`is.in.data` *Check a point belong to a given data set*

Description

returns `TRUE` if the point `p` of any dimension is inside the data set `Xp` of the same dimension as `p`; otherwise returns `FALSE`.

Usage

```
is.in.data(p, Xp)
```

Arguments

`p`	A 2D point for which the function checks membership to the data set `Xp`.
`Xp`	A set of 2D points representing the set of data points.

Value

`TRUE` if `p` belongs to the data set `Xp`.

Author(s)

Elvan Ceyhan

Examples

```
n<-10
Xp<-cbind(runif(n),runif(n))

P<-Xp[7,]
is.in.data(P,Xp)
is.in.data(P,Xp[7,])

P<-Xp[7,]+10^(-7)
is.in.data(P,Xp)

P<-Xp[7,]+10^(-9)
is.in.data(P,Xp)

is.in.data(P,P)

is.in.data(c(2,2),c(2,2))

#for 1D data
n<-10
Xp<-runif(n)

P<-Xp[7]
```

```
is.in.data(P,Xp[7])  #this works because both entries are treated as 1D vectors but
#is.in.data(P,Xp) does not work since entries are treated as vectors of different dimensions

Xp<-as.matrix(Xp)
is.in.data(P,Xp)
#this works, because P is a 1D point, and Xp is treated as a set of 10 1D points

P<-Xp[7]+10^(-7)
is.in.data(P,Xp)

P<-Xp[7]+10^(-9)
is.in.data(P,Xp)

is.in.data(P,P)

#for 3D data
n<-10
Xp<-cbind(runif(n),runif(n),runif(n))

P<-Xp[7,]
is.in.data(P,Xp)
is.in.data(P,Xp[7,])

P<-Xp[7,]+10^(-7)
is.in.data(P,Xp)

P<-Xp[7,]+10^(-9)
is.in.data(P,Xp)

is.in.data(P,P)

n<-10
Xp<-cbind(runif(n),runif(n))
P<-Xp[7,]
is.in.data(P,Xp)
```

`is.point` *Check the argument is a point of a given dimension*

Description

Returns `TRUE` if the argument `p` is a `numeric` point of dimension `dim` (default is `dim=2`); otherwise returns `FALSE`.

Usage

```
is.point(p, dim = 2)
```

Arguments

`p`	A `vector` to be checked to see it is a point of dimension `dim` or not.
`dim`	A positive integer representing the dimension of the argument `p`.

Value

`TRUE` if `p` is a `vector` of dimension `dim`.

Author(s)

Elvan Ceyhan

See Also

`dimension`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75,4)
is.point(A)
is.point(A,1)

is.point(B)
is.point(B,3)
```

`is.std.eq.tri` *Check whether a triangle is a standard equilateral triangle*

Description

Checks whether the triangle, `tri`, is the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ or not.

Usage

```
is.std.eq.tri(tri)
```

Arguments

`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

`TRUE` if `tri` is a standard equilateral triangle, else `FALSE`.

Author(s)

Elvan Ceyhan

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)  #try adding +10^(-16) to each vertex
is.std.eq.tri(Te)

is.std.eq.tri(rbind(B,C,A))

Tr<-rbind(A,B,-C)
is.std.eq.tri(Tr)

A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
is.std.eq.tri(Tr)
```

`kfr2vertsCCvert.reg` *The* `k` *furthest points in a data set from vertices in each CC-vertex region in a triangle*

Description

An object of class `"Extrema"`. Returns the `k` furthest data points among the data set, `Xp`, in each CC-vertex region from the vertex in the triangle, `tri`$= T(A, B, C)$, vertices are stacked row-wise. Vertex region labels/numbers correspond to the row number of the vertex in `tri`.

`ch.all.intri` is for checking whether all data points are inside `tri` (default is `FALSE`). If some of the data points are not inside `tri` and `ch.all.intri=TRUE`, then the function yields an error message. If some of the data points are not inside `tri` and `ch.all.intri=FALSE`, then the function yields the closest points to edges among the data points inside `tri` (yields `NA` if there are no data points inside `tri`).

In the extrema, ext, in the output, the first `k` entries are the `k` furthest points from vertex 1, second `k` entries are `k` furthest points are from vertex 2, and last `k` entries are the `k` furthest points from vertex 3. If data size does not allow, `NA`'s are inserted for some or all of the `k` furthest points for each vertex.

Usage

```
kfr2vertsCCvert.reg(Xp, tri, k, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`k`	A positive integer. `k` furthest data points in each CC-vertex region are to be found if exists, else `NA` are provided for (some of) the `k` furthest points.
`ch.all.intri`	A logical argument (default=`FALSE`) to check whether all data points are inside the triangle `tri`. So, if it is `TRUE`, the function checks if all data points are inside the closure of the triangle (i.e., interior and boundary combined) else it does not.

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A shorter description of the distances as `"Distances of k furthest points in the vertex regions to Vertices"`.
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, `k` furthest points from vertices in each CC-vertex region in the triangle `tri`.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, it is `tri` for this function.
`cent`	The center point used for construction of vertex regions
`ncent`	Name of the center, `cent`, it is circumcenter `"CC"` for this function.
`regions`	Vertex regions inside the triangle, `tri`, provided as a `list`
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside T_b.
`dist2ref`	Distances from `k` furthest points in each vertex region to the corresponding vertex (each row representing a vertex in `tri`). Among the distances the first `k` entries are the distances from the `k` furthest points from vertex 1 to vertex 1, second `k` entries are distances from the `k` furthest points from vertex 2 to vertex 2, and the last `k` entries are the distances from the `k` furthest points from vertex 3 to vertex 3.

Author(s)

Elvan Ceyhan

See Also

`fr2vertsCCvert.reg.basic.tri`, `fr2vertsCCvert.reg.basic.tri`, `fr2vertsCCvert.reg`, and `fr2edgesCMedge.reg.std.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20
k<-3

set.seed(1)
Xp<-runif.tri(n,Tr)$g

Ext<-kfr2vertsCCvert.reg(Xp,Tr,k)
Ext
summary(Ext)
plot(Ext)

Xp2<-rbind(Xp,c(.2,.4))
kfr2vertsCCvert.reg(Xp2,Tr,k)
#try also kfr2vertsCCvert.reg(Xp2,Tr,k,ch.all.intri = TRUE)

kf2v<-Ext

CC<-circumcenter.tri(Tr)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",
main=paste(k," Furthest Points in CC-Vertex Regions \n from the Vertices",sep=""),
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(Xp)
points(kf2v$ext,pch=4,col=2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.06,.08,.05,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.04,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
```

`kfr2vertsCCvert.reg.basic.tri`

The k *furthest points from vertices in each CC-vertex region in a standard basic triangle*

Description

An object of class "`Extrema`". Returns the `k` furthest data points among the data set, `Xp`, in each CC-vertex region from the vertex in the standard basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2))$.

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle is useful for simulation studies under the uniformity hypothesis.

`ch.all.intri` is for checking whether all data points are inside T_b (default is `FALSE`). In the extrema, ext, in the output, the first `k` entries are the `k` furthest points from vertex 1, second `k` entries are `k` furthest points are from vertex 2, and last `k` entries are the `k` furthest points from vertex 3 If data size does not allow, `NA`'s are inserted for some or all of the `k` furthest points for each vertex.

Usage

```
kfr2vertsCCvert.reg.basic.tri(Xp, c1, c2, k, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points representing the set of data points.
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle. adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$
`k`	A positive integer. `k` furthest data points in each CC-vertex region are to be found if exists, else `NA` are provided for (some of) the `k` furthest points.
`ch.all.intri`	A logical argument for checking whether all data points are inside T_b (default is `FALSE`).

Value

A `list` with the elements

`txt1`	Vertex labels are $A = 1$, $B = 2$, and $C = 3$ (correspond to row number in Extremum Points).
`txt2`	A shorter description of the distances as "`Distances of k furthest points in the vertex regions to Vertices`".
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The "`main`" title for the plot of the extrema
`ext`	The extrema points, here, `k` furthest points from vertices in each vertex region.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is T_b.
`cent`	The center point used for construction of edge regions.

`ncent`	Name of the center, `cent`, it is circumcenter `"CC"` for this function.
`regions`	Vertex regions inside the triangle, T_b, provided as a `list`.
`region.names`	Names of the vertex regions as `"vr=1"`, `"vr=2"`, and `"vr=3"`
`region.centers`	Centers of mass of the vertex regions inside T_b.
`dist2ref`	Distances from `k` furthest points in each vertex region to the corresponding vertex (each row representing a vertex).

Author(s)

Elvan Ceyhan

See Also

`fr2vertsCCvert.reg.basic.tri`, `fr2vertsCCvert.reg`, `fr2edgesCMedge.reg.std.tri`, and `kfr2vertsCCvert.reg`

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-20
k<-3

set.seed(1)
Xp<-runif.basic.tri(n,c1,c2)$g

Ext<-kfr2vertsCCvert.reg.basic.tri(Xp,c1,c2,k)
Ext
summary(Ext)
plot(Ext)

kf2v<-Ext

CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",
main=paste(k," Furthest Points in CC-Vertex Regions \n from the Vertices",sep=""),
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(Xp)
points(kf2v$ext,pch=4,col=2)
```

```
txt<-rbind(Tb,CC,Ds)
xc<-txt[,1]+c(-.03,.03,.02,.07,.06,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,-.02,.02,.03,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
```

`Line` *The line joining two distinct 2D points* `a` *and* `b`

Description

An object of class `"Lines"`. Returns the `equation`, `slope`, `intercept`, and y-coordinates of the line crossing two distinct 2D points `a` and `b` with x-coordinates provided in `vector` `x`.

This function is different from the `line` function in the standard `stats` package in `R` in the sense that `Line(a,b,x)` fits the line passing through points `a` and `b` and returns various quantities (see below) for this line and `x` is the x-coordinates of the points we want to find on the `Line(a,b,x)` while `line(a,b)` fits the line robustly whose x-coordinates are in `a` and y-coordinates are in `b`.

`Line(a,b,x)` and `line(x,Line(A,B,x)$y)` would yield the same straight line (i.e., the line with the same coefficients.)

Usage

```
Line(a, b, x)
```

Arguments

`a, b`	2D points that determine the straight line (i.e., through which the straight line passes).
`x`	A scalar or a `vector` of scalars representing the x-coordinates of the line.

Value

A `list` with the elements

`desc`	A description of the line
`mtitle`	The `"main"` title for the plot of the line
`points`	The input points `a` and `b` through which the straight line passes (stacked row-wise, i.e., row 1 is point `a` and row 2 is point `b`).
`x`	The input scalar or `vector` which constitutes the x-coordinates of the point(s) of interest on the line.
`y`	The output scalar or `vector` which constitutes the y-coordinates of the point(s) of interest on the line. If `x` is a scalar, then `y` will be a scalar and if `x` is a `vector` of scalars, then `y` will be a `vector` of scalars.

`slope`	Slope of the line, `Inf` is allowed, passing through points `a` and `b`
`intercept`	Intercept of the line passing through points `a` and `b`
`equation`	Equation of the line passing through points `a` and `b`

Author(s)

Elvan Ceyhan

See Also

`slope`, `paraline`, `perpline`, `line` in the generic `stats` package and and `Line3D`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)

xr<-range(A,B);
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100

lnAB<-Line(A,B,x)
lnAB
summary(lnAB)
plot(lnAB)

line(A,B)
#this takes vector A as the x points and vector B as the y points and fits the line
#for example, try
x=runif(100); y=x+(runif(100,-.05,.05))
plot(x,y)
line(x,y)

x<-lnAB$x
y<-lnAB$y
Xlim<-range(x,A,B)
if (!is.na(y[1])) {Ylim<-range(y,A,B)} else {Ylim<-range(A,B)}
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
pf<-c(xd,-yd)*.025

#plot of the line joining A and B
plot(rbind(A,B),pch=1,xlab="x",ylab="y",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
if (!is.na(y[1])) {lines(x,y,lty=1)} else {abline(v=A[1])}
text(rbind(A+pf,B+pf),c("A","B"))
int<-round(lnAB$intercep,2)  #intercept
sl<-round(lnAB$slope,2)  #slope
text(rbind((A+B)/2+pf*3),ifelse(is.na(int),paste("x=",A[1]),
ifelse(sl==0,paste("y=",int),
ifelse(sl==1,ifelse(sign(int)<0,paste("y=x",int),paste("y=x+",int)),
ifelse(sign(int)<0,paste("y=",sl,"x",int),paste("y=",sl,"x+",int))))))
```

`Line3D` *The line crossing 3D point* `p` *in the direction of* `vector v` *(or if* `v` *is a point, in direction of* $v - r\_0$*)*

Description

An object of class `"Lines3D"`. Returns the equation, x-, y-, and z-coordinates of the line crossing 3D point r_0 in the direction of `vector v` (of if `v` is a point, in the direction of $v - r_0$) with the parameter `t` being provided in `vector t`.

Usage

```
Line3D(p, v, t, dir.vec = TRUE)
```

Arguments

`p`	A 3D point through which the straight line passes.
`v`	A 3D `vector` which determines the direction of the straight line (i.e., the straight line would be parallel to this vector) if the `dir.vec=TRUE`, otherwise it is 3D point and $v - r_0$ determines the direction of the the straight line.
`t`	A scalar or a `vector` of scalars representing the parameter of the coordinates of the line (for the form: $x = p_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $r_0 = (p_0, y_0, z_0)$ and $v = (a, b, c)$ if `dir.vec=TRUE`, else $v - r_0 = (a, b, c)$).
`dir.vec`	A logical argument about `v`, if `TRUE` `v` is treated as a vector, else `v` is treated as a point and so the direction `vector` is taken to be $v - r_0$.

Value

A `list` with the elements

`desc`	A description of the line
`mtitle`	The `"main"` title for the plot of the line
`pts`	The input points that determine a line and/or a plane, `NULL` for this function.
`pnames`	The names of the input points that determine a line and/or a plane, `NULL` for this function.
`vecs`	The point `p` and the `vector v` (if `dir.vec=TRUE`) or the point `v` (if `dir.vec=FALSE`). The first row is `p` and the second row is `v`.
`vec.names`	The names of the point `p` and the `vector v` (if `dir.vec=TRUE`) or the point `v` (if `dir.vec=FALSE`).
`x, y, z`	The x-, y-, and z-coordinates of the point(s) of interest on the line.

`tsq`	The scalar or the `vector` of the parameter in defining each coordinate of the line for the form: $x = p_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $r_0 = (p_0, y_0, z_0)$ and $v = (a, b, c)$ if `dir.vec=TRUE`, else $v - r_0 = (a, b, c)$.
`equation`	Equation of the line passing through point `p` in the direction of the `vector` `v` (if `dir.vec=TRUE`) else in the direction of $v - r_0$. The line equation is in the form: $x = p_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $r_0 = (p_0, y_0, z_0)$ and $v = (a, b, c)$ if `dir.vec=TRUE`, else $v - r_0 = (a, b, c)$.

Author(s)

Elvan Ceyhan

See Also

`line`, `paraline3D`, and `Plane`

Examples

```
A<-c(1,10,3); B<-c(1,1,3);

vecs<-rbind(A,B)

Line3D(A,B,.1)
Line3D(A,B,.1,dir.vec=FALSE)

tr<-range(vecs);
tf<-(tr[2]-tr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=5)  #try also l=10, 20, or 100

lnAB3D<-Line3D(A,B,tsq)
#try also lnAB3D<-Line3D(A,B,tsq,dir.vec=FALSE)
lnAB3D
summary(lnAB3D)
plot(lnAB3D)

x<-lnAB3D$x
y<-lnAB3D$y
z<-lnAB3D$z

zr<-range(z)
zf<-(zr[2]-zr[1])*.2
Bv<-B*tf*5

Xlim<-range(x)
Ylim<-range(y)
Zlim<-range(z)

xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]
```

```
Dr<-A+min(tsq)*B

plot3D::lines3D(x, y, z, phi = 0, bty = "g",
main="Line Crossing A \n in the Direction of OB",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),
zlim=Zlim+zd*c(-.1,.1),
        pch = 20, cex = 2, ticktype = "detailed")
plot3D::arrows3D(Dr[1],Dr[2],Dr[3]+zf,Dr[1]+Bv[1],
Dr[2]+Bv[2],Dr[3]+zf+Bv[3], add=TRUE)
plot3D::points3D(A[1],A[2],A[3],add=TRUE)
plot3D::arrows3D(A[1],A[2],A[3]-2*zf,A[1],A[2],A[3],lty=2, add=TRUE)
plot3D::text3D(A[1],A[2],A[3]-2*zf,labels="initial point",add=TRUE)
plot3D::text3D(A[1],A[2],A[3]+zf/2,labels=expression(r[0]),add=TRUE)
plot3D::arrows3D(Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+3*zf+Bv[3]/2,
Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+zf+Bv[3]/2,lty=2, add=TRUE)
plot3D::text3D(Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+3*zf+Bv[3]/2,
labels="direction vector",add=TRUE)
plot3D::text3D(Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,
Dr[3]+zf+Bv[3]/2,labels="v",add=TRUE)
plot3D::text3D(0,0,0,labels="O",add=TRUE)
```

`NASbasic.tri` *The vertices of the Arc Slice (AS) Proximity Region in the standard basic triangle*

Description

Returns the end points of the line segments and arc-slices that constitute the boundary of AS proximity region for a point in the standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle T_b or based on circumcenter of T_b; default is `M="CC"`, i.e., circumcenter of T_b. `rv` is the index of the vertex region `p` resides, with default=`NULL`.

If `p` is outside T_b, it returns `NULL` for the proximity region. `dec` is the number of decimals (default is 4) to round the barycentric coordinates when checking whether the end points fall on the boundary of the triangle T_b or not (so as not to miss the intersection points due to precision in the decimals).

Any given triangle can be mapped to the standard basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence standard basic triangle is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
NASbasic.tri(p, c1, c2, M = "CC", rv = NULL, dec = 4)
```

Arguments

`p`	A 2D point whose AS proximity region is to be computed.
`c1, c2`	Positive real numbers representing the top vertex in standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \le 1$.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle T_b or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of T_b.
`rv`	The index of the `M`-vertex region containing the point, either `1,2,3` or `NULL` (default is `NULL`).
`dec`	a positive integer the number of decimals (default is 4) to round the barycentric coordinates when checking whether the end points fall on the boundary of the triangle T_b or not.

Value

A `list` with the elements

`L, R`	The end points of the line segments on the boundary of the AS proximity region. Each row in `L` and `R` constitute a line segment on the boundary.
`Arc.Slices`	The end points of the arc-slices on the circular parts of the AS proximity region. Here points in row 1 and row 2 constitute the end points of one arc-slice, points on row 3 and row 4 constitute the end points for the next arc-slice and so on.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`NAStri` and `IarcASbasic.tri`

Examples

```
c1<-.4; c2<-.6  #try also c1<-.2; c2<-.2;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)

set.seed(1)
M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.2)

P1<-as.numeric(runif.basic.tri(1,c1,c2)$g);  #try also P1<-c(.3,.2)
NASbasic.tri(P1,c1,c2)  #default with M="CC"
NASbasic.tri(P1,c1,c2,M)

#or try
Rv<-rel.vert.basic.triCC(P1,c1,c2)$rv
NASbasic.tri(P1,c1,c2,M,Rv)

NASbasic.tri(c(3,5),c1,c2,M)

P2<-c(.5,.4)
NASbasic.tri(P2,c1,c2,M)

P3<-c(1.5,.4)
NASbasic.tri(P3,c1,c2,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

#plot of the NAS region
P1<-as.numeric(runif.basic.tri(1,c1,c2)$g);
CC<-circumcenter.basic.tri(c1,c2)

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
rv<-rel.vert.basic.triCC(P1,c1,c2)$rv
} else
{cent<-M
cent.name<-"M"
Ds<-prj.cent2edges.basic.tri(c1,c2,M)
rv<-rel.vert.basic.tri(P1,c1,c2,M)$rv
}
RV<-Tb[rv,]
rad<-Dist(P1,RV)

Int.Pts<-NASbasic.tri(P1,c1,c2,M)

Xlim<-range(Tb[,1],P1[1]+rad,P1[1]-rad)
Ylim<-range(Tb[,2],P1[2]+rad,P1[2]-rad)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(rbind(Tb,P1,rbind(Int.Pts$L,Int.Pts$R)))
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
interp::circles(P1[1],P1[2],rad,lty=2)
L<-Int.Pts$L; R<-Int.Pts$R
segments(L[,1], L[,2], R[,1], R[,2], lty=1,col=2)
Arcs<-Int.Pts$a;
if (!is.null(Arcs))
{
  K<-nrow(Arcs)/2
  for (i in 1:K)
  {A1<-Arcs[2*i-1,]; A2<-Arcs[2*i,];
  angles<-angle.str2end(A1,P1,A2)$c

  plotrix::draw.arc(P1[1],P1[2],rad,angle1=angles[1],angle2=angles[2],col=2)
  }
}

#proximity region with the triangle (i.e., for labeling the vertices of the NAS)
IP.txt<-intpts<-c()
if (!is.null(Int.Pts$a))
{
 intpts<-unique(round(Int.Pts$a,7))
  #this part is for labeling the intersection points of the spherical
  for (i in 1:(length(intpts)/2))
    IP.txt<-c(IP.txt,paste("I",i+1, sep = ""))
}
txt<-rbind(Tb,P1,cent,intpts)
txt.str<-c("A","B","C","P1",cent.name,IP.txt)
text(txt+cbind(rep(xd*.02,nrow(txt)),rep(-xd*.03,nrow(txt))),txt.str)

c1<-.4; c2<-.6;
P1<-c(.3,.2)
NASbasic.tri(P1,c1,c2,M)
```

`NAStri` *The vertices of the Arc Slice (AS) Proximity Region in a general triangle*

Description

Returns the end points of the line segments and arc-slices that constitute the boundary of AS proximity region for a point in the triangle `tri`$= T(A,B,C) =$`(rv=1,rv=2,rv=3)`.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`. `rv` is the index of the vertex region `p1` resides, with default=`NULL`.

If `p` is outside of `tri`, it returns `NULL` for the proximity region. `dec` is the number of decimals (default is 4) to round the barycentric coordinates when checking the points fall on the boundary of the triangle `tri` or not (so as not to miss the intersection points due to precision in the decimals).

See also (Ceyhan (2005, 2010)).

Usage

```
NAStri(p, tri, M = "CC", rv = NULL, dec = 4)
```

Arguments

`p`	A 2D point whose AS proximity region is to be computed.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is `M="CC"` i.e., the circumcenter of `tri`.
`rv`	Index of the `M`-vertex region containing the point `p`, either `1,2,3` or `NULL` (default is `NULL`).
`dec`	a positive integer the number of decimals (default is 4) to round the barycentric coordinates when checking whether the end points fall on the boundary of the triangle `tri` or not.

Value

A `list` with the elements

`L,R`	End points of the line segments on the boundary of the AS proximity region. Each row in `L` and `R` constitute a pair of points that determine a line segment on the boundary.
`arc.slices`	The end points of the arc-slices on the circular parts of the AS proximity region. Here points in rows 1 and 2 constitute the end points of the first arc-slice, points on rows 3 and 4 constitute the end points for the next arc-slice and so on.
`Angles`	The angles (in radians) between the vectors joining arc slice end points to the point `p` with the horizontal line crossing the point `p`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.”

Computational Geometry: Theory and Applications, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`NASbasic.tri`, `NPEtri`, `NCStri` and `IarcAStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(.6,.2)

P1<-as.numeric(runif.tri(1,Tr)$g)  #try also P1<-c(1.3,1.2)
NAStri(P1,Tr,M)

#or try
Rv<-rel.vert.triCC(P1,Tr)$rv
NAStri(P1,Tr,M,Rv)

NAStri(c(3,5),Tr,M)

P2<-c(1.5,1.4)
NAStri(P2,Tr,M)

P3<-c(1.5,.4)
NAStri(P3,Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

CC<-circumcenter.tri(Tr)  #the circumcenter

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
rv<-rel.vert.triCC(P1,Tr)$rv
} else
{cent<-M
cent.name<-"M"
Ds<-prj.cent2edges(Tr,M)
rv<-rel.vert.tri(P1,Tr,M)$rv
}
RV<-Tr[rv,]
rad<-Dist(P1,RV)
```

```
Int.Pts<-NAStri(P1,Tr,M)

#plot of the NAS region
Xlim<-range(Tr[,1],P1[1]+rad,P1[1]-rad)
Ylim<-range(Tr[,2],P1[2]+rad,P1[2]-rad)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
#asp=1 must be the case to have the arc properly placed in the figure
polygon(Tr)
points(rbind(Tr,P1,rbind(Int.Pts$L,Int.Pts$R)))
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
interp::circles(P1[1],P1[2],rad,lty=2)
L<-Int.Pts$L; R<-Int.Pts$R
segments(L[,1], L[,2], R[,1], R[,2], lty=1,col=2)
Arcs<-Int.Pts$a;
if (!is.null(Arcs))
{
  K<-nrow(Arcs)/2
  for (i in 1:K)
  {A1<-Int.Pts$arc[2*i-1,]; A2<-Int.Pts$arc[2*i,];
  angles<-angle.str2end(A1,P1,A2)$c

  test.ang1<-angles[1]+(.01)*(angles[2]-angles[1])
  test.Pnt<-P1+rad*c(cos(test.ang1),sin(test.ang1))
 if (!in.triangle(test.Pnt,Tr,boundary = TRUE)$i) {angles<-c(min(angles),max(angles)-2*pi)}
  plotrix::draw.arc(P1[1],P1[2],rad,angle1=angles[1],angle2=angles[2],col=2)
  }
}

#proximity region with the triangle (i.e., for labeling the vertices of the NAS)
IP.txt<-intpts<-c()
if (!is.null(Int.Pts$a))
{
 intpts<-unique(round(Int.Pts$a,7))
  #this part is for labeling the intersection points of the spherical
  for (i in 1:(length(intpts)/2))
    IP.txt<-c(IP.txt,paste("I",i+1, sep = ""))
}
txt<-rbind(Tr,P1,cent,intpts)
txt.str<-c("A","B","C","P1",cent.name,IP.txt)
text(txt+cbind(rep(xd*.02,nrow(txt)),rep(-xd*.03,nrow(txt))),txt.str)

P1<-c(.3,.2)
NAStri(P1,Tr,M)
```

`NCSint` *The end points of the Central Similarity (CS) Proximity Region for a point - one interval case*

Description

Returns the end points of the interval which constitutes the CS proximity region for a point in the interval `int=` (a, b) `=(rv=1,rv=2)`.

CS proximity region is constructed with respect to the interval `int` with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$.

Vertex regions are based on the (parameterized) center, M_c, which is $M_c = a + c(b - a)$ for the interval, `int=` (a, b). The CS proximity region is constructed whether `x` is inside or outside the interval `int`.

See also (Ceyhan (2016)).

Usage

```
NCSint(x, int, t, c = 0.5)
```

Arguments

`x`	A 1D point for which CS proximity region is constructed.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int=` (a, b) with the default `c=.5`. For the interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

The interval which constitutes the CS proximity region for the point `x`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`NPEint` and `NCStri`

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

NCSint(7,int,t,c)
NCSint(17,int,t,c)
NCSint(1,int,t,c)
NCSint(-1,int,t,c)

NCSint(3,int,t,c)
NCSint(4,int,t,c)
NCSint(a,int,t,c)
```

`NCStri` *The vertices of the Central Similarity (CS) Proximity Region in a general triangle*

Description

Returns the vertices of the CS proximity region (which is itself a triangle) for a point in the triangle `tri`$= T(A, B, C) =$`(rv=1,rv=2,rv=3)`.

CS proximity region is defined with respect to the triangle `tri` with expansion parameter $t > 0$ and edge regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

Edge regions are labeled as `1,2,3` rowwise for the corresponding vertices of the triangle `tri`. `re` is the index of the edge region `p` resides, with default=`NULL`. If `p` is outside of `tri`, it returns `NULL` for the proximity region.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
NCStri(p, tri, t, M = c(1, 1, 1), re = NULL)
```

Arguments

`p`	A 2D point whose CS proximity region is to be computed.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.
`re`	Index of the `M`-edge region containing the point `p`, either `1,2,3` or `NULL` (default is `NULL`).

Value

Vertices of the triangular region which constitutes the CS proximity region with expansion parameter $t > 0$ and center `M` for a point `p`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`NPEtri`, `NAStri`, and `IarcCStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
tau<-1.5

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

n<-3
set.seed(1)
Xp<-runif.tri(n,Tr)$g

NCStri(Xp[1,],Tr,tau,M)

P1<-as.numeric(runif.tri(1,Tr)$g)  #try also P1<-c(.4,.2)
NCStri(P1,Tr,tau,M)

#or try
re<-rel.edges.tri(P1,Tr,M)$re
NCStri(P1,Tr,tau,M,re)
```

`NPEbasic.tri` *The vertices of the Proportional Edge (PE) Proximity Region in a standard basic triangle*

Description

Returns the vertices of the PE proximity region (which is itself a triangle) for a point in the standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2)) =$`(rv=1,rv=2,rv=3)`.

PE proximity region is defined with respect to the standard basic triangle T_b with expansion parameter $r \geq 1$ and vertex regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the basic triangle T_b or based on the circumcenter of T_b; default is $M = (1,1,1)$, i.e., the center of mass of T_b.

Vertex regions are labeled as $1,2,3$ rowwise for the vertices of the triangle T_b. `rv` is the index of the vertex region `p` resides, with default=`NULL`. If `p` is outside of `tri`, it returns `NULL` for the proximity region.

See also (Ceyhan (2005, 2010)).

Usage

```
NPEbasic.tri(p, r, c1, c2, M = c(1, 1, 1), rv = NULL)
```

Arguments

`p`	A 2D point whose PE proximity region is to be computed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c1, c2`	Positive real numbers representing the top vertex in standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle T_b or the circumcenter of T_b which may be entered as "CC" as well; default is $M = (1,1,1)$, i.e., the center of mass of T_b.
`rv`	Index of the `M`-vertex region containing the point `p`, either `1,2,3` or `NULL` (default is `NULL`).

Value

Vertices of the triangular region which constitutes the PE proximity region with expansion parameter `r` and center `M` for a point `p`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`NPEtri`, `NAStri`, `NCStri`, and `IarcPEbasic.tri`

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.2)

r<-2

P1<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also P1<-c(.4,.2)
NPEbasic.tri(P1,r,c1,c2,M)

#or try
Rv<-rel.vert.basic.tri(P1,c1,c2,M)$rv
NPEbasic.tri(P1,r,c1,c2,M,Rv)

P1<-c(1.4,1.2)
P2<-c(1.5,1.26)
NPEbasic.tri(P1,r,c1,c2,M) #gives an error if M=c(1.3,1.3)
#since center is not the circumcenter or not in the interior of the triangle
```

`NPEint` *The end points of the Proportional Edge (PE) Proximity Region for a point - one interval case*

Description

Returns the end points of the interval which constitutes the PE proximity region for a point in the interval `int`$=(a,b)=$`(rv=1,rv=2)`. PE proximity region is constructed with respect to the interval `int` with expansion parameter $r \geq 1$ and centrality parameter $c \in (0,1)$.

Vertex regions are based on the (parameterized) center, M_c, which is $M_c = a + c(b - a)$ for the interval, `int`$= (a, b)$. The PE proximity region is constructed whether `x` is inside or outside the interval `int`.

See also (Ceyhan (2012)).

Usage

```
NPEint(x, int, r, c = 0.5)
```

Arguments

`x`	A 1D point for which PE proximity region is constructed.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

The interval which constitutes the PE proximity region for the point `x`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

NCSint, NPEtri and NPEtetra

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

NPEint(7,int,r,c)
NPEint(17,int,r,c)
NPEint(1,int,r,c)
NPEint(-1,int,r,c)
```

`NPEstd.tetra` *The vertices of the Proportional Edge (PE) Proximity Region in the standard regular tetrahedron*

Description

Returns the vertices of the PE proximity region (which is itself a tetrahedron) for a point in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3)) =$ `(rv=1,rv=2,rv=3,rv=4)`.

PE proximity region is defined with respect to the tetrahedron T_h with expansion parameter $r \geq 1$ and vertex regions based on the circumcenter of T_h (which is equivalent to the center of mass in the standard regular tetrahedron).

Vertex regions are labeled as `1,2,3,4` rowwise for the vertices of the tetrahedron T_h. `rv` is the index of the vertex region `p` resides, with default=`NULL`. If `p` is outside of T_h, it returns `NULL` for the proximity region.

See also (Ceyhan (2005, 2010)).

Usage

```
NPEstd.tetra(p, r, rv = NULL)
```

Arguments

`p`	A 3D point whose PE proximity region is to be computed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`rv`	Index of the vertex region containing the point, either `1,2,3,4` or `NULL` (default is `NULL`).

Value

Vertices of the tetrahedron which constitutes the PE proximity region with expansion parameter `r` and circumcenter (or center of mass) for a point `p` in the standard regular tetrahedron

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

NPEtetra, NPEtri and NPEint

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-3
Xp<-runif.std.tetra(n)$g
r<-1.5
NPEstd.tetra(Xp[1,],r)

#or try
RV<-rel.vert.tetraCC(Xp[1,],tetra)$rv
NPEstd.tetra(Xp[1,],r,rv=RV)

NPEstd.tetra(c(-1,-1,-1),r,rv=NULL)
```

`NPEtetra` *The vertices of the Proportional Edge (PE) Proximity Region in a tetrahedron*

Description

Returns the vertices of the PE proximity region (which is itself a tetrahedron) for a point in the tetrahedron `th`.

PE proximity region is defined with respect to the tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions based on the center `M` which is circumcenter (`"CC"`) or center of mass (`"CM"`) of `th` with default=`"CM"`.

Vertex regions are labeled as `1,2,3,4` rowwise for the vertices of the tetrahedron `th`. `rv` is the index of the vertex region `p` resides, with default=`NULL`. If `p` is outside of `th`, it returns `NULL` for the proximity region.

See also (Ceyhan (2005, 2010)).

Usage

```
NPEtetra(p, th, r, M = "CM", rv = NULL)
```

Arguments

`p`	A 3D point whose PE proximity region is to be computed.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.
`rv`	Index of the vertex region containing the point, either `1,2,3,4` (default is `NULL`).

Value

Vertices of the tetrahedron which constitutes the PE proximity region with expansion parameter `r` and circumcenter (or center of mass) for a point `p` in the tetrahedron

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`NPEstd.tetra`, `NPEtri` and `NPEint`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
set.seed(1)
tetra<-rbind(A,B,C,D)+matrix(runif(12,-.25,.25),ncol=3)
n<-3  #try also n<-20

Xp<-runif.tetra(n,tetra)$g

M<-"CM"  #try also M<-"CC"
r<-1.5

NPEtetra(Xp[1,],tetra,r)  #uses the default M="CM"
NPEtetra(Xp[1,],tetra,r,M="CC")

#or try
RV<-rel.vert.tetraCM(Xp[1,],tetra)$rv
NPEtetra(Xp[1,],tetra,r,M,rv=RV)

P1<-c(.1,.1,.1)
NPEtetra(P1,tetra,r,M)
```

`NPEtri` *The vertices of the Proportional Edge (PE) Proximity Region in a general triangle*

Description

Returns the vertices of the PE proximity region (which is itself a triangle) for a point in the triangle `tri`$= T(A,B,C) =$`(rv=1,rv=2,rv=3)`.

PE proximity region is defined with respect to the triangle `tri` with expansion parameter $r \geq 1$ and vertex regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1,1,1)$, i.e., the center of mass of `tri`.

Vertex regions are labeled as $1, 2, 3$ rowwise for the vertices of the triangle `tri`. `rv` is the index of the vertex region `p` resides, with default=`NULL`. If `p` is outside of `tri`, it returns `NULL` for the proximity region.

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
NPEtri(p, tri, r, M = c(1, 1, 1), rv = NULL)
```

Arguments

`p`	A 2D point whose PE proximity region is to be computed.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1,1,1)$, i.e., the center of mass of `tri`.
`rv`	Index of the `M`-vertex region containing the point `p`, either `1,2,3` or `NULL` (default is `NULL`).

Value

Vertices of the triangular region which constitutes the PE proximity region with expansion parameter `r` and center `M` for a point `p`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`NPEbasic.tri`, `NAStri`, `NCStri`, and `IarcPEtri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5

n<-3
set.seed(1)
Xp<-runif.tri(n,Tr)$g

NPEtri(Xp[3,],Tr,r,M)

P1<-as.numeric(runif.tri(1,Tr)$g)  #try also P1<-c(.4,.2)
NPEtri(P1,Tr,r,M)

M<-c(1.3,1.3)
r<-2

P1<-c(1.4,1.2)
P2<-c(1.5,1.26)
NPEtri(P1,Tr,r,M)
NPEtri(P2,Tr,r,M)

#or try
Rv<-rel.vert.tri(P1,Tr,M)$rv
NPEtri(P1,Tr,r,M,Rv)
```

`num.arcsAS` *Number of arcs of Arc Slice Proximity Catch Digraphs (AS-PCDs) and related quantities of the induced subdigraphs for points in the Delaunay triangles - multiple triangle case*

Description

An object of class `"NumArcs"`. Returns the number of arcs and various other quantities related to the Delaunay triangles for Arc Slice Proximity Catch Digraph (AS-PCD) whose vertices are the data points in `Xp` in the multiple triangle case (with triangulation based on `Yp` points).

AS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points and vertex regions in each triangle are based on the center `M="CC"` for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is `M="CC"` i.e., circumcenter of each triangle.

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005, 2010)) for more on AS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
num.arcsAS(Xp, Yp, M = "CC")
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`M`	The center of the triangle. `"CC"` stands for circumcenter of each Delaunay triangle or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is `M="CC"` i.e., the circumcenter of each triangle.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and related quantities for the induced subdigraphs in the Delaunay triangles
`num.arcs`	Total number of arcs in all triangles, i.e., the number of arcs for the entire AS-PCD
`num.in.conv.hull`	Number of `Xp` points in the convex hull of `Yp` points
`num.in.tris`	The vector of number of `Xp` points in the Delaunay triangles based on `Yp` points

`weight.vec`	The `vector` of the areas of Delaunay triangles based on `Yp` points
`tri.num.arcs`	The `vector` of the number of arcs of the components of the AS-PCD in the Delaunay triangles based on `Yp` points
`del.tri.ind`	A matrix of indices of Delaunay triangles based on `Yp` points, each column corresponds to the vector of indices of the vertices of one of the Delaunay triangle.
`data.tri.ind`	A `vector` of indices of vertices of the Delaunay triangles in which data points reside, i.e., column number of `del.tri.ind` for each `Xp` point.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the Delaunay triangulation based on `Yp` points.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`num.arcsAStri`, `num.arcsPE`, and `num.arcsCS`

Examples

```
nx<-15; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-"CC"  #try also M<-c(1,1,1)
Narcs = num.arcsAS(Xp,Yp,M)
Narcs
```

```
summary(Narcs)
plot(Narcs)
```

---

num.arcsAStri *Number of arcs of Arc Slice Proximity Catch Digraphs (AS-PCDs) and quantities related to the triangle - one triangle case*

Description

An object of class `"NumArcs"`. Returns the number of arcs of Arc Slice Proximity Catch Digraphs (AS-PCDs) whose vertices are the 2D data set, `Xp`. It also provides number of vertices (i.e., number of data points inside the triangle) and indices of the data points that reside in the triangle.

The data points could be inside or outside a general triangle `tri`$= T(A,B,C) =$`(rv=1,rv=2,rv=3)`, with vertices of `tri` stacked row-wise.

AS proximity regions are defined with respect to the triangle `tri` and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`. For the number of arcs, loops are not allowed, so arcs are only possible for points inside the triangle, `tri`.

See also (Ceyhan (2005, 2010)).

Usage

```
num.arcsAStri(Xp, tri, M = "CC")
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the digraph (i.e., AS-PCD).
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M="CC"` i.e., the circumcenter of `tri`.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and quantities related to the triangle
`num.arcs`	Number of arcs of the AS-PCD
`tri.num.arcs`	Number of arcs of the induced subdigraph of the AS-PCD for vertices in the triangle `tri`

`num.in.tri`	Number of `Xp` points in the triangle, `tri`
`ind.in.tri`	The vector of indices of the `Xp` points that reside in the triangle
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support triangle `tri`.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`num.arcsAS`, `num.arcsPEtri`, and `num.arcsCStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-10  #try also n<-20
set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

Narcs = num.arcsAStri(Xp,Tr,M)
Narcs
summary(Narcs)
plot(Narcs)
```

`num.arcsCS` *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) and related quantities of the induced subdigraphs for points in the Delaunay triangles - multiple triangle case*

Description

An object of class `"NumArcs"`. Returns the number of arcs and various other quantities related to the Delaunay triangles for Central Similarity Proximity Catch Digraph (CS-PCD) whose vertices are the data points in `Xp` in the multiple triangle case.

CS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $t > 0$ and edge regions in each triangle is based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (nonscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) for more on CS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
num.arcsCS(Xp, Yp, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle, default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and related quantities for the induced subdigraphs in the Delaunay triangles
`num.arcs`	Total number of arcs in all triangles, i.e., the number of arcs for the entire CS-PCD

`num.in.conv.hull`	Number of `Xp` points in the convex hull of `Yp` points
`num.in.tris`	The vector of number of `Xp` points in the Delaunay triangles based on `Yp` points
`weight.vec`	The `vector` of the areas of Delaunay triangles based on `Yp` points
`tri.num.arcs`	The `vector` of the number of arcs of the components of the CS-PCD in the Delaunay triangles based on `Yp` points
`del.tri.ind`	A matrix of indices of vertices of the Delaunay triangles based on `Yp` points, each column corresponds to the vector of indices of the vertices of one triangle.
`data.tri.ind`	A `vector` of indices of vertices of the Delaunay triangles in which data points reside, i.e., column number of `del.tri.ind` for each `Xp` point.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the Delaunay triangulation based on `Yp` points.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2014). “Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering.” *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`num.arcsCStri`, `num.arcsCSstd.tri`, `num.arcsPE`, and `num.arcsAS`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
```

```
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

Narcs = num.arcsCS(Xp,Yp,t=1,M)
Narcs
summary(Narcs)
plot(Narcs)
```

`num.arcsCS1D` *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) and related quantities of the induced subdigraphs for points in the partition intervals - multiple interval case*

Description

An object of class "NumArcs". Returns the number of arcs and various other quantities related to the partition intervals for Central Similarity Proximity Catch Digraph (CS-PCD) whose vertices are the data points in Xp in the multiple interval case.

For this function, CS proximity regions are constructed data points inside or outside the intervals based on Yp points with expansion parameter $t \geq 0$ and centrality parameter $c \in (0, 1)$. That is, for this function, arcs may exist for points in the middle or end-intervals.

Range (or convex hull) of Yp (i.e., the interval $(\min(Yp), \max(Yp))$) is partitioned by the spacings based on Yp points (i.e., multiple intervals are these partition intervals based on the order statistics of Yp points whose union constitutes the range of Yp points). If there are duplicates of Yp points, only one point is retained for each duplicate value, and a warning message is printed. For the number of arcs, loops are not counted.

Usage

```
num.arcsCS1D(Xp, Yp, t, c = 0.5)
```

Arguments

Xp	A set or `vector` of 1D points which constitute the vertices of the CS-PCD.
Yp	A set or `vector` of 1D points which constitute the end points of the partition intervals.
t	A positive real number which serves as the expansion parameter in CS proximity region; must be > 0.
c	A positive real number in $(0, 1)$ parameterizing the center inside the middle (partition) intervals with the default `c=.5`. For an interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and related quantities for the induced subdigraphs in the partition intervals
`num.arcs`	Total number of arcs in all intervals (including the end-intervals), i.e., the number of arcs for the entire CS-PCD
`num.in.range`	Number of `Xp` points in the range or convex hull of `Yp` points
`num.in.ints`	The vector of number of `Xp` points in the partition intervals (including the end-intervals) based on `Yp` points
`weight.vec`	The `vector` of the lengths of the middle partition intervals (i.e., end-intervals excluded) based on `Yp` points
`int.num.arcs`	The `vector` of the number of arcs of the components of the CS-PCD in the partition intervals (including the end-intervals) based on `Yp` points
`part.int`	A list of partition intervals based on `Yp` points
`data.int.ind`	A `vector` of indices of partition intervals in which data points reside, i.e., column number of `part.int` is provided for each `Xp` point. Partition intervals are numbered from left to right with 1 being the left end-interval.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the partition intervals based on `Yp` points.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

`num.arcsCSint`, `num.arcsCSmid.int`, `num.arcsCSend.int`, and `num.arcsPE1D`

Examples

```
tau<-1.5
c<-.4
a<-0; b<-10; int<-c(a,b);

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(nx,a-xf,b+xf)
```

```
Yp<-runif(ny,a,b)

Narcs = num.arcsCS1D(Xp,Yp,tau,c)
Narcs
summary(Narcs)
plot(Narcs)
```

`num.arcsCSend.int` *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) - end-interval case*

Description

Returns the number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are a 1D numerical data set, `Xp`, outside the interval `int`= (a, b).

CS proximity region is constructed only with expansion parameter $t > 0$ for points outside the interval (a, b).

End vertex regions are based on the end points of the interval, i.e., the corresponding end vertex region is an interval as $(-\infty, a)$ or (b, ∞) for the interval (a, b). For the number of arcs, loops are not allowed, so arcs are only possible for points outside the interval, `int`, for this function.

See also (Ceyhan (2016)).

Usage

```
num.arcsCSend.int(Xp, int, t)
```

Arguments

`Xp`	A `vector` of 1D points which constitute the vertices of the digraph.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.

Value

Number of arcs for the CS-PCD with vertices being 1D data set, `Xp`, expansion parameter, `t`, for the end-intervals.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

num.arcsCSmid.int, num.arcsPEmid.int, and num.arcsPEend.int

Examples

```
a<-0; b<-10; int<-c(a,b)

n<-5
XpL<-runif(n,a-5,a)
XpR<-runif(n,b,b+5)
Xp<-c(XpL,XpR)

num.arcsCSend.int(Xp,int,t=2)

num.arcsCSend.int(Xp,int,t=1.2)

num.arcsCSend.int(Xp,int,t=4)

num.arcsCSend.int(Xp,int,t=2+5)
#num.arcsCSend.int(Xp,int,t=c(-5,15))

n<-10  #try also n<-20
Xp2<-runif(n,a-5,b+5)
num.arcsCSend.int(Xp2,int,t=2)

t<-.5
num.arcsCSend.int(Xp,int,t)
```

`num.arcsCSint` *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) and quantities related to the interval - one interval case*

Description

An object of class `"NumArcs"`. Returns the number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are the data points in `Xp` in the one middle interval case. It also provides number of vertices (i.e., number of data points inside the intervals) and indices of the data points that reside in the intervals.

The data points could be inside or outside the interval is `int`$= (a, b)$.

CS proximity region is constructed with an expansion parameter $t > 0$ and a centrality parameter $c \in (0, 1)$. CS proximity region is constructed for both points inside and outside the interval, hence the arcs may exist for all points inside or outside the interval.

See also (Ceyhan (2016)).

Usage

```
num.arcsCSint(Xp, int, t, c = 0.5)
```

Arguments

`Xp`	A set of 1D points which constitute the vertices of CS-PCD.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$ with the default `c=.5`. For the interval, `int`$=(a,b)$, the parameterized center is $M_c = a + c(b-a)$.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and quantities related to the interval
`num.arcs`	Total number of arcs in all intervals (including the end-intervals), i.e., the number of arcs for the entire CS-PCD
`num.in.range`	Number of `Xp` points in the interval `int`
`num.in.ints`	The vector of number of `Xp` points in the partition intervals (including the end-intervals)
`int.num.arcs`	The `vector` of the number of arcs of the components of the CS-PCD in the partition intervals (including the end-intervals)
`data.int.ind`	A `vector` of indices of partition intervals in which data points reside. Partition intervals are numbered from left to right with 1 being the left end-interval.
`ind.left.end`, `ind.mid`, `ind.right.end`	Indices of data points in the left end-interval, middle interval, and right end-interval (respectively)
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the end points of the support interval `int`.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`num.arcsCSmid.int`, `num.arcsCSend.int`, and `num.arcsPEint`

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

n<-10
set.seed(1)
Xp<-runif(n,a,b)
Narcs = num.arcsCSint(Xp,int,t,c)
Narcs
summary(Narcs)
plot(Narcs)
```

`num.arcsCSmid.int` *Number of Arcs of of Central Similarity Proximity Catch Digraphs (CS-PCDs) - middle interval case*

Description

Returns the number of arcs of of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are the given 1D numerical data set, `Xp`.

CS proximity region $N_{CS}(x, t, c)$ is defined with respect to the interval `int`$= (a, b)$ for this function. CS proximity region is constructed with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$.

Vertex regions are based on the center associated with the centrality parameter $c \in (0, 1)$. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$ and for the number of arcs, loops are not allowed so arcs are only possible for points inside the middle interval `int` for this function.

See also (Ceyhan (2016)).

Usage

```
num.arcsCSmid.int(Xp, int, t, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of CS-PCD.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

Number of arcs for the CS-PCD whose vertices are the 1D data set, `Xp`, with expansion parameter, $r \geq 1$, and centrality parameter, $c \in (0,1)$. PE proximity regions are defined only for `Xp` points inside the interval `int`, i.e., arcs are possible for such points only.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`num.arcsCSend.int`, `num.arcsPEmid.int`, and `num.arcsPEend.int`

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

n<-10
Xp<-runif(n,a,b)
num.arcsCSmid.int(Xp,int,t,c)

num.arcsCSmid.int(Xp,int,t,c=.3)

num.arcsCSmid.int(Xp,int,t=1.5,c)

#num.arcsCSmid.int(Xp,int,t,c+5) #gives error
#num.arcsCSmid.int(Xp,int,t,c+10)

n<-10  #try also n<-20
Xp<-runif(n,a-5,b+5)
num.arcsCSint(Xp,int,t,c)

Xp<-runif(n,a+10,b+10)
num.arcsCSmid.int(Xp,int,t,c)

n<-10
Xp<-runif(n,a,b)
num.arcsCSmid.int(Xp,int,t,c)
```

`num.arcsCSstd.tri` *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) and quantities related to the triangle - standard equilateral triangle case*

Description

An object of class `"NumArcs"`. Returns the number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are the given 2D numerical data set, `Xp`. It also provides number of vertices (i.e., number of data points inside the standard equilateral triangle T_e) and indices of the data points that reside in T_e.

CS proximity region $N_{CS}(x, t)$ is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e., the center of mass of T_e. For the number of arcs, loops are not allowed so arcs are only possible for points inside T_e for this function.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
num.arcsCSstd.tri(Xp, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the digraph.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates. which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and quantities related to the standard equilateral triangle
`num.arcs`	Number of arcs of the CS-PCD
`tri.num.arcs`	Number of arcs of the induced subdigraph of the CS-PCD for vertices in the standard equilateral triangle T_e
`num.in.tri`	Number of `Xp` points in the standard equilateral triangle, T_e
`ind.in.tri`	The vector of indices of the `Xp` points that reside in T_e
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support triangle T_e.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`num.arcsCStri`, `num.arcsCS`, and `num.arcsPEstd.tri`,

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

Narcs = num.arcsCSstd.tri(Xp,t=.5,M)
Narcs
summary(Narcs)
oldpar <- par(pty="s")
plot(Narcs,asp=1)
par(oldpar)
```

---

num.arcsCStri   *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) and quantities related to the triangle - one triangle case*

Description

An object of class "NumArcs". Returns the number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are the given 2D numerical data set, `Xp`. It also provides number of vertices (i.e., number of data points inside the triangle) and indices of the data points that reside in the triangle.

CS proximity region $N_{CS}(x,t)$ is defined with respect to the triangle, `tri` with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`. For the number of arcs, loops are not allowed so arcs are only possible for points inside `tri` for this function.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
num.arcsCStri(Xp, tri, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of CS-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and quantities related to the triangle
`num.arcs`	Number of arcs of the CS-PCD
`tri.num.arcs`	Number of arcs of the induced subdigraph of the CS-PCD for vertices in the triangle `tri`
`num.in.tri`	Number of `Xp` points in the triangle, `tri`
`ind.in.tri`	The vector of indices of the `Xp` points that reside in the triangle
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support triangle `tri`.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`num.arcsCSstd.tri`, `num.arcsCS`, `num.arcsPEtri`, and `num.arcsAStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-10  #try also n<-20
set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

Narcs = num.arcsCStri(Xp,Tr,t=.5,M)
Narcs
summary(Narcs)
plot(Narcs)
```

---

num.arcsPE *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) and related quantities of the induced subdigraphs for points in the Delaunay triangles - multiple triangle case*

Description

An object of class `"NumArcs"`. Returns the number of arcs and various other quantities related to the Delaunay triangles for Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in `Xp` in the multiple triangle case.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle is based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (nonscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2006)) for more on PE-PCDs. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
num.arcsPE(Xp, Yp, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this, argument should be set as `M="CC"`), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and related quantities for the induced subdigraphs in the Delaunay triangles
`num.arcs`	Total number of arcs in all triangles, i.e., the number of arcs for the entire PE-PCD
`num.in.conv.hull`	Number of `Xp` points in the convex hull of `Yp` points
`num.in.tris`	The vector of number of `Xp` points in the Delaunay triangles based on `Yp` points
`weight.vec`	The `vector` of the areas of Delaunay triangles based on `Yp` points
`tri.num.arcs`	The `vector` of the number of arcs of the components of the PE-PCD in the Delaunay triangles based on `Yp` points
`del.tri.ind`	A matrix of indices of vertices of the Delaunay triangles based on `Yp` points, each column corresponds to the vector of indices of the vertices of one triangle.
`data.tri.ind`	A `vector` of indices of vertices of the Delaunay triangles in which data points reside, i.e., column number of `del.tri.ind` for each `Xp` point.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the Delaunay triangulation based on `Yp` points.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`num.arcsPEtri`, `num.arcsPEstd.tri`, `num.arcsCS`, and `num.arcsAS`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

Narcs = num.arcsPE(Xp,Yp,r=1.25,M)
Narcs
summary(Narcs)
plot(Narcs)
```

`num.arcsPE1D` *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) and related quantities of the induced subdigraphs for points in the partition intervals - multiple interval case*

Description

An object of class `"NumArcs"`. Returns the number of arcs and various other quantities related to the partition intervals for Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in `Xp` in the multiple interval case.

For this function, PE proximity regions are constructed data points inside or outside the intervals based on `Yp` points with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$. That is, for this function, arcs may exist for points in the middle or end-intervals.

Range (or convex hull) of `Yp` (i.e., the interval $(\min(Yp), \max(Yp))$) is partitioned by the spacings based on `Yp` points (i.e., multiple intervals are these partition intervals based on the order statistics of `Yp` points whose union constitutes the range of `Yp` points). If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed. For the number of arcs, loops are not counted.

See also (Ceyhan (2012)).

Usage

```
num.arcsPE1D(Xp, Yp, r, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set or `vector` of 1D points which constitute the end points of the partition intervals.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside the middle (partition) intervals with the default `c=.5`. For an interval, (a, b), the parameterized center is $M_c = a + c(b - a)$.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and related quantities for the induced subdigraphs in the partition intervals
`num.arcs`	Total number of arcs in all intervals (including the end-intervals), i.e., the number of arcs for the entire PE-PCD
`num.in.range`	Number of `Xp` points in the range or convex hull of `Yp` points
`num.in.ints`	The vector of number of `Xp` points in the partition intervals (including the end-intervals) based on `Yp` points
`weight.vec`	The `vector` of the lengths of the middle partition intervals (i.e., end-intervals excluded) based on `Yp` points
`int.num.arcs`	The `vector` of the number of arcs of the components of the PE-PCD in the partition intervals (including the end-intervals) based on `Yp` points
`part.int`	A matrix with columns corresponding to the partition intervals based on `Yp` points.

`data.int.ind`	A `vector` of indices of partition intervals in which data points reside, i.e., column number of `part.int` is provided for each `Xp` point. Partition intervals are numbered from left to right with 1 being the left end-interval.
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation is the partition intervals based on `Yp` points.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`num.arcsPEint`, `num.arcsPEmid.int`, `num.arcsPEend.int`, and `num.arcsCS1D`

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b);

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Narcs = num.arcsPE1D(Xp,Yp,r,c)
Narcs
summary(Narcs)
plot(Narcs)
```

`num.arcsPEend.int` *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - end-interval case*

Description

Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are a 1D numerical data set, `Xp`, outside the interval `int`$= (a, b)$.

PE proximity region is constructed only with expansion parameter $r \geq 1$ for points outside the interval (a, b). End vertex regions are based on the end points of the interval, i.e., the corresponding vertex region is an interval as $(-\infty, a)$ or (b, ∞) for the interval (a, b). For the number of arcs, loops are not allowed, so arcs are only possible for points outside the interval, `int`, for this function.

See also (Ceyhan (2012)).

Usage

```
num.arcsPEend.int(Xp, int, r)
```

Arguments

`Xp`	A `vector` of 1D points which constitute the vertices of the digraph.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

Value

Number of arcs for the PE-PCD with vertices being 1D data set, `Xp`, expansion parameter, $r \geq 1$, for the end-intervals.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`num.arcsPEmid.int`, `num.arcsPE1D`, `num.arcsCSmid.int`, and `num.arcsCSend.int`

Examples

```
a<-0; b<-10; int<-c(a,b)

n<-5
XpL<-runif(n,a-5,a)
XpR<-runif(n,b,b+5)
Xp<-c(XpL,XpR)

r<-1.2
num.arcsPEend.int(Xp,int,r)
```

```
num.arcsPEend.int(Xp,int,r=2)
```

---

num.arcsPEint *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) and quantities related to the interval - one interval case*

Description

An object of class `"NumArcs"`. Returns the number of arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in `Xp` in the one middle interval case. It also provides number of vertices (i.e., number of data points inside the intervals) and indices of the data points that reside in the intervals.

The data points could be inside or outside the interval is `int`$=(a,b)$. PE proximity region is constructed with an expansion parameter $r \geq 1$ and a centrality parameter $c \in (0,1)$. `int` determines the end points of the interval.

The PE proximity region is constructed for both points inside and outside the interval, hence the arcs may exist for all points inside or outside the interval.

See also (Ceyhan (2012)).

Usage

```
num.arcsPEint(Xp, int, r, c = 0.5)
```

Arguments

`Xp`	A set of 1D points which constitute the vertices of PE-PCD.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$ with the default `c=.5`. For the interval, `int`$=(a,b)$, the parameterized center is $M_c = a + c(b-a)$.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and quantities related to the interval
`num.arcs`	Total number of arcs in all intervals (including the end-intervals), i.e., the number of arcs for the entire PE-PCD
`num.in.range`	Number of `Xp` points in the interval `int`

`num.in.ints`	The vector of number of `Xp` points in the partition intervals (including the end-intervals)
`int.num.arcs`	The `vector` of the number of arcs of the components of the PE-PCD in the partition intervals (including the end-intervals)
`data.int.ind`	A `vector` of indices of partition intervals in which data points reside. Partition intervals are numbered from left to right with 1 being the left end-interval.
`ind.left.end`, `ind.mid`, `ind.right.end`	Indices of data points in the left end-interval, middle interval, and right end-interval (respectively)
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the end points of the support interval `int`.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`num.arcsPEmid.int`, `num.arcsPEend.int`, and `num.arcsCSint`

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

xf<-(int[2]-int[1])*.1

set.seed(123)

n<-10
Xp<-runif(n,a-xf,b+xf)
Narcs = num.arcsPEint(Xp,int,r,c)
Narcs
summary(Narcs)
plot(Narcs)
```

`num.arcsPEmid.int` *Number of Arcs for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - middle interval case*

Description

Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are the given 1D numerical data set, `Xp`. PE proximity region $N_{PE}(x, r, c)$ is defined with respect to the interval `int`$= (a, b)$ for this function.

PE proximity region is constructed with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$.

Vertex regions are based on the center associated with the centrality parameter $c \in (0, 1)$. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$ and for the number of arcs, loops are not allowed so arcs are only possible for points inside the middle interval `int` for this function.

See also (Ceyhan (2012)).

Usage

```
num.arcsPEmid.int(Xp, int, r, c = 0.5)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of PE-PCD.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int`$= (a, b)$ with the default `c=.5`. For the interval, `int`$= (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

Number of arcs for the PE-PCD whose vertices are the 1D data set, `Xp`, with expansion parameter, $r \geq 1$, and centrality parameter, $c \in (0, 1)$. PE proximity regions are defined only for `Xp` points inside the interval `int`, i.e., arcs are possible for such points only.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

num.arcsPEend.int, num.arcsPE1D, num.arcsCSmid.int, and num.arcsCSend.int

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

n<-10
Xp<-runif(n,a,b)
num.arcsPEmid.int(Xp,int,r,c)
num.arcsPEmid.int(Xp,int,r=1.5,c)
```

---

num.arcsPEstd.tri *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) and quantities related to the triangle - standard equilateral triangle case*

Description

An object of class `"NumArcs"`. Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are the given 2D numerical data set, `Xp` in the standard equilateral triangle. It also provides number of vertices (i.e., number of data points inside the standard equilateral triangle T_e) and indices of the data points that reside in T_e.

PE proximity region $N_{PE}(x, r)$ is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$, i.e., the center of mass of T_e. For the number of arcs, loops are not allowed so arcs are only possible for points inside T_e for this function.

See also (Ceyhan et al. (2006)).

Usage

```
num.arcsPEstd.tri(Xp, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter for PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and quantities related to the standard equilateral triangle
`num.arcs`	Number of arcs of the PE-PCD
`tri.num.arcs`	Number of arcs of the induced subdigraph of the PE-PCD for vertices in the standard equilateral triangle T_e
`num.in.tri`	Number of `Xp` points in the standard equilateral triangle, T_e
`ind.in.tri`	The vector of indices of the `Xp` points that reside in T_e
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support triangle T_e.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`num.arcsPEtri`, `num.arcsPE`, and `num.arcsCSstd.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-c(.6,.2)  #try also M<-c(1,1,1)

Narcs = num.arcsPEstd.tri(Xp,r=1.25,M)
Narcs
summary(Narcs)
oldpar <- par(pty="s")
plot(Narcs,asp=1)
par(oldpar)
```

`num.arcsPEtetra`	*Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) and quantities related to the tetrahedron - one tetrahedron case*

Description

An object of class `"NumArcs"`. Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are the given 3D numerical data set, `Xp`. It also provides number of vertices (i.e., number of data points inside the tetrahedron) and indices of the data points that reside in the tetrahedron.

PE proximity region is constructed with respect to the tetrahedron `th` and vertex regions are based on the center `M` which is circumcenter (`"CC"`) or center of mass (`"CM"`) of `th` with default=`"CM"`. For the number of arcs, loops are not allowed so arcs are only possible for points inside the tetrahedron `th` for this function.

See also (Ceyhan (2005, 2010)).

Usage

```
num.arcsPEtetra(Xp, th, r, M = "CM")
```

Arguments

`Xp`	A set of 3D points which constitute the vertices of PE-PCD.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and quantities related to the tetrahedron
`num.arcs`	Number of arcs of the PE-PCD
`tri.num.arcs`	Number of arcs of the induced subdigraph of the PE-PCD for vertices in the tetrahedron `th`
`num.in.tetra`	Number of `Xp` points in the tetrahedron, `th`
`ind.in.tetra`	The vector of indices of the `Xp` points that reside in the tetrahedron
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support tetrahedron `th`.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`num.arcsPEtri`, `num.arcsCStri`, and `num.arcsAStri`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-10  #try also n<-20
set.seed(1)
Xp<-runif.tetra(n,tetra)$g

M<-"CM"  #try also M<-"CC"
r<-1.25

Narcs = num.arcsPEtetra(Xp,tetra,r,M)
Narcs
summary(Narcs)
#plot(Narcs)
```

`num.arcsPEtri` *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) and quantities related to the triangle - one triangle case*

Description

An object of class `"NumArcs"`. Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are the given 2D numerical data set, `Xp`. It also provides number of vertices (i.e., number of data points inside the triangle) and indices of the data points that reside in the triangle.

PE proximity region $N_{PE}(x,r)$ is defined with respect to the triangle, `tri` with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or

$M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`. For the number of arcs, loops are not allowed so arcs are only possible for points inside the triangle `tri` for this function.

See also (Ceyhan (2005, 2016)).

Usage

```
num.arcsPEtri(Xp, tri, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of PE-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.

Value

A `list` with the elements

`desc`	A short description of the output: number of arcs and quantities related to the triangle
`num.arcs`	Number of arcs of the PE-PCD
`tri.num.arcs`	Number of arcs of the induced subdigraph of the PE-PCD for vertices in the triangle `tri`
`num.in.tri`	Number of `Xp` points in the triangle, `tri`
`ind.in.tri`	The vector of indices of the `Xp` points that reside in the triangle
`tess.points`	Tessellation points, i.e., points on which the tessellation of the study region is performed, here, tessellation points are the vertices of the support triangle `tri`.
`vertices`	Vertices of the digraph, `Xp`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2016). "Edge Density of New Graph Types Based on a Random Digraph Family." *Statistical Methodology*, **33**, 31-54.

See Also

`num.arcsPEstd.tri`, `num.arcsPE`, `num.arcsCStri`, and `num.arcsAStri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-10  #try also n<-20
set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

Narcs = num.arcsPEtri(Xp,Tr,r=1.25,M)
Narcs
summary(Narcs)
plot(Narcs)
```

`num.delaunay.tri` *Number of Delaunay triangles based on a 2D data set*

Description

Returns the number of Delaunay triangles based on the 2D set of points `Yp`. See (Okabe et al. (2000); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
num.delaunay.tri(Yp)
```

Arguments

`Yp`	A set of 2D points which constitute the vertices of Delaunay triangles.

Value

Number of Delaunay triangles based on `Yp` points.

Author(s)

Elvan Ceyhan

References

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`plotDelaunay.tri`

Examples

```
ny<-10

set.seed(1)
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

num.delaunay.tri(Yp)
```

`paraline` *The line at a point* `p` *parallel to the line segment joining two distinct 2D points* `a` *and* `b`

Description

An object of class `"Lines"`. Returns the `equation`, `slope`, `intercept`, and y-coordinates of the line crossing the point `p` and parallel to the line passing through the points `a` and `b` with x-coordinates are provided in `vector` `x`.

Usage

```
paraline(p, a, b, x)
```

Arguments

`p`	A 2D point at which the parallel line to line segment joining `a` and `b` crosses.
`a, b`	2D points that determine the line segment (the line will be parallel to this line segment).
`x`	A scalar or a `vector` of scalars representing the x-coordinates of the line parallel to `ab` and crossing `p`.

Value

A `list` with the elements

`desc`	Description of the line passing through point `p` and parallel to line segment joining `a` and `b`
`mtitle`	The `"main"` title for the plot of the line passing through point `p` and parallel to line segment joining `a` and `b`
`points`	The input points `p`, `a`, and `b` (stacked row-wise, i.e., point `p` is in row 1, point `a` is in row 2 and point `b` is in row 3). Line parallel to `ab` crosses `p`.
`x`	The input vector. It can be a scalar or a `vector` of scalars, which constitute the x-coordinates of the point(s) of interest on the line passing through point `p` and parallel to line segment joining `a` and `b`.
`y`	The output scalar or `vector` which constitutes the y-coordinates of the point(s) of interest on the line passing through point `p` and parallel to line segment joining `a` and `b`. If `x` is a scalar, then `y` will be a scalar and if `x` is a `vector` of scalars, then `y` will be a `vector` of scalars.
`slope`	Slope of the line, `Inf` is allowed, passing through point `p` and parallel to line segment joining `a` and `b`
`intercept`	Intercept of the line passing through point `p` and parallel to line segment joining `a` and `b`
`equation`	Equation of the line passing through point `p` and parallel to line segment joining `a` and `b`

Author(s)

Elvan Ceyhan

See Also

`slope`, `Line`, and `perpline`, `line` in the generic `stats` package, and `paraline3D`

Examples

```
A<-c(1.1,1.2); B<-c(2.3,3.4); p<-c(.51,2.5)

paraline(p,A,B,.45)

pts<-rbind(A,B,p)
xr<-range(pts[,1])
xf<-(xr[2]-xr[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100

plnAB<-paraline(p,A,B,x)
plnAB
summary(plnAB)
plot(plnAB)
```

```
y<-plnAB$y
Xlim<-range(x,pts[,1])
if (!is.na(y[1])) {Ylim<-range(y,pts[,2])} else {Ylim<-range(pts[,2])}
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
pf<-c(xd,-yd)*.025

plot(A,pch=".",xlab="",ylab="",main="Line Crossing P and Parallel to AB",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(pts)
txt.str<-c("A","B","p")
text(pts+rbind(pf,pf,pf),txt.str)

segments(A[1],A[2],B[1],B[2],lty=2)
if (!is.na(y[1])) {lines(x,y,type="l",lty=1,xlim=Xlim,ylim=Ylim)} else {abline(v=p[1])}
tx<-(A[1]+B[1])/2;
if (!is.na(y[1])) {ty<-paraline(p,A,B,tx)$y} else {ty=p[2]}
text(tx,ty,"line parallel to AB\n and crossing p")
```

`paraline3D` *The line crossing the 3D point* `p` *and parallel to line joining 3D points* `a` *and* `b`

Description

An object of class `"Lines3D"`. Returns the equation, x-, y-, and z-coordinates of the line crossing 3D point `p` and parallel to the line joining 3D points `a` and `b` (i.e., the line is in the direction of `vector` `b-a`) with the parameter `t` being provided in `vector` `t`.

Usage

```
paraline3D(p, a, b, t)
```

Arguments

`p`	A 3D point through which the straight line passes.
`a, b`	3D points which determine the straight line to which the line passing through point `p` would be parallel (i.e., $b-a$ determines the direction of the straight line passing through `p`).
`t`	A scalar or a `vector` of scalars representing the parameter of the coordinates of the line (for the form: $x = p_0 + At$, $y = y_0 + Bt$, and $z = z_0 + Ct$ where $p = (p_0, y_0, z_0)$ and $b - a = (A, B, C)$).

Value

A `list` with the elements

`desc`	A description of the line
`mtitle`	The `"main"` title for the plot of the line
`points`	The input points that determine the line to which the line crossing point `p` would be parallel.
`pnames`	The names of the input points that determine the line to which the line crossing point `p` would be parallel.
`vecs`	The points `p`, `a`, and `b` stacked row-wise in this order.
`vec.names`	The names of the points `p`, `a`, and `b`.
`x, y, z`	The x-, y-, and z-coordinates of the point(s) of interest on the line parallel to the line determined by points `a` and `b`.
`tsq`	The scalar or the `vector` of the parameter in defining each coordinate of the line for the form: $x = p_0 + At$, $y = y_0 + Bt$, and $z = z_0 + Ct$ where $p = (p_0, y_0, z_0)$ and $b - a = (A, B, C)$.
`equation`	Equation of the line passing through point `p` and parallel to the line joining points `a` and `b` (i.e., in the direction of the `vector` `b-a`). The line equation is in the form: $x = p_0 + At$, $y = y_0 + Bt$, and $z = z_0 + Ct$ where $p = (p_0, y_0, z_0)$ and $b - a = (A, B, C)$.

Author(s)

Elvan Ceyhan

See Also

`Line3D`, `perpline2plane`, and `paraline`

Examples

```
P<-c(1,10,4); Q<-c(1,1,3); R<-c(3,9,12)

vecs<-rbind(P,R-Q)
pts<-rbind(P,Q,R)
paraline3D(P,Q,R,.1)

tr<-range(pts,vecs);
tf<-(tr[2]-tr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=5)  #try also l=10, 20, or 100

pln3D<-paraline3D(P,Q,R,tsq)
pln3D
summary(pln3D)
plot(pln3D)

x<-pln3D$x
```

```
y<-pln3D$y
z<-pln3D$z

zr<-range(z)
zf<-(zr[2]-zr[1])*.2
Qv<-(R-Q)*tf*5

Xlim<-range(x,pts[,1])
Ylim<-range(y,pts[,2])
Zlim<-range(z,pts[,3])

xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

Dr<-P+min(tsq)*(R-Q)

plot3D::lines3D(x, y, z, phi = 0, bty = "g",
main="Line Crossing P \n in the direction of R-Q",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),
zlim=Zlim+zd*c(-.1,.1)+c(-zf,zf),
        pch = 20, cex = 2, ticktype = "detailed")
plot3D::arrows3D(Dr[1],Dr[2],Dr[3]+zf,Dr[1]+Qv[1],
Dr[2]+Qv[2],Dr[3]+zf+Qv[3], add=TRUE)
plot3D::points3D(pts[,1],pts[,2],pts[,3],add=TRUE)
plot3D::text3D(pts[,1],pts[,2],pts[,3],labels=c("P","Q","R"),add=TRUE)
plot3D::arrows3D(P[1],P[2],P[3]-2*zf,P[1],P[2],P[3],lty=2, add=TRUE)
plot3D::text3D(P[1],P[2],P[3]-2*zf,labels="initial point",add=TRUE)
plot3D::arrows3D(Dr[1]+Qv[1]/2,Dr[2]+Qv[2]/2,
Dr[3]+3*zf+Qv[3]/2,Dr[1]+Qv[1]/2,
Dr[2]+Qv[2]/2,Dr[3]+zf+Qv[3]/2,lty=2, add=TRUE)
plot3D::text3D(Dr[1]+Qv[1]/2,Dr[2]+Qv[2]/2,Dr[3]+3*zf+Qv[3]/2,
labels="direction vector",add=TRUE)
plot3D::text3D(Dr[1]+Qv[1]/2,Dr[2]+Qv[2]/2,
Dr[3]+zf+Qv[3]/2,labels="R-Q",add=TRUE)
```

`paraplane` *The plane at a point and parallel to the plane spanned by three distinct 3D points* `a`*,* `b`*, and* `c`

Description

An object of class `"Planes"`. Returns the equation and z-coordinates of the plane passing through point `p` and parallel to the plane spanned by three distinct 3D points `a`, `b`, and `c` with x- and y-coordinates are provided in vectors `x` and `y`, respectively.

Usage

```
paraplane(p, a, b, c, x, y)
```

Arguments

`p`	A 3D point which the plane parallel to the plane spanned by three distinct 3D points `a`, `b`, and `c` crosses.
`a, b, c`	3D points that determine the plane to which the plane crossing point `p` is parallel to.
`x, y`	Scalars or `vectors` of scalars representing the x- and y-coordinates of the plane parallel to the plane spanned by points `a`, `b`, and `c` and passing through point `p`.

Value

A `list` with the elements

`desc`	Description of the plane passing through point `p` and parallel to plane spanned by points `a`, `b` and `c`
`points`	The input points `a`, `b`, `c`, and `p`. Plane is parallel to the plane spanned by `a`, `b`, and `c` and passes through point `p` (stacked row-wise, i.e., row 1 is point `a`, row 2 is point `b`, row 3 is point `c`, and row 4 is point `p`).
`x, y`	The input vectors which constitutes the x- and y-coordinates of the point(s) of interest on the plane. `x` and `y` can be scalars or vectors of scalars.
`z`	The output `vector` which constitutes the z-coordinates of the point(s) of interest on the plane. If `x` and `y` are scalars, `z` will be a scalar and if `x` and `y` are vectors of scalars, then `z` needs to be a `matrix` of scalars, containing the z-coordinate for each pair of `x` and `y` values.
`coeff`	Coefficients of the plane (in the $z = Ax + By + C$ form).
`equation`	Equation of the plane in long form
`equation2`	Equation of the plane in short form, to be inserted on the plot

Author(s)

Elvan Ceyhan

See Also

`Plane`

Examples

```
Q<-c(1,10,3); R<-c(1,1,3); S<-c(3,9,12); P<-c(1,1,0)

pts<-rbind(Q,R,S,P)
paraplane(P,Q,R,S,.1,.2)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.25
#how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100
```

```
y<-seq(yr[1]-yf,yr[2]+yf,l=5)  #try also l=10, 20, or 100

plP2QRS<-paraplane(P,Q,R,S,x,y)
plP2QRS
summary(plP2QRS)
plot(plP2QRS,theta = 225, phi = 30, expand = 0.7, facets = FALSE, scale = TRUE)

paraplane(P,Q,R,Q+R,.1,.2)

z.grid<-plP2QRS$z

plQRS<-Plane(Q,R,S,x,y)
plQRS
pl.grid<-plQRS$z

zr<-max(z.grid)-min(z.grid)
Pts<-rbind(Q,R,S,P)+rbind(c(0,0,zr*.1),c(0,0,zr*.1),
c(0,0,zr*.1),c(0,0,zr*.1))
Mn.pts<-apply(Pts[1:3,],2,mean)

plot3D::persp3D(z = pl.grid, x = x, y = y, theta =225, phi = 30,
ticktype = "detailed",
main="Plane Crossing Points Q, R, S\n and Plane Passing P Parallel to it")
#plane spanned by points Q, R, S
plot3D::persp3D(z = z.grid, x = x, y = y,add=TRUE)
#plane parallel to the original plane and passing thru point \code{P}

plot3D::persp3D(z = z.grid, x = x, y = y, theta =225, phi = 30,
ticktype = "detailed",
main="Plane Crossing Point P \n and Parallel to the Plane Crossing Q, R, S")
#plane spanned by points Q, R, S
#add the defining points
plot3D::points3D(Pts[,1],Pts[,2],Pts[,3], add=TRUE)
plot3D::text3D(Pts[,1],Pts[,2],Pts[,3], c("Q","R","S","P"),add=TRUE)
plot3D::text3D(Mn.pts[1],Mn.pts[2],Mn.pts[3],plP2QRS$equation,add=TRUE)
plot3D::polygon3D(Pts[1:3,1],Pts[1:3,2],Pts[1:3,3], add=TRUE)
```

`Pdom.num2PE1Dasy` *The asymptotic probability of domination number $= 2$ for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - middle interval case*

Description

Returns the asymptotic $P(\text{domination number} \leq 1)$ for PE-PCD whose vertices are a uniform data set in a finite interval (a,b).

The PE proximity region $N_{PE}(x,r,c)$ is defined with respect to (a,b) with centrality parameter c in $(0,1)$ and expansion parameter $r = 1/\max(c, 1-c)$.

Usage

```
Pdom.num2PE1Dasy(c)
```

Arguments

`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$. For the interval, (a,b), the parameterized center is $M_c = a + c(b-a)$.

Value

The asymptotic $P(\text{domination number} \leq 1)$ for PE-PCD whose vertices are a uniform data set in a finite interval (a,b)

Author(s)

Elvan Ceyhan

See Also

`Pdom.num2PE1D` and `Pdom.num2PEtri`

Examples

```
c<-.5

Pdom.num2PE1Dasy(c)

Pdom.num2PE1Dasy(c=1/1.5)
Pdom.num2PE1D(r=1.5,c=1/1.5,n=10)
Pdom.num2PE1D(r=1.5,c=1/1.5,n=100)
```

`Pdom.num2PEtri` *Asymptotic probability that domination number of Proportional Edge Proximity Catch Digraphs (PE-PCDs) equals 2 where vertices of the digraph are uniform points in a triangle*

Description

Returns $P(\text{domination number} = 2)$ for PE-PCD for uniform data in a triangle, when the sample size n goes to infinity (i.e., asymptotic probability of domination number $= 2$).

PE proximity regions are constructed with respect to the triangle with the expansion parameter $r \geq 1$ and M-vertex regions where M is the vertex that renders the asymptotic distribution of the domination number non-degenerate for the given value of `r` in $(1, 1.5]$.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011)).

Usage

```
Pdom.num2PEtri(r)
```

Arguments

`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ to attain non-degenerate asymptotic distribution for the domination number.

Value

$P(\text{domination number}= 2)$ for PE-PCD for uniform data on an triangle as the sample size n goes to infinity

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`Pdom.num2PE1D`

Examples

```
Pdom.num2PEtri(r=1.5)
Pdom.num2PEtri(r=1.4999999999)

Pdom.num2PEtri(r=1.5) / Pdom.num2PEtri(r=1.4999999999)

rseq<-seq(1.01,1.49999999999,l=20)  #try also l=100
lrseq<-length(rseq)

pg2<-vector()
for (i in 1:lrseq)
{
  pg2<-c(pg2,Pdom.num2PEtri(rseq[i]))
}
```

```
plot(rseq, pg2,type="l",xlab="r",
ylab=expression(paste("P(", gamma, "=2)")),
     lty=1,xlim=range(rseq)+c(0,.01),ylim=c(0,1))
points(rbind(c(1.50,Pdom.num2PEtri(1.50))),pch=".",cex=3)
```

`PEarc.dens.test` *A test of segregation/association based on arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the convex hull of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points) and association (where `Xp` points cluster around `Yp` points) based on the normal approximation of the arc density of the PE-PCD for uniform 2D data.

The function yields the test statistic, p-value for the corresponding `alternative`, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the convex hull of `Yp` points, arc density of PE-PCD whose vertices are `Xp` points equals to its expected value under the uniform distribution and `alternative` could be two-sided, or left-sided (i.e., data is accumulated around the `Yp` points, or association) or right-sided (i.e., data is accumulated around the centers of the triangles, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and CM-vertex regions (i.e., the test is not available for a general center M at this version of the function).

Caveat: This test is currently a conditional test, where `Xp` points are assumed to be random, while `Yp` points are assumed to be fixed (i.e., the test is conditional on `Yp` points). Furthermore, the test is a large sample test when `Xp` points are substantially larger than `Yp` points, say at least 5 times more. This test is more appropriate when supports of `Xp` and `Yp` have a substantial overlap. Currently, the `Xp` points outside the convex hull of `Yp` points are handled with a convex hull correction factor, `ch.cor`, which is derived under the assumption of uniformity of `Xp` and `Yp` points in the study window, (see the description below and the function code.) However, in the special case of no `Xp` points in the convex hull of `Yp` points, arc density is taken to be 1, as this is clearly a case of segregation. Removing the conditioning and extending it to the case of non-concurring supports is an ongoing topic of research of the author of the package.

`ch.cor` is for convex hull correction (default is `"no convex hull correction"`, i.e., `ch.cor=FALSE`) which is recommended when both `Xp` and `Yp` have the same rectangular support.

See also (Ceyhan (2005); Ceyhan et al. (2006)) for more on the test based on the arc density of PE-PCDs.

Usage

```
PEarc.dens.test(
  Xp,
  Yp,
  r,
  ch.cor = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`ch.cor`	A logical argument for convex hull correction, default `ch.cor=FALSE`, recommended when both `Xp` and `Yp` have the same rectangular support.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the arc density of PE-PCD based on the 2D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`
`conf.int`	Confidence interval for the arc density at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., arc density
`null.value`	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`CSarc.dens.test` and `PEarc.dens.test1D`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-100; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

plotDelaunay.tri(Xp,Yp,xlab="",ylab="")

PEarc.dens.test(Xp,Yp,r=1.25)
PEarc.dens.test(Xp,Yp,r=1.25,ch=TRUE)
#since Y points are not uniform, convex hull correction is invalid here
```

`PEarc.dens.test.int` *A test of uniformity of 1D data in a given interval based on Proportional Edge Proximity Catch Digraph (PE-PCD)*

Description

An object of class `"htest"`. This is an `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of uniformity of 1D data in one interval based on the normal approximation of the arc density of the PE-PCD with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$.

The function yields the test statistic, p-value for the corresponding `alternative`, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

The null hypothesis is that data is uniform in a finite interval (i.e., arc density of PE-PCD equals to its expected value under uniform distribution) and `alternative` could be two-sided, or left-sided (i.e., data is accumulated around the end points) or right-sided (i.e., data is accumulated around the mid point or center M_c).

See also (Ceyhan (2012, 2016)).

Usage

```
PEarc.dens.test.int(
  Xp,
  int,
  r,
  c = 0.5,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set or `vector` of 1D points which constitute the vertices of PE-PCD.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$= (a,b)$ with the default `c=.5`. For the interval, `int`$= (a,b)$, the parameterized center is $M_c = a + c(b-a)$.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the arc density of PE-PCD based on the 1D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`
`conf.int`	Confidence interval for the arc density at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., arc density
`null.value`	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`CSarc.dens.test.int`

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

n<-100  #try also n<-20, 1000
Xp<-runif(n,a,b)

PEarc.dens.test.int(Xp,int,r,c)
PEarc.dens.test.int(Xp,int,r,c,alt="g")
PEarc.dens.test.int(Xp,int,r,c,alt="l")
```

`PEarc.dens.test1D` *A test of segregation/association based on arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the range (i.e., range) of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points) and association (where `Xp` points cluster around `Yp` points) based on the normal approximation of the arc density of the PE-PCD for uniform 1D data.

The function yields the test statistic, p-value for the corresponding `alternative`, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the range of `Yp` points, arc density of PE-PCD whose vertices are `Xp` points equals to its expected value under the uniform distribution and `alternative` could be two-sided, or left-sided (i.e., data is accumulated around the `Yp` points, or association) or right-sided (i.e., data is accumulated around the centers of the intervals, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and centrality parameter `c` which yields M-vertex regions. More precisely, for a middle interval $(y_{(i)}, y_{(i+1)})$, the center is

$M = y_{(i)} + c(y_{(i+1)} - y_{(i)})$ for the centrality parameter $c \in (0, 1)$. If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

Caveat: This test is currently a conditional test, where `Xp` points are assumed to be random, while `Yp` points are assumed to be fixed (i.e., the test is conditional on `Yp` points). Furthermore, the test is a large sample test when `Xp` points are substantially larger than `Yp` points, say at least 5 times more. This test is more appropriate when supports of `Xp` and `Yp` have a substantial overlap. Currently, the `Xp` points outside the range of `Yp` points are handled with a range correction (or end-interval correction) factor (see the description below and the function code.) However, in the special case of no `Xp` points in the range of `Yp` points, arc density is taken to be 1, as this is clearly a case of segregation. Removing the conditioning and extending it to the case of non-concurring supports is an ongoing line of research of the author of the package.

`end.int.cor` is for end-interval correction, (default is "no end-interval correction", i.e., `end.int.cor=FALSE`), recommended when both `Xp` and `Yp` have the same interval support.

See also (Ceyhan (2012)) for more on the uniformity test based on the arc density of PE-PCDs.

Usage

```
PEarc.dens.test1D(
  Xp,
  Yp,
  r,
  c = 0.5,
  support.int = NULL,
  end.int.cor = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 1D points which constitute the end points of the partition intervals.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number which serves as the centrality parameter in PE proximity region; must be in $(0, 1)$ (default `c=.5`).
`support.int`	Support interval (a, b) with $a < b$. Uniformity of `Xp` points in this interval is tested. Default is `NULL`.
`end.int.cor`	A logical argument for end-interval correction, default is `FALSE`, recommended when both `Xp` and `Yp` have the same interval support.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the arc density PE-PCD whose vertices are the 1D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`.
`conf.int`	Confidence interval for the arc density at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., arc density
`null.value`	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`PEarc.dens.test`, `PEdom.num.binom.test1D`, and `PEarc.dens.test.int`

Examples

```
r<-2
c<-.4
a<-0; b<-10; int=c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-100; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

PEarc.dens.test1D(Xp,Yp,r,c,int)
#try also PEarc.dens.test1D(Xp,Yp,r,c,int,alt="l") and PEarc.dens.test1D(Xp,Yp,r,c,int,alt="g")

PEarc.dens.test1D(Xp,Yp,r,c,int,end.int.cor = TRUE)
```

`PEarc.dens.tetra` *Arc density of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns the arc density of PE-PCD whose vertex set is the given 2D numerical data set, `Xp`, (some of its members are) in the tetrahedron `th`.

PE proximity region is constructed with respect to the tetrahedron `th` and vertex regions are based on the center `M` which is circumcenter (`"CC"`) or center of mass (`"CM"`) of `th` with default=`"CM"`. For the number of arcs, loops are not allowed so arcs are only possible for points inside the tetrahedron `th` for this function.

`th.cor` is a logical argument for tetrahedron correction (default is `TRUE`), if `TRUE`, only the points inside the tetrahedron are considered (i.e., digraph induced by these vertices are considered) in computing the arc density, otherwise all points are considered (for the number of vertices in the denominator of arc density).

See also (Ceyhan (2005, 2010)).

Usage

```
PEarc.dens.tetra(Xp, th, r, M = "CM", th.cor = FALSE)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.
`th.cor`	A logical argument for computing the arc density for only the points inside the tetrahedron, `th`. (default is `th.cor=FALSE`), i.e., if `th.cor=TRUE` only the induced digraph with the vertices inside `th` are considered in the computation of arc density.

Value

Arc density of PE-PCD whose vertices are the 2D numerical data set, `Xp`; PE proximity regions are defined with respect to the tetrahedron `th` and `M`-vertex regions

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`PEarc.dens.tri` and `num.arcsPEtetra`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.tetra(n,tetra)$g

M<-"CM"  #try also M<-"CC"
r<-1.5

num.arcsPEtetra(Xp,tetra,r,M)
PEarc.dens.tetra(Xp,tetra,r,M)
PEarc.dens.tetra(Xp,tetra,r,M,th.cor = FALSE)
```

`PEarc.dens.tri` *Arc density of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns the arc density of PE-PCD whose vertex set is the given 2D numerical data set, `Xp`, (some of its members are) in the triangle `tri`.

PE proximity regions is defined with respect to `tri` with expansion parameter $r \geq 1$ and vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`. The function also provides arc density standardized by the mean and asymptotic variance of the arc density of PE-PCD for uniform data in the triangle `tri` only when `M` is the center of mass. For the number of arcs, loops are not allowed.

`in.tri.only` is a logical argument (default is `FALSE`) for considering only the points inside the triangle or all the points as the vertices of the digraph. if `in.tri.only=TRUE`, arc density is computed only for the points inside the triangle (i.e., arc density of the subdigraph induced by the vertices in the triangle is computed), otherwise arc density of the entire digraph (i.e., digraph with all the vertices) is computed.

See also (Ceyhan (2005); Ceyhan et al. (2006)).

Usage

```
PEarc.dens.tri(Xp, tri, r, M = c(1, 1, 1), in.tri.only = FALSE)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.
`in.tri.only`	A logical argument (default is `in.tri.only=FALSE`) for computing the arc density for only the points inside the triangle, `tri`. That is, if `in.tri.only=TRUE` arc density of the induced subdigraph with the vertices inside `tri` is computed, otherwise otherwise arc density of the entire digraph (i.e., digraph with all the vertices) is computed.

Value

A `list` with the elements

`arc.dens`	Arc density of PE-PCD whose vertices are the 2D numerical data set, `Xp`; PE proximity regions are defined with respect to the triangle `tri` and `M`-vertex regions
`std.arc.dens`	Arc density standardized by the mean and asymptotic variance of the arc density of PE-PCD for uniform data in the triangle `tri`. This will only be returned, if `M` is the center of mass.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

ASarc.dens.tri, CSarc.dens.tri, and num.arcsPEtri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

num.arcsPEtri(Xp,Tr,r=1.5,M)
PEarc.dens.tri(Xp,Tr,r=1.5,M)
PEarc.dens.tri(Xp,Tr,r=1.5,M,in.tri.only = TRUE)
```

PEdom.num *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) - multiple triangle case*

Description

Returns the domination number, indices of a minimum dominating set of PE-PCD whose vertices are the data points in `Xp` in the multiple triangle case and domination numbers for the Delaunay triangles based on `Yp` points.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (nonscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). Loops are allowed for the domination number.

See (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)) for more on the domination number of PE-PCDs. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
PEdom.num(Xp, Yp, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this, argument should be set as `M="CC"`), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A `list` with three elements

`dom.num`	Domination number of the PE-PCD whose vertices are `Xp` points. PE proximity regions are constructed with respect to the Delaunay triangles based on the `Yp` points with expansion parameter $r \geq 1$.
`#`	
`mds`	A minimum dominating set of the PE-PCD whose vertices are `Xp` points
`ind.mds`	The vector of data indices of the minimum dominating set of the PE-PCD whose vertices are `Xp` points.
`tri.dom.nums`	The vector of domination numbers of the PE-PCD components for the Delaunay triangles.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`PEdom.num.tri`, `PEdom.num.tetra`, `dom.num.exact`, and `dom.num.greedy`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)
r<-1.5  #try also r<-2
PEdom.num(Xp,Yp,r,M)
```

`PEdom.num.binom.test` *A test of segregation/association based on domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data - Binomial Approximation*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the convex hull of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points i.e., cluster around the centers of the Delaunay triangles) and association (where `Xp` points cluster around `Yp` points) based on the (asymptotic) binomial distribution of the domination number of PE-PCD for uniform 2D data in the convex hull of `Yp` points.

The function yields the test statistic, p-value for the corresponding `alternative`, the confidence interval, estimate and null value for the parameter of interest (which is $Pr(\text{domination number} \leq 2)$), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the convex hull of `Yp` points, probability of success (i.e., $Pr(\text{domination number} \leq 2)$) equals to its expected value under the uniform distribution) and `alternative` could be two-sided, or right-sided (i.e., data is accumulated around the `Yp` points, or association) or left-sided (i.e., data is accumulated around the centers of the triangles, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and M-vertex regions where M is a center that yields non-degenerate asymptotic distribution of the domination number.

The test statistic is based on the binomial distribution, when success is defined as domination number being less than or equal to 2 in the one triangle case (i.e., number of failures is equal to number of times restricted domination number = 3 in the triangles). That is, the test statistic is based on the domination number for `Xp` points inside convex hull of `Yp` points for the PE-PCD and default convex hull correction, `ch.cor`, is `FALSE` where `M` is the center that yields nondegenerate asymptotic distribution for the domination number. For this approximation to work, number of `Xp` points must be at least 7 times more than number of `Yp` points.

PE proximity region is constructed with the expansion parameter $r \geq 1$ and CM-vertex regions (i.e., the test is not available for a general center M at this version of the function).

Caveat: This test is currently a conditional test, where `Xp` points are assumed to be random, while `Yp` points are assumed to be fixed (i.e., the test is conditional on `Yp` points). Furthermore, the test is a large sample test when `Xp` points are substantially larger than `Yp` points, say at least 7 times more. This test is more appropriate when supports of `Xp` and `Yp` have a substantial overlap. Currently, the `Xp` points outside the convex hull of `Yp` points are handled with a convex hull correction factor (see the description below and the function code.) Removing the conditioning and extending it to the case of non-concurring supports is an ongoing topic of research of the author of the package.

See also (Ceyhan (2011)).

Usage

```
PEdom.num.binom.test(
  Xp,
  Yp,
  r,
  ch.cor = FALSE,
  ndt = NULL,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$.
`ch.cor`	A logical argument for convex hull correction, default `ch.cor=FALSE`, recommended when both `Xp` and `Yp` have the same rectangular support.
`ndt`	Number of Delaunay triangles based on `Yp` points, default is `NULL`.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the probability of success (i.e., $Pr($domination number$=3)$ for PE-PCD whose vertices are the 2D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`
`conf.int`	Confidence interval for Pr(Domination Number≤ 2) at the given level `conf.level` and depends on the type of `alternative`.
`estimate`	A `vector` with two entries: first is is the estimate of the parameter, i.e., Pr(Domination Number$= 3$) and second is the domination number
`null.value`	Hypothesized value for the parameter, i.e., the null value for Pr(Domination Number≤ 2)
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

`PEdom.num.norm.test`

Examples

```
nx<-100; ny<-5 #try also nx<-1000; ny<-10
r<-1.4  #try also r<-1.5

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

plotDelaunay.tri(Xp,Yp,xlab="",ylab="")
PEdom.num.binom.test(Xp,Yp,r) #try also #PEdom.num.binom.test(Xp,Yp,r,alt="l") and
# PEdom.num.binom.test(Xp,Yp,r,alt="g")
PEdom.num.binom.test(Xp,Yp,r,ch=TRUE)

#or try
ndt<-num.delaunay.tri(Yp)
PEdom.num.binom.test(Xp,Yp,r,ndt=ndt)
```

```
#values might differ due to the random of choice of the three centers M1,M2,M3
#for the non-degenerate asymptotic distribution of the domination number
```

`PEdom.num.binom.test1D` *A test of segregation/association based on domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - Binomial Approximation*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points within the partition intervals based on `Yp` points (both residing in the support interval (a, b)). The test is for testing the spatial interaction between `Xp` and `Yp` points.

The null hypothesis is uniformity of `Xp` points on $(y_{\min}, y_{\max})$ (by default) where $y_{\min}$ and $y_{\max}$ are minimum and maximum of `Yp` points, respectively. `Yp` determines the end points of the intervals (i.e., partition the real line via its spacings called intervalization) where end points are the order statistics of `Yp` points. If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

The alternatives are segregation (where `Xp` points cluster away from `Yp` points i.e., cluster around the centers of the partition intervals) and association (where `Xp` points cluster around `Yp` points). The test is based on the (asymptotic) binomial distribution of the domination number of PE-PCD for uniform 1D data in the partition intervals based on `Yp` points.

The test by default is restricted to the range of `Yp` points, and so ignores `Xp` points outside this range. However, a correction for the `Xp` points outside the range of `Yp` points is available by setting `end.int.cor=TRUE`, which is recommended when both `Xp` and `Yp` have the same interval support.

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is $Pr(\text{domination number} \leq 1)$), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the intervals based on `Yp` points, probability of success (i.e., $Pr(\text{domination number} \leq 1)$) equals to its expected value) and `alternative` could be two-sided, or left-sided (i.e., data is accumulated around the `Yp` points, or association) or right-sided (i.e., data is accumulated around the centers of the partition intervals, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and centrality parameter `c` which yields M-vertex regions. More precisely, for a middle interval $(y_{(i)}, y_{(i+1)})$, the center is $M = y_{(i)} + c(y_{(i+1)} - y_{(i)})$ for the centrality parameter `c`. For a given $c \in (0, 1)$, the expansion parameter r is taken to be $1/\max(c, 1-c)$ which yields non-degenerate asymptotic distribution of the domination number.

The test statistic is based on the binomial distribution, when success is defined as domination number being less than or equal to 1 in the one interval case (i.e., number of successes is equal to domination number ≤ 1 in the partition intervals). That is, the test statistic is based on the domination number for `Xp` points inside range of `Yp` points (the domination numbers are summed over the $|Yp| - 1$ middle intervals) for the PE-PCD and default end-interval correction, `end.int.cor`, is `FALSE` and the center Mc is chosen so that asymptotic distribution for the domination number is nondegenerate. For this test to work, `Xp` must be at least 10 times more than `Yp` points (or `Xp` must be at least 5 or more per partition interval). Probability of success is the exact probability of success for the binomial distribution.

Caveat: This test is currently a conditional test, where `Xp` points are assumed to be random, while `Yp` points are assumed to be fixed (i.e., the test is conditional on `Yp` points). This test is more appropriate when supports of `Xp` and `Yp` have a substantial overlap. Currently, the `Xp` points outside the range of `Yp` points are handled with an end-interval correction factor (see the description below and the function code.) Removing the conditioning and extending it to the case of non-concurring supports is an ongoing line of research of the author of the package.

See also (Ceyhan (2020)) for more on the uniformity test based on the arc density of PE-PCDs.

Usage

```
PEdom.num.binom.test1D(
  Xp,
  Yp,
  c = 0.5,
  support.int = NULL,
  end.int.cor = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 1D points which constitute the end points of the partition intervals.
`c`	A positive real number which serves as the centrality parameter in PE proximity region; must be in $(0, 1)$ (default `c=.5`).
`support.int`	Support interval (a, b) with $a < b$. Uniformity of `Xp` points in this interval is tested. Default is `NULL`.
`end.int.cor`	A logical argument for end-interval correction, default is `FALSE`, recommended when both `Xp` and `Yp` have the same interval support.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the probability of success (i.e., $Pr($domination number$\leq 1)$ for PE-PCD whose vertices are the 1D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`.

`conf.int`	Confidence interval for Pr(domination number≤ 1) at the given level `conf.level` and depends on the type of `alternative`.
`estimate`	A `vector` with two entries: first is is the estimate of the parameter, i.e., Pr(domination number≤ 1) and second is the domination number
`null.value`	Hypothesized value for the parameter, i.e., the null value for Pr(domination number≤ 1)
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

Ceyhan E (2020). “Domination Number of an Interval Catch Digraph Family and Its Use for Testing Uniformity.” *Statistics*, **54(2)**, 310-339.

See Also

`PEdom.num.binom.test` and `PEdom.num1D`

Examples

```
a<-0; b<-10; supp<-c(a,b)
c<-.4

r<-1/max(c,1-c)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-100; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)
PEdom.num.binom.test1D(Xp,Yp,c,supp)
PEdom.num.binom.test1D(Xp,Yp,c,supp,alt="l")
PEdom.num.binom.test1D(Xp,Yp,c,supp,alt="g")
PEdom.num.binom.test1D(Xp,Yp,c,supp,end=TRUE)
```

`PEdom.num.binom.test1Dint` *A test of uniformity for 1D data based on domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) - Binomial Approximation*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of uniformity of `Xp` points in the support interval (a,b)).

The support interval (a,b) is partitioned as `(b-a)*(0:nint)/nint` where `nint=round(sqrt(nx),0)` and `nx` is number of `Xp` points, and the test is for testing the uniformity of `Xp` points in the interval (a,b).

The null hypothesis is uniformity of `Xp` points on (a,b). The alternative is deviation of distribution of `Xp` points from uniformity. The test is based on the (asymptotic) binomial distribution of the domination number of PE-PCD for uniform 1D data in the partition intervals based on partition of (a,b).

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is $Pr(\text{domination number} \leq 1)$), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the support interval, probability of success (i.e., $Pr(\text{domination number} \leq 1)$) equals to its expected value) and `alternative` could be two-sided, or left-sided (i.e., data is accumulated around the end points of the partition intervals of the support) or right-sided (i.e., data is accumulated around the centers of the partition intervals).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and centrality parameter `c` which yields M-vertex regions. More precisely $M_c = a + c(b-a)$ for the centrality parameter `c` and for a given $c \in (0,1)$, the expansion parameter r is taken to be $1/\max(c, 1-c)$ which yields non-degenerate asymptotic distribution of the domination number.

The test statistic is based on the binomial distribution, when success is defined as domination number being less than or equal to 1 in the one interval case (i.e., number of failures is equal to number of times restricted domination number = 1 in the intervals). That is, the test statistic is based on the domination number for `Xp` points inside the partition intervals for the PE-PCD. For this approach to work, `Xp` must be large for each partition interval, but 5 or more per partition interval seems to work in practice.

Probability of success is chosen in the following way for various parameter choices. `asy.bin` is a logical argument for the use of asymptotic probability of success for the binomial distribution, default is `asy.bin=FALSE`. When `asy.bin=TRUE`, asymptotic probability of success for the binomial distribution is used. When `asy.bin=FALSE`, the finite sample probability of success for the binomial distribution is used with number of trials equals to expected number of `Xp` points per partition interval.

Usage

```
PEdom.num.binom.test1Dint(
```

```
  Xp,
  support.int,
  c = 0.5,
  asy.bin = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 1D points which constitute the vertices of the PE-PCD.
`support.int`	Support interval (a, b) with $a < b$. Uniformity of `Xp` points in this interval is tested.
`c`	A positive real number which serves as the centrality parameter in PE proximity region; must be in $(0, 1)$ (default `c=.5`).
`asy.bin`	A logical argument for the use of asymptotic probability of success for the binomial distribution, default `asy.bin=FALSE`. When `asy.bin=TRUE`, asymptotic probability of success for the binomial distribution is used. When `asy.bin=FALSE`, the finite sample asymptotic probability of success for the binomial distribution is used with number of trials equals to expected number of `Xp` points per partition interval.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the probability of success (i.e., $Pr($domination number$\leq 1)$ for PE-PCD whose vertices are the 1D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`
`conf.int`	Confidence interval for $Pr($domination number$\leq 1)$ at the given level `conf.level` and depends on the type of `alternative`.
`estimate`	A `vector` with two entries: first is is the estimate of the parameter, i.e., $Pr($domination number$\leq 1)$ and second is the domination number
`null.value`	Hypothesized value for the parameter, i.e., the null value for $Pr($domination number$\leq 1)$
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

`PEdom.num.binom.test`, `PEdom.num1D` and `PEdom.num1Dnondeg`

Examples

```
a<-0; b<-10; supp<-c(a,b)
c<-.4

r<-1/max(c,1-c)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-100; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)

PEdom.num.binom.test1Dint(Xp,supp,c,alt="t")
PEdom.num.binom.test1Dint(Xp,support.int = supp,c=c,alt="t")
PEdom.num.binom.test1Dint(Xp,supp,c,alt="l")
PEdom.num.binom.test1Dint(Xp,supp,c,alt="g")
PEdom.num.binom.test1Dint(Xp,supp,c,alt="t",asy.bin = TRUE)
```

`PEdom.num.nondeg` *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) with non-degeneracy centers - multiple triangle case*

Description

Returns the domination number, indices of a minimum dominating set of PE-PCD whose vertices are the data points in `Xp` in the multiple triangle case and domination numbers for the Delaunay triangles based on `Yp` points when PE-PCD is constructed with vertex regions based on non-degeneracy centers.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center M which is one of the 3 centers that renders the asymptotic distribution of domination number to be non-degenerate for a given value of `r` in $(1, 1.5)$ and `M` is center of mass for $r = 1.5$.

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). Loops are allowed for the domination number.

See (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)) more on the domination number of PE-PCDs. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
PEdom.num.nondeg(Xp, Yp, r)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ here.

Value

A `list` with three elements

`dom.num`	Domination number of the PE-PCD whose vertices are `Xp` points. PE proximity regions are constructed with respect to the Delaunay triangles based on the `Yp` points with expansion parameter $rin(1, 1.5]$.
`#`	
`mds`	A minimum dominating set of the PE-PCD whose vertices are `Xp` points.
`ind.mds`	The data indices of the minimum dominating set of the PE-PCD whose vertices are `Xp` points.
`tri.dom.nums`	Domination numbers of the PE-PCD components for the Delaunay triangles.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`PEdom.num.tri`, `PEdom.num.tetra`, `dom.num.exact`, and `dom.num.greedy`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

r<-1.5  #try also r<-2

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

PEdom.num.nondeg(Xp,Yp,r)
```

`PEdom.num.norm.test` *A test of segregation/association based on domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data - Normal Approximation*

Description

An object of class `"htest"` (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the convex hull of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points i.e., cluster around the centers of the Delaunay triangles) and association (where `Xp` points cluster around `Yp` points) based on the normal approximation to the binomial distribution of the domination number of PE-PCD for uniform 2D data in the convex hull of `Yp` points

The function yields the test statistic, p-value for the corresponding `alternative`, the confidence interval, estimate and null value for the parameter of interest (which is $Pr(\text{domination number} \leq 2)$), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the convex hull of `Yp` points, probability of success (i.e., $Pr(\text{domination number} \leq 2)$) equals to its expected value under the uniform distribution) and `alternative` could be two-sided, or right-sided (i.e., data is accumulated around the `Yp` points, or association) or left-sided (i.e., data is accumulated around the centers of the triangles, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and M-vertex regions where M is a center that yields non-degenerate asymptotic distribution of the domination number.

The test statistic is based on the normal approximation to the binomial distribution, when success is defined as domination number being less than or equal to 2 in the one triangle case (i.e., number of failures is equal to number of times restricted domination number = 3 in the triangles). That is, the test statistic is based on the domination number for `Xp` points inside convex hull of `Yp` points for the PE-PCD and default convex hull correction, `ch.cor`, is `FALSE` where `M` is the center that yields nondegenerate asymptotic distribution for the domination number.

For this approximation to work, number of `Yp` points must be at least 5 (i.e., about 7 or more Delaunay triangles) and number of `Xp` points must be at least 7 times more than the number of `Yp` points.

See also (Ceyhan (2011)).

Usage

```
PEdom.num.norm.test(
  Xp,
  Yp,
  r,
  ch.cor = FALSE,
  ndt = NULL,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$.
`ch.cor`	A logical argument for convex hull correction, default `ch.cor=FALSE`, recommended when both `Xp` and `Yp` have the same rectangular support.
`ndt`	Number of Delaunay triangles based on `Yp` points, default is `NULL`.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`.
`conf.level`	Level of the confidence interval, default is `0.95`, for the domination number of PE-PCD whose vertices are the 2D data set `Xp`.

Value

A `list` with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding `alternative`

`conf.int`	Confidence interval for the domination number at the given level `conf.level` and depends on the type of `alternative`.
`estimate`	A `vector` with two entries: first is the domination number, and second is the estimate of the parameter, i.e., $Pr(\text{Domination Number}=3)$
`null.value`	Hypothesized value for the parameter, i.e., the null value for expected domination number
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set

Author(s)

Elvan Ceyhan

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

`PEdom.num.binom.test`

Examples

```
nx<-100; ny<-5 #try also nx<-1000; ny<-10
r<-1.5  #try also r<-2 or r<-1.25

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

plotDelaunay.tri(Xp,Yp,xlab="",ylab="")
PEdom.num.norm.test(Xp,Yp,r) #try also PEdom.num.norm.test(Xp,Yp,r, alt="l")

PEdom.num.norm.test(Xp,Yp,1.25,ch=TRUE)

#or try
ndt<-num.delaunay.tri(Yp)
PEdom.num.norm.test(Xp,Yp,r,ndt=ndt)
#values might differ due to the random of choice of the three centers M1,M2,M3
#for the non-degenerate asymptotic distribution of the domination number
```

`PEdom.num.tetra` *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) - one tetrahedron case*

Description

Returns the domination number of PE-PCD whose vertices are the data points in `Xp`.

PE proximity region is defined with respect to the tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions are based on the center `M` which is circumcenter (`"CC"`) or center of mass (`"CM"`) of `th` with default=`"CM"`.

See also (Ceyhan (2005, 2010)).

Usage

```
PEdom.num.tetra(Xp, th, r, M = "CM")
```

Arguments

`Xp`	A set of 3D points which constitute the vertices of the digraph.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.

Value

A `list` with two elements

`dom.num`	Domination number of PE-PCD with vertex set = `Xp` and expansion parameter $r \geq 1$ and center `M`
`mds`	A minimum dominating set of PE-PCD with vertex set = `Xp` and expansion parameter $r \geq 1$ and center `M`
`ind.mds`	Indices of the minimum dominating set `mds`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

PEdom.num.tri

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-10  #try also n<-20

Xp<-runif.tetra(n,tetra)$g

M<-"CM"  #try also M<-"CC"
r<-1.25

PEdom.num.tetra(Xp,tetra,r,M)

P1<-c(.5,.5,.5)
PEdom.num.tetra(P1,tetra,r,M)
```

`PEdom.num.tri` *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) - one triangle case*

Description

Returns the domination number of PE-PCD whose vertices are the data points in `Xp`.

PE proximity region is defined with respect to the triangle `tri` with expansion parameter $r \geq 1$ and vertex regions are constructed with center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or the circumcenter of `tri`.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
PEdom.num.tri(Xp, tri, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the digraph.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $(1, 1, 1)$, i.e., the center of mass.

Value

A `list` with two elements

`dom.num`	Domination number of PE-PCD with vertex set = `Xp` and expansion parameter $r \geq 1$ and center `M`
`mds`	A minimum dominating set of PE-PCD with vertex set = `Xp` and expansion parameter $r \geq 1$ and center `M`
`ind.mds`	Indices of the minimum dominating set `mds`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE (2007). “On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs.” *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`PEdom.num.nondeg`, `PEdom.num`, and `PEdom.num1D`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2)
Tr<-rbind(A,B,C)
n<-10  #try also n<-20
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1,1,1)

r<-1.4

PEdom.num.tri(Xp,Tr,r,M)
IM<-inci.matPEtri(Xp,Tr,r,M)
dom.num.greedy #try also dom.num.exact(IM)

gr.gam<-dom.num.greedy(IM)
gr.gam
```

```
Xp[gr.gam$i,]

PEdom.num.tri(Xp,Tr,r,M=c(.4,.4))
```

PEdom.num1D *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data*

Description

Returns the domination number, a minimum dominating set of PE-PCD whose vertices are the 1D data set `Xp`, and the domination numbers for partition intervals based on `Yp`.

`Yp` determines the end points of the intervals (i.e., partition the real line via intervalization). It also includes the domination numbers in the end-intervals, with interval label 1 for the left end-interval and $|Yp|+1$ for the right end-interval.

If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

PE proximity region is constructed with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$.

Usage

```
PEdom.num1D(Xp, Yp, r, c = 0.5)
```

Arguments

`Xp`	A set of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 1D points which constitute the end points of the intervals which partition the real line.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside `int` (default `c=.5`).

Value

A `list` with three elements

`dom.num`	Domination number of PE-PCD with vertex set `Xp` and expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$.
`mds`	A minimum dominating set of the PE-PCD.
`ind.mds`	The data indices of the minimum dominating set of the PE-PCD whose vertices are `Xp` points.
`int.dom.nums`	Domination numbers of the PE-PCD components for the partition intervals.

Author(s)

Elvan Ceyhan

See Also

`PEdom.num.nondeg`

Examples

```
a<-0; b<-10
c<-.4
r<-2

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

PEdom.num1D(Xp,Yp,r,c)

PEdom.num1D(Xp,Yp,r,c=.25)
PEdom.num1D(Xp,Yp,r=1.25,c)
```

`PEdom.num1Dnondeg` *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) with non-degeneracy centers - multiple interval case*

Description

Returns the domination number, a minimum dominating set of PE-PCD whose vertices are the 1D data set `Xp`, and the domination numbers for partition intervals based on `Yp` when PE-PCD is constructed with vertex regions based on non-degeneracy centers.

`Yp` determines the end points of the intervals (i.e., partition the real line via intervalization). If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

PE proximity regions are defined with respect to the intervals based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each interval are based on the centrality parameter `c` which is one of the 2 values of `c` (i.e., $c \in \{(r-1)/r, 1/r\}$) that renders the asymptotic distribution of domination number to be non-degenerate for a given value of `r` in $(1, 2)$ and `c` is center of mass for $r = 2$. These values are called non-degeneracy centrality parameters and the corresponding centers are called nondegeneracy centers.

Usage

```
PEdom.num1Dnondeg(Xp, Yp, r)
```

Arguments

`Xp`	A set of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 1D points which constitute the end points of the intervals which partition the real line.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 2]$ here.

Value

A `list` with three elements

`dom.num`	Domination number of PE-PCD with vertex set `Xp` and expansion parameter $rin(1, 2]$ and centrality parameter $c \in \{(r-1)/r, 1/r\}$.
`mds`	A minimum dominating set of the PE-PCD.
`ind.mds`	The data indices of the minimum dominating set of the PE-PCD whose vertices are `Xp` points.
`int.dom.nums`	Domination numbers of the PE-PCD components for the partition intervals.

Author(s)

Elvan Ceyhan

See Also

`PEdom.num.nondeg`

Examples

```
a<-0; b<-10
r<-1.5

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

PEdom.num1Dnondeg(Xp,Yp,r)
PEdom.num1Dnondeg(Xp,Yp,r=1.25)
```

`perpline` *The line passing through a point and perpendicular to the line segment joining two points*

Description

An object of class `"Lines"`. Returns the `equation`, `slope`, `intercept`, and y-coordinates of the line crossing the point `p` and perpendicular to the line passing through the points `a` and `b` with x-coordinates are provided in `vector` `x`.

Usage

```
perpline(p, a, b, x)
```

Arguments

`p`	A 2D point at which the perpendicular line to line segment joining `a` and `b` crosses.
`a, b`	2D points that determine the line segment (the line will be perpendicular to this line segment).
`x`	A scalar or a `vector` of scalars representing the x-coordinates of the line perpendicular to line joining `a` and `b` and crossing `p`.

Value

A `list` with the elements

`desc`	Description of the line passing through point `p` and perpendicular to line joining `a` and `b`
`mtitle`	The `"main"` title for the plot of the line passing through point `p` and perpendicular to line joining `a` and `b`
`points`	The input points `a` and `b` (stacked row-wise, i.e., row 1 is point `a` and row 2 is point `b`). Line passing through point `p` is perpendicular to line joining `a` and `b`
`x`	The input vector, can be a scalar or a `vector` of scalars, which constitute the x-coordinates of the point(s) of interest on the line passing through point `p` and perpendicular to line joining `a` and `b`
`y`	The output `vector` which constitutes the y-coordinates of the point(s) of interest on the line passing through point `p` and perpendicular to line joining `a` and `b`. If `x` is a scalar, then `y` will be a scalar and if `x` is a `vector` of scalars, then `y` will be a `vector` of scalars.
`slope`	Slope of the line passing through point `p` and perpendicular to line joining `a` and `b`
`intercept`	Intercept of the line passing through point `p` and perpendicular to line joining `a` and `b`
`equation`	Equation of the line passing through point `p` and perpendicular to line joining `a` and `b`

Author(s)

Elvan Ceyhan

See Also

`slope`, `Line`, and `paraline`

Examples

```
A<-c(1.1,1.2); B<-c(2.3,3.4); p<-c(.51,2.5)

perpline(p,A,B,.45)

pts<-rbind(A,B,p)
xr<-range(pts[,1])
xf<-(xr[2]-xr[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100

plnAB<-perpline(p,A,B,x)
plnAB
summary(plnAB)
plot(plnAB,asp=1)

y<-plnAB$y
Xlim<-range(x,pts[,1])
if (!is.na(y[1])) {Ylim<-range(y,pts[,2])} else {Ylim<-range(pts[,2])}
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
pf<-c(xd,-yd)*.025

plot(A,asp=1,pch=".",xlab="",ylab="",
main="Line Crossing p and Perpendicular to AB",
xlim=Xlim+xd*c(-.5,.5),ylim=Ylim+yd*c(-.05,.05))
points(pts)
txt.str<-c("A","B","p")
text(pts+rbind(pf,pf,pf),txt.str)

segments(A[1],A[2],B[1],B[2],lty=2)
if (!is.na(y[1])) {lines(x,y,type="l",lty=1,
xlim=Xlim,ylim=Ylim)} else {abline(v=p[1])}
tx<-p[1]+abs(xf-p[1])/2;
if (!is.na(y[1])) {ty<-perpline(p,A,B,tx)$y} else {ty=p[2]}
text(tx,ty,"line perpendicular to AB\n and crossing p")
```

`perpline2plane` *The line crossing the 3D point* `p` *and perpendicular to the plane spanned by 3D points* `a`, `b`, *and* `c`

Description

An object of class `"Lines3D"`. Returns the equation, x-, y-, and z-coordinates of the line crossing 3D point `p` and perpendicular to the plane spanned by 3D points `a`, `b`, and `c` (i.e., the line is in the direction of normal `vector` of this plane) with the parameter `t` being provided in `vector` `t`.

Usage

```
perpline2plane(p, a, b, c, t)
```

Arguments

`p`	A 3D point through which the straight line passes.
`a, b, c`	3D points which determine the plane to which the line passing through point `p` would be perpendicular (i.e., the normal `vector` of this plane determines the direction of the straight line passing through `p`).
`t`	A scalar or a `vector` of scalars representing the parameter of the coordinates of the line (for the form: $x = p_0 + At$, $y = y_0 + Bt$, and $z = z_0 + Ct$ where $p = (p_0, y_0, z_0)$ and normal vector= (A, B, C)).

Value

A `list` with the elements

`desc`	A description of the line
`mtitle`	The `"main"` title for the plot of the line
`points`	The input points that determine the line and plane, line crosses point `p` and plane is determined by 3D points `a`, `b`, and `c`.
`pnames`	The names of the input points that determine the line and plane; line would be perpendicular to the plane.
`vecs`	The point `p` and normal vector.
`vec.names`	The names of the point `p` and the second entry is "normal vector".
`x, y, z`	The x-, y-, and z-coordinates of the point(s) of interest on the line perpendicular to the plane determined by points `a`, `b`, and `c`.
`tsq`	The scalar or the `vector` of the parameter in defining each coordinate of the line for the form: $x = p_0 + At$, $y = y_0 + Bt$, and $z = z_0 + Ct$ where $p = (p_0, y_0, z_0)$ and normal vector= (A, B, C).
`equation`	Equation of the line passing through point `p` and perpendicular to the plane determined by points `a`, `b`, and `c` (i.e., line is in the direction of the normal `vector` N of the plane). The line equation is in the form: $x = p_0 + At$, $y = y_0 + Bt$, and $z = z_0 + Ct$ where $p = (p_0, y_0, z_0)$ and normal vector= (A, B, C).

Author(s)

Elvan Ceyhan

See Also

`Line3D`, `paraline3D`, and `perpline`

Examples

```
P<-c(1,1,1); Q<-c(1,10,4); R<-c(1,1,3); S<-c(3,9,12)

cf<-as.numeric(Plane(Q,R,S,1,1)$coeff)
a<-cf[1]; b<-cf[2]; c<- -1;

vecs<-rbind(Q,c(a,b,c))
pts<-rbind(P,Q,R,S)
perpline2plane(P,Q,R,S,.1)

tr<-range(pts,vecs);
tf<-(tr[2]-tr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=5)  #try also l=10, 20, or 100

pln2pl<-perpline2plane(P,Q,R,S,tsq)
pln2pl
summary(pln2pl)
plot(pln2pl,theta = 225, phi = 30, expand = 0.7,
facets = FALSE, scale = TRUE)

xc<-pln2pl$x
yc<-pln2pl$y
zc<-pln2pl$z

zr<-range(zc)
zf<-(zr[2]-zr[1])*.2
Rv<- -c(a,b,c)*zf*5

Dr<-(Q+R+S)/3

pts2<-rbind(Q,R,S)
xr<-range(pts2[,1],xc); yr<-range(pts2[,2],yc)
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1
#how far to go at the lower and upper ends in the y-coordinate
xs<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100
ys<-seq(yr[1]-yf,yr[2]+yf,l=5)  #try also l=10, 20, or 100

plQRS<-Plane(Q,R,S,xs,ys)
z.grid<-plQRS$z

Xlim<-range(xc,xs,pts[,1])
```

```
Ylim<-range(yc,ys,pts[,2])
Zlim<-range(zc,z.grid,pts[,3])

xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::persp3D(z = z.grid, x = xs, y = ys, theta =225, phi = 30,
main="Line Crossing P and \n Perpendicular to the Plane Defined by Q, R, S",
col="lightblue", ticktype = "detailed",
        xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),
        zlim=Zlim+zd*c(-.05,.05))
        #plane spanned by points Q, R, S
plot3D::lines3D(xc, yc, zc, bty = "g",pch = 20, cex = 2,col="red",
ticktype = "detailed",add=TRUE)
plot3D::arrows3D(Dr[1],Dr[2],Dr[3],Dr[1]+Rv[1],Dr[2]+Rv[2],
Dr[3]+Rv[3], add=TRUE)
plot3D::points3D(pts[,1],pts[,2],pts[,3],add=TRUE)
plot3D::text3D(pts[,1],pts[,2],pts[,3],labels=c("P","Q","R","S"),add=TRUE)
plot3D::arrows3D(P[1],P[2],P[3]-zf,P[1],P[2],P[3],lty=2, add=TRUE)
plot3D::text3D(P[1],P[2],P[3]-zf,labels="initial point",add=TRUE)
plot3D::text3D(P[1],P[2],P[3]+zf/2,labels="P",add=TRUE)
plot3D::arrows3D(Dr[1],Dr[2],Dr[3],Dr[1]+Rv[1]/2,Dr[2]+Rv[2]/2,
Dr[3]+Rv[3]/2,lty=2, add=TRUE)
plot3D::text3D(Dr[1]+Rv[1]/2,Dr[2]+Rv[2]/2,Dr[3]+Rv[3]/2,
labels="normal vector",add=TRUE)
```

`Plane` *The plane passing through three distinct 3D points* `a`, `b`, *and* `c`

Description

An object of class `"Planes"`. Returns the equation and z-coordinates of the plane passing through three distinct 3D points `a`, `b`, and `c` with x- and y-coordinates are provided in vectors `x` and `y`, respectively.

Usage

```
Plane(a, b, c, x, y)
```

Arguments

a, b, c	3D points that determine the plane (i.e., through which the plane is passing).
x, y	Scalars or vectors of scalars representing the x- and y-coordinates of the plane.

Value

A `list` with the elements

`desc`	A description of the plane
`points`	The input points `a`, `b`, and `c` through which the plane is passing (stacked row-wise, i.e., row 1 is point `a`, row 2 is point `b` and row 3 is point `c`).
`x, y`	The input vectors which constitutes the x- and y-coordinates of the point(s) of interest on the plane. `x` and `y` can be scalars or vectors of scalars.
`z`	The output `vector` which constitutes the z-coordinates of the point(s) of interest on the plane. If `x` and `y` are scalars, `z` will be a scalar and if `x` and `y` are vectors of scalars, then `z` needs to be a `matrix` of scalars, containing the z-coordinate for each pair of `x` and `y` values.
`coeff`	Coefficients of the plane (in the $z = Ax + By + C$ form).
`equation`	Equation of the plane in long form
`equation2`	Equation of the plane in short form, to be inserted on the plot

Author(s)

Elvan Ceyhan

See Also

`paraplane`

Examples

```
P1<-c(1,10,3); P2<-c(1,1,3); P3<-c(3,9,12) #also try P2=c(2,2,3)

pts<-rbind(P1,P2,P3)
Plane(P1,P2,P3,.1,.2)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1
#how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5)  #try also l=10, 20, or 100
y<-seq(yr[1]-yf,yr[2]+yf,l=5)  #try also l=10, 20, or 100

plP123<-Plane(P1,P2,P3,x,y)
plP123
summary(plP123)
plot(plP123,theta = 225, phi = 30, expand = 0.7, facets = FALSE, scale = TRUE)

z.grid<-plP123$z

persp(x,y,z.grid, xlab="x",ylab="y",zlab="z",
theta = -30, phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.05, ticktype = "detailed")
```

```
zr<-max(z.grid)-min(z.grid)
Pts<-rbind(P1,P2,P3)+rbind(c(0,0,zr*.1),c(0,0,zr*.1),c(0,0,zr*.1))
Mn.pts<-apply(Pts,2,mean)

plot3D::persp3D(z = z.grid, x = x, y = y,theta = 225, phi = 30, expand = 0.3,
main = "Plane Crossing Points P1, P2, and P3", facets = FALSE, scale = TRUE)
#plane spanned by points P1, P2, P3
#add the defining points
plot3D::points3D(Pts[,1],Pts[,2],Pts[,3], add=TRUE)
plot3D::text3D(Pts[,1],Pts[,2],Pts[,3], c("P1","P2","P3"),add=TRUE)
plot3D::text3D(Mn.pts[1],Mn.pts[2],Mn.pts[3],plP123$equation,add=TRUE)
#plot3D::polygon3D(Pts[,1],Pts[,2],Pts[,3], add=TRUE)
```

`plot.Extrema` *Plot an* `Extrema` object

Description

Plots the data points and extrema among these points together with the reference object (e.g., boundary of the support region)

Usage

```
## S3 method for class 'Extrema'
plot(x, asp = NA, xlab = "", ylab = "", zlab = "", ...)
```

Arguments

`x`	Object of class `Extrema`.
`asp`	A `numeric` value, giving the aspect ratio for y-axis to x-axis y/x for the 2D case, it is redundant in the 3D case (default is `NA`), see the official help for `asp` by typing "? asp".
`xlab, ylab, zlab`	Titles for the x and y axes in the 2D case, and x, y, and z axes in the 3D case, respectively (default is `""` for all).
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.Extrema`, `summary.Extrema`, and `print.summary.Extrema`

Examples

```
n<-10
Xp<-runif.std.tri(n)$gen.points
Ext<-cl2edges.std.tri(Xp)
Ext
plot(Ext,asp=1)
```

`plot.Lines` *Plot a* `Lines` object

Description

Plots the line together with the defining `points`.

Usage

```
## S3 method for class 'Lines'
plot(x, asp = NA, xlab = "x", ylab = "y", ...)
```

Arguments

`x`	Object of class `Lines`.
`asp`	A `numeric` value, giving the aspect ratio for y-axis to x-axis y/x (default is `NA`), see the official help for `asp` by typing "`? asp`".
`xlab, ylab`	Titles for the x and y axes, respectively (default is `xlab="x"` and `ylab="y"`).
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.Lines`, `summary.Lines`, and `print.summary.Lines`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)
xr<-range(A,B);
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=3) #try also l=10, 20 or 100

lnAB<-Line(A,B,x)
lnAB
plot(lnAB)
```

`plot.Lines3D` *Plot a* `Lines3D` object

Description

Plots the line together with the defining vectors (i.e., the initial and direction vectors).

Usage

```
## S3 method for class 'Lines3D'
plot(x, xlab = "x", ylab = "y", zlab = "z", phi = 40, theta = 40, ...)
```

Arguments

`x`	Object of class `Lines3D`.
`xlab, ylab, zlab`	Titles for the x, y, and z axes, respectively (default is `xlab="x"`, `ylab="y"` and `zlab="z"`).
`theta, phi`	The angles defining the viewing direction. `theta` gives the azimuthal direction and `phi` the colatitude. See `persp3D` for more details.
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.Lines3D`, `summary.Lines3D`, and `print.summary.Lines3D`

Examples

```
P<-c(1,10,3); Q<-c(1,1,3);
vecs<-rbind(P,Q)
Line3D(P,Q,.1)
Line3D(P,Q,.1,dir.vec=FALSE)

tr<-range(vecs);
tf<-(tr[2]-tr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=3) #try also l=10, 20 or 100

lnPQ3D<-Line3D(P,Q,tsq)
lnPQ3D
plot(lnPQ3D)
```

---

plot.NumArcs *Plot a* `NumArcs` object

Description

Plots the scatter plot of the data points (i.e. vertices of the PCDs) and the Delaunay tessellation of the nontarget points marked with number of arcs in the centroid of the Delaunay cells.

Usage

```
## S3 method for class 'NumArcs'
plot(x, Jit = 0.1, ...)
```

Arguments

`x`	Object of class `NumArcs`.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default is 0.1, for the 1D case, the vertices of the PCD are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of vertices and the interval end points; it is redundant in the 2D case.
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.NumArcs`, `summary.NumArcs`, and `print.summary.NumArcs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10
Xp<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)
Arcs<-arcsAStri(Xp,Tr,M)
Arcs
plot(Arcs)
```

`plot.Patterns` *Plot a* `Patterns` object

Description

Plots the points generated from the pattern (color coded for each class) together with the study window

Usage

```
## S3 method for class 'Patterns'
plot(x, asp = NA, xlab = "x", ylab = "y", ...)
```

Arguments

`x`	Object of class `Patterns`.
`asp`	A `numeric` value, giving the aspect ratio for y-axis to x-axis y/x (default is `NA`), see the official help for `asp` by typing "`? asp`".
`xlab, ylab`	Titles for the x and y axes, respectively (default is `xlab="x"` and `ylab="y"`).
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.Patterns`, `summary.Patterns`, and `print.summary.Patterns`

Examples

```
nx<-10; #try also 100 and 1000
ny<-5; #try also 1
e<-.15;
Y<-cbind(runif(ny),runif(ny))
#with default bounding box (i.e., unit square)

Xdt<-rseg.circular(nx,Y,e)
Xdt
plot(Xdt,asp=1)
```

`plot.PCDs` *Plot a* `PCDs` `object`

Description

Plots the vertices and the arcs of the PCD together with the vertices and boundaries of the partition cells (i.e., intervals in the 1D case and triangles in the 2D case)

Usage

```
## S3 method for class 'PCDs'
plot(x, Jit = 0.1, ...)
```

Arguments

`x`	Object of class `PCDs`.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default is 0.1, for the 1D case, the vertices of the PCD are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of vertices and the interval end points; it is redundant in the 2D case.
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.PCDs`, `summary.PCDs`, and `print.summary.PCDs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10
Xp<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)
Arcs<-arcsAStri(Xp,Tr,M)
Arcs
plot(Arcs)
```

`plot.Planes` *Plot a* `Planes` `object`

Description

Plots the plane together with the defining 3D `points`.

Usage

```
## S3 method for class 'Planes'
plot(
  x,
  x.grid.size = 10,
  y.grid.size = 10,
  xlab = "x",
  ylab = "y",
  zlab = "z",
  phi = 40,
  theta = 40,
  ...
)
```

Arguments

`x`	Object of class `Planes`.
`x.grid.size`, `y.grid.size`	the size of the grids for the x and y axes, default is 10 for both
`xlab`, `ylab`, `zlab`	Titles for the x, y, and z axes, respectively (default is `xlab="x"`, `ylab="y"`, and `zlab="z"`).
`theta`, `phi`	The angles defining the viewing direction, default is 40 for both. `theta` gives the azimuthal direction and `phi` the colatitude. see `persp`.
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.Planes`, `summary.Planes`, and `print.summary.Planes`

Examples

```
P<-c(1,10,3); Q<-c(1,1,3); C<-c(3,9,12)
pts<-rbind(P,Q,C)

xr<-range(pts[,1]); yr<-range(pts[,2])
```

```
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1
#how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5) #try also l=10, 20 or 100
y<-seq(yr[1]-yf,yr[2]+yf,l=5) #try also l=10, 20 or 100

plPQC<-Plane(P,Q,C,x,y)
plPQC
plot(plPQC,theta = 225, phi = 30, expand = 0.7,
facets = FALSE, scale = TRUE)
```

`plot.TriLines` *Plot a* `TriLines` object

Description

Plots the line together with the defining triangle.

Usage

```
## S3 method for class 'TriLines'
plot(x, xlab = "x", ylab = "y", ...)
```

Arguments

`x`	Object of class `TriLines`.
`xlab, ylab`	Titles for the x and y axes, respectively (default is `xlab="x"` and `ylab="y"`).
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.TriLines`, `summary.TriLines`, and `print.summary.TriLines`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,l=3)

lnACM<-lineA2CMinTe(x)
```

```
lnACM
plot(lnACM)
```

`plot.Uniform` *Plot a* `Uniform` object

Description

Plots the points generated from the uniform distribution together with the support region

Usage

```
## S3 method for class 'Uniform'
plot(x, asp = NA, xlab = "x", ylab = "y", zlab = "z", ...)
```

Arguments

`x`	Object of class `Uniform`.
`asp`	A `numeric` value, giving the aspect ratio for y-axis to x-axis y/x for the 2D case, it is redundant in the 3D case (default is `NA`), see the official help for `asp` by typing "? asp".
`xlab`, `ylab`, `zlab`	Titles for the x and y axes in the 2D case, and x, y, and z axes in the 3D case, respectively (default is `xlab="x"`, `ylab="y"`, and `zlab="z"`).
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.Uniform`, `summary.Uniform`, and `print.summary.Uniform`

Examples

```
n<-10 #try also 20, 100, and 1000
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)

Xdt<-runif.tri(n,Tr)
Xdt
plot(Xdt,asp=1)
```

`plotASarcs` *The plot of the arcs of Arc Slice Proximity Catch Digraph (AS-PCD) for a 2D data set - multiple triangle case*

Description

Plots the arcs of AS-PCD whose vertices are the data points in `Xp` and Delaunay triangles based on `Yp` points.

AS proximity regions are constructed with respect to the Delaunay triangles based on `Yp` points, i.e., AS proximity regions are defined only for `Xp` points inside the convex hull of `Yp` points. That is, arcs may exist for `Xp` points only inside the convex hull of `Yp` points. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

Vertex regions are based on the center `M="CC"` for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is `M="CC"` i.e., circumcenter of each triangle. When the center is the circumcenter, `CC`, the vertex regions are constructed based on the orthogonal projections to the edges, while with any interior center `M`, the vertex regions are constructed using the extensions of the lines combining vertices with `M`.

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). Loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005, 2010)) for more on AS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotASarcs(
  Xp,
  Yp,
  M = "CC",
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangulation. The Delaunay triangles partition the convex hull of `Yp` points.

`M`	The center of the triangle. `"CC"` stands for circumcenter of each Delaunay triangle or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is `M="CC"` i.e., the circumcenter of each triangle.
`asp`	A `numeric` value, giving the aspect ratio for y axis to x-axis y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`".
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`...`	Additional `plot` parameters.

Value

A plot of the arcs of the AS-PCD for a 2D data set `Xp` where AS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points; also plots the Delaunay triangles based on `Yp` points.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`plotASarcs.tri`, `plotPEarcs.tri`, `plotPEarcs`, `plotCSarcs.tri`, and `plotCSarcs`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
```

```
set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

plotASarcs(Xp,Yp,M,asp=1,xlab="",ylab="")

plotASarcs(Xp,Yp[1:3,],M,asp=1,xlab="",ylab="")
```

---

plotASarcs.tri *The plot of the arcs of Arc Slice Proximity Catch Digraph (AS-PCD) for a 2D data set - one triangle case*

Description

Plots the arcs of AS-PCD whose vertices are the data points, `Xp` and also the triangle `tri`. AS proximity regions are constructed with respect to the triangle `tri`, i.e., only for `Xp` points inside the triangle `tri`. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`. When the center is the circumcenter, `CC`, the vertex regions are constructed based on the orthogonal projections to the edges, while with any interior center `M`, the vertex regions are constructed using the extensions of the lines combining vertices with `M`.

See also (Ceyhan (2005, 2010)).

Usage

```
plotASarcs.tri(
  Xp,
  tri,
  M = "CC",
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  vert.reg = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of `tri`.
`asp`	A `numeric` value, giving the aspect ratio for y axis to x-axis y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`".
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`vert.reg`	A logical argument to add vertex regions to the plot, default is `vert.reg=FALSE`.
`...`	Additional `plot` parameters.

Value

A plot of the arcs of the AS-PCD for a 2D data set `Xp` where AS proximity regions are defined with respect to the triangle `tri`; also plots the triangle `tri`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`plotASarcs`, `plotPEarcs.tri`, `plotPEarcs`, `plotCSarcs.tri`, and `plotCSarcs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp<-runif.tri(n,Tr)$g  #try also Xp<-cbind(runif(n,1,2),runif(n,0,2))

M<-as.numeric(runif.tri(1,Tr)$g)  #try also  #M<-c(1.6,1.2)

plotASarcs.tri(Xp,Tr,M,main="Arcs of AS-PCD",xlab="",ylab="")

plotASarcs.tri(Xp,Tr,M,main="Arcs of AS-PCD",xlab="",ylab="",vert.reg = TRUE)

# or try the default center
#plotASarcs.tri(Xp,Tr,asp=1,main="arcs of AS-PCD",xlab="",ylab="",vert.reg = TRUE);
#M = (arcsAStri(Xp,Tr)$param)$c #the part "M = as.numeric(arcsAStri(Xp,Tr)$param)" is optional,
#for the below annotation of the plot

#can add vertex labels and text to the figure (with vertex regions)
#but first we need to determine whether the center used for vertex regions is CC or not
#see the description for more detail.
CC<-circumcenter.tri(Tr)

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-prj.cent2edges(Tr,M)
}

#now we add the vertex names and annotation
txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]+c(-.02,.02,.02,.01,.05,-0.03,-.01)
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.05,-.06)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)
```

`plotASregs` *The plot of the Arc Slice (AS) Proximity Regions for a 2D data set - multiple triangle case*

Description

Plots the `Xp` points in and outside of the convex hull of `Yp` points and also plots the AS proximity regions for `Xp` points and Delaunay triangles based on `Yp` points.

AS proximity regions are constructed with respect to the Delaunay triangles based on `Yp` points (these triangles partition the convex hull of `Yp` points), i.e., AS proximity regions are only defined for `Xp` points inside the convex hull of `Yp` points.

Vertex regions are based on the center `M="CC"` for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is `M="CC"` i.e., circumcenter of each triangle.

See (Ceyhan (2005, 2010)) for more on AS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotASregs(
  Xp,
  Yp,
  M = "CC",
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points for which AS proximity regions are constructed.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangulation. The Delaunay triangles partition the convex hull of `Yp` points.
`M`	The center of the triangle. `"CC"` stands for circumcenter of each Delaunay triangle or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is `M="CC"` i.e., the circumcenter of each triangle.
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`...`	Additional `plot` parameters.

Value

Plot of the `Xp` points, Delaunay triangles based on `Yp` and also the AS proximity regions for `Xp` points inside the convex hull of `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`plotASregs.tri`, `plotPEregs.tri`, `plotPEregs`, `plotCSregs.tri`, and `plotCSregs`

Examples

```
nx<-10 ; ny<-5

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3) #or M="CC"

plotASregs(Xp,Yp,M,xlab="",ylab="")

plotASregs(Xp,Yp[1:3,],M,xlab="",ylab="")

Xp<-c(.5,.5)
plotASregs(Xp,Yp,M,xlab="",ylab="")
```

`plotASregs.tri` *The plot of the Arc Slice (AS) Proximity Regions for a 2D data set - one triangle case*

Description

Plots the points in and outside of the triangle `tri` and also the AS proximity regions for points in data set `Xp`.

AS proximity regions are defined with respect to the triangle `tri`, so AS proximity regions are defined only for points inside the triangle `tri` and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is `M="CC"`, i.e., circumcenter of `tri`. When vertex regions are constructed with circumcenter, `CC`, the vertex regions are constructed based on the orthogonal projections to the edges, while with any interior center `M`, the vertex regions are constructed using the extensions of the lines combining vertices with `M`.

See also (Ceyhan (2005, 2010)).

Usage

```
plotASregs.tri(
  Xp,
  tri,
  M = "CC",
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  vert.reg = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 2D points for which AS proximity regions are constructed.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. `"CC"` stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M="CC"` i.e., the circumcenter of `tri`.
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).

`vert.reg`	A logical argument to add vertex regions to the plot, default is `vert.reg=FALSE`.
`...`	Additional `plot` parameters.

Value

Plot of the AS proximity regions for points inside the triangle `tri` (and only the points outside `tri`)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`plotASregs`, `plotPEregs.tri`, `plotPEregs`, `plotCSregs.tri`, and `plotCSregs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp0<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)  #try also  #M<-c(1.6,1.2);

plotASregs.tri(Xp0,Tr,M,main="Proximity Regions for AS-PCD", xlab="",ylab="")
Xp = Xp0[1,]
plotASregs.tri(Xp,Tr,M,main="Proximity Regions for AS-PCD", xlab="",ylab="")

#can plot the arcs of the AS-PCD
#plotASarcs.tri(Xp,Tr,M,main="Arcs of AS-PCD",xlab="",ylab="")

plotASregs.tri(Xp,Tr,M,main="Proximity Regions for AS-PCD", xlab="",ylab="",vert.reg=TRUE)

# or try the default center
#plotASregs.tri(Xp,Tr,main="Proximity Regions for AS-PCD", xlab="",ylab="",vert.reg=TRUE);
M = (arcsAStri(Xp,Tr)$param)$c #the part "M = as.numeric(arcsAStri(Xp,Tr)$param)" is optional,
#for the below annotation of the plot
```

```
#can add vertex labels and text to the figure (with vertex regions)
#but first we need to determine whether the center used for vertex regions is CC or not
#see the description for more detail.
CC<-circumcenter.tri(Tr)
#Arcs<-arcsAStri(Xp,Tr,M)
#M = as.numeric(Arcs$parameters)
if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-prj.cent2edges(Tr,M)
}

#now we add the vertex names and annotation
txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]+c(-.02,.03,.03,.03,.05,-0.03,-.01)
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.05,-.06)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)
```

`plotCSarcs` *The plot of the arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for a 2D data set - multiple triangle case*

Description

Plots the arcs of Central Similarity Proximity Catch Digraph (CS-PCD) whose vertices are the data points in `Xp` in the multiple triangle case and the Delaunay triangles based on `Yp` points. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

CS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $t > 0$ and edge regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). Loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) more on the CS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotCSarcs(
  Xp,
  Yp,
  t,
  M = c(1, 1, 1),
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle, default for $M = (1, 1, 1)$ which is the center of mass of each triangle.
`asp`	A `numeric` value, giving the aspect ratio y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`"
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both)
`...`	Additional `plot` parameters.

Value

A plot of the arcs of the CS-PCD whose vertices are the points in data set `Xp` and the Delaunay triangles based on `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`plotCSarcs.tri`, `plotASarcs`, and `plotPEarcs`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)
t<-1.5  #try also t<-2

plotCSarcs(Xp,Yp,t,M,xlab="",ylab="")
```

`plotCSarcs.int` *The plot of the arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) for 1D data (vertices jittered along y-coordinate) - one interval case*

Description

Plots the arcs of CS-PCD whose vertices are the 1D points, `Xp`. CS proximity regions are constructed with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$ and the intervals are based on the interval `int=` (a, b) That is, data set `Xp` constitutes the vertices of the digraph and `int` determines the end points of the interval. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$) is added to the y-direction where `Jit` equals to the range of $\{$`Xp`, `int`$\}$ multiplied by `Jit` with default for $Jit = .1$). `center` is a logical argument, if `TRUE`, plot includes the center of the interval `int` as a vertical line in the plot, else center of the interval is not plotted.

Usage

```
plotCSarcs.int(
  Xp,
  int,
  t,
  c = 0.5,
  Jit = 0.1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  center = FALSE,
  ...
)
```

Arguments

`Xp`	A `vector` of 1D points constituting the vertices of the CS-PCD.
`int`	A `vector` of two 1D points constituting the end points of the interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center of the interval with the default `c=.5`. For the interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default=`0.1` and `Xp` points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of range of $\{$`Xp`, `int`$\}$ multiplied by `Jit`).
`main`	An overall title for the plot (default=`NULL`).
`xlab`, `ylab`	Titles of the x and y axes in the plot (default=`NULL` for both).
`xlim`, `ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`center`	A logical argument, if `TRUE`, plot includes the center of the interval `int` as a vertical line in the plot, else center of the interval is not plotted.
`...`	Additional `plot` parameters.

Value

A plot of the arcs of CS-PCD whose vertices are the 1D data set `Xp` in which vertices are jittered along y-axis for better visualization.

Author(s)

Elvan Ceyhan

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

`plotCSarcs1D` and `plotPEarcs.int`

Examples

```
tau<-2
c<-.4
a<-0; b<-10; int<-c(a,b)

#n is number of X points
n<-10;  #try also n<-20;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(n,a-xf,b+xf)

Xlim=range(Xp,int)
Ylim=3*c(-1,1)

jit<-.1
plotCSarcs.int(Xp,int,t=tau,c,jit,xlab="",ylab="",xlim=Xlim,ylim=Ylim)

set.seed(1)
plotCSarcs.int(Xp,int,t=1.5,c=.3,jit,xlab="",ylab="",center=TRUE)
set.seed(1)
plotCSarcs.int(Xp,int,t=2,c=.4,jit,xlab="",ylab="",center=TRUE)
```

`plotCSarcs.tri` *The plot of the arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for a 2D data set - one triangle case*

Description

Plots the arcs of CS-PCD whose vertices are the data points, `Xp` and the triangle `tri`. CS proximity regions are constructed with respect to the triangle `tri` with expansion parameter $t > 0$, i.e., arcs may exist only for `Xp` points inside the triangle `tri`. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

Edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
plotCSarcs.tri(
  Xp,
  tri,
  t,
  M = c(1, 1, 1),
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  edge.reg = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.
`asp`	A `numeric` value, giving the aspect ratio y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`".
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`edge.reg`	A logical argument to add edge regions to the plot, default is `edge.reg=FALSE`.
`...`	Additional `plot` parameters.

Value

A plot of the arcs of the CS-PCD whose vertices are the points in data set `Xp` and the triangle `tri`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). “Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering.” *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`plotCSarcs`, `plotPEarcs.tri` and `plotASarcs.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

t<-1.5  #try also t<-2

plotCSarcs.tri(Xp,Tr,t,M,main="Arcs of CS-PCD with t=1.5",
xlab="",ylab="",edge.reg = TRUE)

# or try the default center
#plotCSarcs.tri(Xp,Tr,t,main="Arcs of CS-PCD with t=1.5",xlab="",ylab="",edge.reg = TRUE);
#M=(arcsCStri(Xp,Tr,r)$param)$c #the part "M=(arcsPEtri(Xp,Tr,r)$param)$cent" is optional,
#for the below annotation of the plot

#can add vertex labels and text to the figure (with edge regions)
txt<-rbind(Tr,M)
xc<-txt[,1]+c(-.02,.02,.02,.03)
yc<-txt[,2]+c(.02,.02,.02,.03)
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)
```

`plotCSarcs1D` *The plot of the arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) for 1D data (vertices jittered along y-coordinate) - multiple interval case*

Description

Plots the arcs of CS-PCD whose vertices are the 1D points, `Xp`. CS proximity regions are constructed with expansion parameter $t > 0$ and centrality parameter $c \in (0,1)$ and the intervals are based on `Yp` points (i.e. the intervalization is based on `Yp` points). That is, data set `Xp` constitutes the vertices of the digraph and `Yp` determines the end points of the intervals. If there are duplicates of `Yp` or `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$) is added to the y-direction where `Jit` equals to the range of `Xp` and `Yp` multiplied by `Jit` with default for $Jit = .1$).

`centers` is a logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2016)).

Usage

```
plotCSarcs1D(
  Xp,
  Yp,
  t,
  c = 0.5,
  Jit = 0.1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

`Xp`	A `vector` of 1D points constituting the vertices of the CS-PCD.
`Yp`	A `vector` of 1D points constituting the end points of the intervals.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0,1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int`$=(a,b)$, the parameterized center is $M_c = a + c(b-a)$.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default=`0.1` and `Xp` points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of `Xp` and `Yp` multiplied by `Jit`).
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles of the x and y axes in the plot (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).

`centers`	A logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.
`...`	Additional `plot` parameters.

Value

A plot of the arcs of CS-PCD whose vertices are the 1D data set `Xp` in which vertices are jittered along y-axis for better visualization.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`plotPEarcs1D`

Examples

```
t<-1.5
c<-.4
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Xlim=range(Xp,Yp)
Ylim=c(-.2,.2)

jit<-.1

plotCSarcs1D(Xp,Yp,t,c,jit,xlab="",ylab="",xlim=Xlim,ylim=Ylim)

set.seed(1)
plotCSarcs1D(Xp,Yp,t=1.5,c=.3,jit,main="t=1.5, c=.3",xlab="",ylab="",centers=TRUE)
set.seed(1)
plotCSarcs1D(Xp,Yp,t=2,c=.3,jit,main="t=2, c=.3",xlab="",ylab="",centers=TRUE)
set.seed(1)
plotCSarcs1D(Xp,Yp,t=1.5,c=.5,jit,main="t=1.5, c=.5",xlab="",ylab="",centers=TRUE)
set.seed(1)
```

```
plotCSarcs1D(Xp,Yp,t=2,c=.5,jit,main="t=2, c=.5",xlab="",ylab="",centers=TRUE)
```

`plotCSregs` *The plot of the Central Similarity (CS) Proximity Regions for a 2D data set - multiple triangle case*

Description

Plots the points in and outside of the Delaunay triangles based on `Yp` points which partition the convex hull of `Yp` points and also plots the CS proximity regions for `Xp` points and the Delaunay triangles based on `Yp` points.

CS proximity regions are constructed with respect to the Delaunay triangles with the expansion parameter $t > 0$.

Edge regions in each triangle is based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle).

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) more on the CS proximity regions. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotCSregs(
  Xp,
  Yp,
  t,
  M = c(1, 1, 1),
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points for which CS proximity regions are constructed.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`.

`asp`	A `numeric` value, giving the aspect ratio y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`".
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`...`	Additional `plot` parameters.

Value

Plot of the `Xp` points, Delaunay triangles based on `Yp` and also the CS proximity regions for `Xp` points inside the convex hull of `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`plotCSregs.tri`, `plotASregs` and `plotPEregs`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
```

```
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)
tau<-1.5  #try also tau<-2

plotCSregs(Xp,Yp,tau,M,xlab="",ylab="")
```

`plotCSregs.int` *The plot of the Central Similarity (CS) Proximity Regions for a general interval (vertices jittered along y-coordinate) - one interval case*

Description

Plots the points in and outside of the interval `int` and also the CS proximity regions (which are also intervals). CS proximity regions are constructed with expansion parameter $t > 0$ and centrality parameter $c \in (0, 1)$.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default is $Jit = .1$) times range of proximity regions and `Xp`) is added to the y-direction. #' If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed. `center` is a logical argument, if `TRUE`, plot includes the center of the interval as a vertical line in the plot, else center of the interval is not plotted.

Usage

```
plotCSregs.int(
  Xp,
  int,
  t,
  c = 0.5,
  Jit = 0.1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  center = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 1D points for which CS proximity regions are to be constructed.
`int`	A `vector` of two real numbers representing an interval.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.

`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$ with the default `c=.5`. For the interval, `int`$=(a,b)$, the parameterized center is $M_c = a + c(b-a)$.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default=`0.1` and `Xp` points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of `Xp` and proximity region intervals multiplied by `Jit`).
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges.
`center`	A logical argument, if `TRUE`, plot includes the center of the interval as a vertical line in the plot, else center of the interval is not plotted.
`...`	Additional `plot` parameters.

Value

Plot of the CS proximity regions for 1D points in or outside the interval `int`

Author(s)

Elvan Ceyhan

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

`plotCSregs1D`, `plotCSregs`, and `plotPEregs.int`

Examples

```
c<-.4
tau<-2
a<-0; b<-10; int<-c(a,b)

n<-10
xf<-(int[2]-int[1])*.1

Xp<-runif(n,a-xf,b+xf)  #try also Xp<-runif(n,a-5,b+5)

plotCSregs.int(7,int,tau,c,xlab="x",ylab="")

plotCSregs.int(Xp,int,tau,c,xlab="x",ylab="")

plotCSregs.int(17,int,tau,c,xlab="x",ylab="")
plotCSregs.int(1,int,tau,c,xlab="x",ylab="")
plotCSregs.int(4,int,tau,c,xlab="x",ylab="")
```

```
plotCSregs.int(-7,int,tau,c,xlab="x",ylab="")
```

`plotCSregs.tri` *The plot of the Central Similarity (CS) Proximity Regions for a 2D data set - one triangle case*

Description

Plots the points in and outside of the triangle `tri` and also the CS proximity regions which are also triangular for points inside the triangle `tri` with edge regions are based on the center of mass CM.

CS proximity regions are defined with respect to the triangle `tri` with expansion parameter $t > 0$, so CS proximity regions are defined only for points inside the triangle `tri`.

Edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
plotCSregs.tri(
  Xp,
  tri,
  t,
  M = c(1, 1, 1),
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  edge.reg = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 2D points for which CS proximity regions are constructed.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e., the center of mass of `tri`.

`asp`	A `numeric` value, giving the aspect ratio y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`".
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`edge.reg`	A logical argument to add edge regions to the plot, default is `edge.reg=FALSE`.
`...`	Additional `plot` parameters.

Value

Plot of the CS proximity regions for points inside the triangle `tri` (and just the points outside `tri`)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`plotCSregs`, `plotASregs.tri` and `plotPEregs.tri`,

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp0<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

t<-.5  #try also t<-2

plotCSregs.tri(Xp0,Tr,t,M,main="Proximity Regions for CS-PCD", xlab="",ylab="")

Xp = Xp0[1,]
plotCSregs.tri(Xp,Tr,t,M,main="CS Proximity Regions with t=.5", xlab="",ylab="",edge.reg=TRUE)
```

```
# or try the default center
plotCSregs.tri(Xp,Tr,t,main="CS Proximity Regions with t=.5", xlab="",ylab="",edge.reg=TRUE);
#M=(arcsCStri(Xp,Tr,r)$param)$c #the part "M=(arcsPEtri(Xp,Tr,r)$param)$cent" is optional,
#for the below annotation of the plot

#can add vertex labels and text to the figure (with edge regions)
txt<-rbind(Tr,M)
xc<-txt[,1]+c(-.02,.02,.02,.02)
yc<-txt[,2]+c(.02,.02,.02,.03)
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)
```

`plotCSregs1D` *The plot of the Central Similarity (CS) Proximity Regions (vertices jittered along y-coordinate) - multiple interval case*

Description

Plots the points in and outside of the intervals based on `Yp` points and also the CS proximity regions (which are also intervals). If there are duplicates of `Yp` or `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

CS proximity region is constructed with expansion parameter $t > 0$ and centrality parameter $c \in (0,1)$. For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default is $Jit = .1$) times range of `Xp` and `Yp` and the proximity regions (intervals)) is added to the y-direction.

`centers` is a logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2016)).

Usage

```
plotCSregs1D(
  Xp,
  Yp,
  t,
  c = 0.5,
  Jit = 0.1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 1D points for which CS proximity regions are plotted.
`Yp`	A set of 1D points which constitute the end points of the intervals which partition the real line.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default=`0.1` and `Xp` points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of `Xp` and `Yp` and the proximity regions (intervals) multiplied by `Jit`).
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles of the x and y axes in the plot (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`centers`	A logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.
`...`	Additional `plot` parameters.

Value

Plot of the CS proximity regions for 1D points located in the middle or end-intervals based on `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`plotCSregs.int` and `plotPEregs1D`

Examples

```
t<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
```

```
set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

plotCSregs1D(Xp,Yp,t,c,xlab="",ylab="")

plotCSregs1D(Xp,Yp+10,t,c,xlab="",ylab="")
```

`plotDelaunay.tri` *The scatterplot of points from one class and plot of the Delaunay triangulation of the other class*

Description

Plots the scatter plot of `Xp` points together with the Delaunay triangles based on the `Yp` points. Both sets of points are of 2D.

See (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotDelaunay.tri(
  Xp,
  Yp,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points whose scatterplot is to be plotted.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both)
`...`	Additional `plot` parameters.

Value

A scatterplot of `Xp` points and the Delaunay triangulation of `Yp` points.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`plot.triSht` in `interp` package

Examples

```
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

plotDelaunay.tri(Xp,Yp,xlab="",ylab="",main="X points and Delaunay Triangulation of Y points")
```

`plotIntervals` *The plot of the subintervals based on* `Yp` *points together with* `Xp` *points*

Description

Plots the `Xp` points and the intervals based on `Yp` points. If there are duplicates of `Yp` points, only one point is retained for each duplicate value, and a warning message is printed.

Usage

```
plotIntervals(
  Xp,
  Yp,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 1D points whose scatter-plot is provided.
`Yp`	A set of 1D points which constitute the end points of the intervals which partition the real line.
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`...`	Additional `plot` parameters.

Value

Plot of the intervals based on `Yp` points and also scatter plot of `Xp` points

Author(s)

Elvan Ceyhan

See Also

`plotPEregs1D` and `plotDelaunay.tri`

Examples

```
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

plotIntervals(Xp,Yp,xlab="",ylab="")
```

`plotPEarcs` *The plot of the arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for a 2D data set - multiple triangle case*

Description

Plots the arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in `Xp` in the multiple triangle case and the Delaunay triangles based on `Yp` points. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). Loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)) for more on the PE-PCDs. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotPEarcs(
  Xp,
  Yp,
  r,
  M = c(1, 1, 1),
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this, argument should be set as `M="CC"`), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.
`asp`	A `numeric` value, giving the aspect ratio y/x (default is `NA`), see the official help page for `asp` by typing `"? asp"`.
`main`	An overall title for the plot (default=`NULL`).
`xlab`, `ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim`, `ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`...`	Additional `plot` parameters.

Value

A plot of the arcs of the PE-PCD whose vertices are the points in data set `Xp` and the Delaunay triangles based on `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`plotPEarcs.tri`, `plotASarcs`, and `plotCSarcs`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

r<-1.5  #try also r<-2

plotPEarcs(Xp,Yp,r,M,xlab="",ylab="")
```

`plotPEarcs.int` *The plot of the arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) for 1D data (vertices jittered along y-coordinate) - one interval case*

Description

Plots the arcs of PE-PCD whose vertices are the 1D points, `Xp`. PE proximity regions are constructed with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$ and the intervals are based on the interval `int`= (a, b) That is, data set `Xp` constitutes the vertices of the digraph and `int` determines the end points of the interval. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$) is added to the y-direction where `Jit` equals to the range of {`Xp`, `int`} multiplied by `Jit` with default for $Jit = .1$). `center` is a logical argument, if `TRUE`, plot includes the center of the interval `int` as a vertical line in the plot, else center of the interval is not plotted.

See also (Ceyhan (2012)).

Usage

```
plotPEarcs.int(
  Xp,
  int,
  r,
  c = 0.5,
  Jit = 0.1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
```

```
  xlim = NULL,
  ylim = NULL,
  center = FALSE,
  ...
)
```

Arguments

`Xp`	A `vector` of 1D points constituting the vertices of the PE-PCD.
`int`	A `vector` of two 1D points constituting the end points of the interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center of the interval with the default `c=.5`. For the interval, `int=` (a, b), the parameterized center is $M_c = a + c(b - a)$.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default=`0.1` and `Xp` points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of range of {`Xp`, `int`} multiplied by `Jit`).
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles of the x and y axes in the plot (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`center`	A logical argument, if `TRUE`, plot includes the center of the interval `int` as a vertical line in the plot, else center of the interval is not plotted.
`...`	Additional `plot` parameters.

Value

A plot of the arcs of PE-PCD whose vertices are the 1D data set `Xp` in which vertices are jittered along y-axis for better visualization.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`plotPEarcs1D` and `plotCSarcs.int`

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b)

#n is number of X points
n<-10;  #try also n<-20;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(n,a-xf,b+xf)

Xlim=range(Xp,int)
Ylim=.1*c(-1,1)

jit<-.1
set.seed(1)
plotPEarcs.int(Xp,int,r=1.5,c=.3,jit,xlab="",ylab="",center=TRUE)
set.seed(1)
plotPEarcs.int(Xp,int,r=2,c=.3,jit,xlab="",ylab="",center=TRUE)
```

`plotPEarcs.tri` *The plot of the arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for a 2D data set - one triangle case*

Description

Plots the arcs of PE-PCD whose vertices are the data points, `Xp` and the triangle `tri`. PE proximity regions are constructed with respect to the triangle `tri` with expansion parameter $r \geq 1$, i.e., arcs may exist only for `Xp` points inside the triangle `tri`. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`. When the center is the circumcenter, `CC`, the vertex regions are constructed based on the orthogonal projections to the edges, while with any interior center `M`, the vertex regions are constructed using the extensions of the lines combining vertices with `M`. `M`-vertex regions are recommended spatial inference, due to geometry invariance property of the arc density and domination number the PE-PCDs based on uniform data.

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
plotPEarcs.tri(
  Xp,
  tri,
```

```
  r,
  M = c(1, 1, 1),
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  vert.reg = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.
`asp`	A `numeric` value, giving the aspect ratio y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`".
`main`	An overall title for the plot (default=`NULL`).
`xlab`, `ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim`, `ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`vert.reg`	A logical argument to add vertex regions to the plot, default is `vert.reg=FALSE`.
`...`	Additional `plot` parameters.

Value

A plot of the arcs of the PE-PCD whose vertices are the points in data set `Xp` and the triangle `tri`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`plotASarcs.tri`, `plotCSarcs.tri`, and `plotPEarcs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)
#try also M<-c(1.6,1.0) or M<-circumcenter.tri(Tr)
r<-1.5  #try also r<-2
plotPEarcs.tri(Xp,Tr,r,M,main="Arcs of PE-PCD with r = 1.5",
xlab="",ylab="",vert.reg = TRUE)

# or try the default center
#plotPEarcs.tri(Xp,Tr,r,main="Arcs of PE-PCD with r = 1.5",
#xlab="",ylab="",vert.reg = TRUE);
#M=(arcsPEtri(Xp,Tr,r)$param)$cent
#the part "M=(arcsPEtri(Xp,Tr,r)$param)$cent" is optional,
#for the below annotation of the plot

#can add vertex labels and text to the figure (with vertex regions)
ifelse(isTRUE(all.equal(M,circumcenter.tri(Tr))),
{Ds<-rbind((B+C)/2,(A+C)/2,(A+B)/2); cent.name="CC"},
{Ds<-prj.cent2edges(Tr,M); cent.name="M"})

txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.02,.02,.02,.04,-0.03,-.01)
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.04,-.06)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)
```

`plotPEarcs1D` *The plot of the arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) for 1D data (vertices jittered along y-coordinate) - multiple interval case*

Description

Plots the arcs of PE-PCD whose vertices are the 1D points, `Xp`. PE proximity regions are constructed with expansion parameter $r \geq 1$ and centrality parameter $c \in (0,1)$ and the intervals are based on `Yp` points (i.e. the intervalization is based on `Yp` points). That is, data set `Xp` constitutes the vertices of the digraph and `Yp` determines the end points of the intervals. If there are duplicates of `Yp` or `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$) is added to the y-direction where `Jit` equals to the range of `Xp` and `Yp` multiplied by `Jit` with default for $Jit = .1$). `centers` is a logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2012)).

Usage

```
plotPEarcs1D(
  Xp,
  Yp,
  r,
  c = 0.5,
  Jit = 0.1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

`Xp`	A `vector` of 1D points constituting the vertices of the PE-PCD.
`Yp`	A `vector` of 1D points constituting the end points of the intervals.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0,1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, (a,b), the parameterized center is $M_c = a + c(b-a)$.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default=`0.1` and `Xp` points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of the union of `Xp` and `Yp` points multiplied by `Jit`).
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles of the x and y axes in the plot (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).

`centers`	A logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.
`...`	Additional `plot` parameters.

Value

A plot of the arcs of PE-PCD whose vertices are the 1D data set `Xp` in which vertices are jittered along y-axis for better visualization.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`plotPEarcs.int` and `plotCSarcs1D`

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Xlim=range(Xp,Yp)
Ylim=.1*c(-1,1)

jit<-.1

set.seed(1)
plotPEarcs1D(Xp,Yp,r=1.5,c=.3,jit,xlab="",ylab="",centers=TRUE)
set.seed(1)
plotPEarcs1D(Xp,Yp,r=2,c=.3,jit,xlab="",ylab="",centers=TRUE)
```

`plotPEregs` *The plot of the Proportional Edge (PE) Proximity Regions for a 2D data set - multiple triangle case*

Description

Plots the points in and outside of the Delaunay triangles based on `Yp` points which partition the convex hull of `Yp` points and also plots the PE proximity regions for `Xp` points and the Delaunay triangles based on `Yp` points.

PE proximity regions are constructed with respect to the Delaunay triangles with the expansion parameter $r \geq 1$.

Vertex regions in each triangle is based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle).

See (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)) for more on the PE proximity regions. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotPEregs(
  Xp,
  Yp,
  r,
  M = c(1, 1, 1),
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points for which PE proximity regions are constructed.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this, argument should be set as `M="CC"`), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

`asp`	A `numeric` value, giving the aspect ratio y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`".
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both)
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`...`	Additional `plot` parameters.

Value

Plot of the `Xp` points, Delaunay triangles based on `Yp` points and also the PE proximity regions for `Xp` points inside the convex hull of `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`plotPEregs.tri`, `plotASregs`, and `plotCSregs`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
```

```
Yp<-cbind(runif(ny,0,.25),
runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)
r<-1.5  #try also r<-2

plotPEregs(Xp,Yp,r,M,xlab="",ylab="")
```

`plotPEregs.int` *The plot of the Proportional Edge (PE) Proximity Regions for a general interval (vertices jittered along y-coordinate) - one interval case*

Description

Plots the points in and outside of the interval `int` and also the PE proximity regions (which are also intervals). PE proximity regions are constructed with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default is $Jit = .1$) times range of proximity regions and `Xp`) is added to the y-direction. If there are duplicates of `Xp` points, only one point is retained for each duplicate value, and a warning message is printed. `center` is a logical argument, if `TRUE`, plot includes the center of the interval as a vertical line in the plot, else center of the interval is not plotted.

See also (Ceyhan (2012)).

Usage

```
plotPEregs.int(
  Xp,
  int,
  r,
  c = 0.5,
  Jit = 0.1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  center = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 1D points for which PE proximity regions are to be constructed.
`int`	A `vector` of two real numbers representing an interval.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$ with the default `c=.5`. For the interval, `int`$=(a,b)$, the parameterized center is $M_c = a + c(b-a)$.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default=`0.1` and `Xp` points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of the union of `Xp` and `Yp` points multiplied by `Jit`).
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges.
`center`	A logical argument, if `TRUE`, plot includes the center of the interval as a vertical line in the plot, else center of the interval is not plotted.
`...`	Additional `plot` parameters.

Value

Plot of the PE proximity regions for 1D points in or outside the interval `int`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`plotPEregs1D`, `plotCSregs.int`, and `plotCSregs.int`

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

n<-10
xf<-(int[2]-int[1])*.1
Xp<-runif(n,a-xf,b+xf)  #try also Xp<-runif(n,a-5,b+5)
plotPEregs.int(Xp,int,r,c,xlab="x",ylab="")
```

```
plotPEregs.int(7,int,r,c,xlab="x",ylab="")
```

`plotPEregs.std.tetra` *The plot of the Proportional Edge (PE) Proximity Regions for a 3D data set - standard regular tetrahedron case*

Description

Plots the points in and outside of the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}$ and also the PE proximity regions for points in data set `Xp`.

PE proximity regions are defined with respect to the standard regular tetrahedron T_h with expansion parameter $r \geq 1$, so PE proximity regions are defined only for points inside T_h.

Vertex regions are based on circumcenter (which is equivalent to the center of mass for the standard regular tetrahedron) of T_h.

See also (Ceyhan (2005, 2010)).

Usage

```
plotPEregs.std.tetra(
  Xp,
  r,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 3D points for which PE proximity regions are constructed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab, zlab`	titles for the x, y, and z axes, respectively (default=`NULL` for all).
`xlim, ylim, zlim`	Two `numeric` vectors of length 2, giving the x-, y-, and z-coordinate ranges (default=`NULL` for all).
`...`	Additional `scatter3D` parameters.

Value

Plot of the PE proximity regions for points inside the standard regular tetrahedron T_h (and just the points outside T_h)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`plotPEregs`, `plotASregs.tri`, `plotASregs`, `plotCSregs.tri`, and `plotCSregs`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
r<-1.5

n<-3  #try also n<-20
Xp<-runif.std.tetra(n)$g  #try also Xp[,1]<-Xp[,1]+1

plotPEregs.std.tetra(Xp[1:3,],r)

P1<-c(.1,.1,.1)
plotPEregs.std.tetra(rbind(P1,P1),r)
```

`plotPEregs.tetra` *The plot of the Proportional Edge (PE) Proximity Regions for a 3D data set - one tetrahedron case*

Description

Plots the points in and outside of the tetrahedron `th` and also the PE proximity regions (which are also tetrahedrons) for points inside the tetrahedron `th`.

PE proximity regions are constructed with respect to tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions are based on the center `M` which is circumcenter (`"CC"`) or center of mass (`"CM"`) of `th` with default=`"CM"`, so PE proximity regions are defined only for points inside the tetrahedron `th`.

See also (Ceyhan (2005, 2010)).

Usage

```
plotPEregs.tetra(
  Xp,
  th,
  r,
  M = "CM",
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 3D points for which PE proximity regions are constructed.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes `"CC"` for circumcenter and `"CM"` for center of mass; default=`"CM"`.
`main`	An overall title for the plot (default=`NULL`).
`xlab`, `ylab`, `zlab`	Titles for the x, y, and z axes, respectively (default=`NULL` for all).
`xlim`, `ylim`, `zlim`	Two `numeric` vectors of length 2, giving the x-, y-, and z-coordinate ranges (default=`NULL` for all).
`...`	Additional `scatter3D` parameters.

Value

Plot of the PE proximity regions for points inside the tetrahedron `th` (and just the points outside `th`)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`plotPEregs.std.tetra`, `plotPEregs.tri` and `plotPEregs.int`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
set.seed(1)
tetra<-rbind(A,B,C,D)+matrix(runif(12,-.25,.25),ncol=3) #adding jitter to make it non-regular

n<-5  #try also n<-20
Xp<-runif.tetra(n,tetra)$g  #try also Xp[,1]<-Xp[,1]+1

M<-"CM"  #try also M<-"CC"
r<-1.5

plotPEregs.tetra(Xp,tetra,r) #uses the default M="CM"
plotPEregs.tetra(Xp,tetra,r,M="CC")

plotPEregs.tetra(Xp[1,],tetra,r)  #uses the default M="CM"
plotPEregs.tetra(Xp[1,],tetra,r,M)
```

`plotPEregs.tri` *The plot of the Proportional Edge (PE) Proximity Regions for a 2D data set - one triangle case*

Description

Plots the points in and outside of the triangle `tri` and also the PE proximity regions for points in data set `Xp`.

PE proximity regions are defined with respect to the triangle `tri` with expansion parameter $r \geq 1$, so PE proximity regions are defined only for points inside the triangle `tri`.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`. When the center is the circumcenter, `CC`, the vertex regions are constructed based on the orthogonal projections to the edges, while with any interior center `M`, the vertex regions are constructed using the extensions of the lines combining vertices with `M`. `M`-vertex regions are recommended spatial inference, due to geometry invariance property of the arc density and domination number the PE-PCDs based on uniform data.

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
plotPEregs.tri(
  Xp,
  tri,
  r,
  M = c(1, 1, 1),
  asp = NA,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  vert.reg = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 2D points for which PE proximity regions are constructed.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri` which may be entered as "CC" as well; default is $M = (1, 1, 1)$, i.e., the center of mass of `tri`.
`asp`	A `numeric` value, giving the aspect ratio y/x (default is `NA`), see the official help page for `asp` by typing "`? asp`".
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`vert.reg`	A logical argument to add vertex regions to the plot, default is `vert.reg=FALSE`.
`...`	Additional `plot` parameters.

Value

Plot of the PE proximity regions for points inside the triangle `tri` (and just the points outside `tri`)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`plotPEregs`, `plotASregs.tri`, and `plotCSregs.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
Xp0<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)
#try also M<-c(1.6,1.0) or M = circumcenter.tri(Tr)
r<-1.5  #try also r<-2

plotPEregs.tri(Xp0,Tr,r,M)
Xp = Xp0[1,]
plotPEregs.tri(Xp,Tr,r,M)

plotPEregs.tri(Xp,Tr,r,M,
main="PE Proximity Regions with r = 1.5",
xlab="",ylab="",vert.reg = TRUE)

# or try the default center
#plotPEregs.tri(Xp,Tr,r,main="PE Proximity Regions with r = 1.5",xlab="",ylab="",vert.reg = TRUE);
#M=(arcsPEtri(Xp,Tr,r)$param)$c
#the part "M=(arcsPEtri(Xp,Tr,r)$param)$cent" is optional,
#for the below annotation of the plot

#can add vertex labels and text to the figure (with vertex regions)
ifelse(isTRUE(all.equal(M,circumcenter.tri(Tr))),
       {Ds<-rbind((B+C)/2,(A+C)/2,(A+B)/2); cent.name="CC"},
       {Ds<-prj.cent2edges(Tr,M); cent.name<-"M"})

txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.02,.02,.02,.03,-0.03,-.01)
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.05,-.06)
```

```
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)
```

---

plotPEregs1D *The plot of the Proportional Edge (PE) Proximity Regions (vertices jittered along y-coordinate) - multiple interval case*

Description

Plots the points in and outside of the intervals based on `Yp` points and also the PE proximity regions (i.e., intervals). PE proximity region is constructed with expansion parameter $r \geq 1$ and centrality parameter $c \in (0, 1)$. If there are duplicates of `Yp` or `Xp` points, only one point is retained for each duplicate value, and a warning message is printed.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default is $Jit = .1$) times range of `Xp` and `Yp` and the proximity regions (intervals)) is added to the y-direction.

`centers` is a logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2012)).

Usage

```
plotPEregs1D(
  Xp,
  Yp,
  r,
  c = 0.5,
  Jit = 0.1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

`Xp`	A set of 1D points for which PE proximity regions are plotted.
`Yp`	A set of 1D points which constitute the end points of the intervals which partition the real line.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

`c`	A positive real number in $(0,1)$ parameterizing the center inside middle intervals with the default `c=.5`. For the interval, (a,b), the parameterized center is $M_c = a + c(b-a)$.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default=`0.1` and `Xp` points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of the union of `Xp` and `Yp` points multiplied by `Jit`).
`main`	An overall title for the plot (default=`NULL`).
`xlab, ylab`	Titles for the x and y axes, respectively (default=`NULL` for both).
`xlim, ylim`	Two `numeric` vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`centers`	A logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted (default is `FALSE`).
`...`	Additional `plot` parameters.

Value

Plot of the PE proximity regions for 1D points located in the middle or end-intervals based on `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

See Also

`plotPEregs1D`, `plotCSregs.int`, and `plotCSregs1D`

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b);

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-15; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xf<-(int[2]-int[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

plotPEregs1D(Xp,Yp,r,c,xlab="x",ylab="")
```

`print.Extrema` *Print a* `Extrema` `object`

Description

Prints the `call` of the `object` of class `"Extrema"` and also the `type` (i.e. a brief description) of the extrema).

Usage

```
## S3 method for class 'Extrema'
print(x, ...)
```

Arguments

`x`	A `Extrema` `object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"Extrema"` and also the `type` (i.e. a brief description) of the extrema).

See Also

`summary.Extrema`, `print.summary.Extrema`, and `plot.Extrema`

Examples

```
n<-10
Xp<-runif.std.tri(n)$gen.points
Ext<-cl2edges.std.tri(Xp)
Ext
print(Ext)

typeof(Ext)
attributes(Ext)
```

`print.Lines` *Print a* `Lines` *object*

Description

Prints the `call` of the `object` of class `"Lines"` and also the `coefficients` of the line (in the form: `y` = `slope` * `x` + `intercept`).

Usage

```
## S3 method for class 'Lines'
print(x, ...)
```

Arguments

`x`	A `Lines` `object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"Lines"` and the `coefficients` of the line (in the form: `y` = `slope` * `x` + `intercept`).

See Also

`summary.Lines`, `print.summary.Lines`, and `plot.Lines`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)
xr<-range(A,B);
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=3) #try also l=10, 20 or 100

lnAB<-Line(A,B,x)
lnAB
print(lnAB)

typeof(lnAB)
attributes(lnAB)
```

`print.Lines3D` *Print a* `Lines3D` `object`

Description

Prints the `call` of the `object` of class `"Lines3D"`, the `coefficients` of the line (in the form: `x=x0 + A*t`, `y=y0 + B*t`, and `z=z0 + C*t`), and the initial point together with the direction vector.

Usage

```
## S3 method for class 'Lines3D'
print(x, ...)
```

Arguments

`x`	A `Lines3D` `object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"Lines3D"`, the `coefficients` of the line (in the form: `x=x0 + A*t`, `y=y0 + B*t`, and `z=z0 + C*t`), and the initial point together with the direction vector.

See Also

`summary.Lines3D`, `print.summary.Lines3D`, and `plot.Lines3D`

Examples

```
P<-c(1,10,3); Q<-c(1,1,3);
vecs<-rbind(P,Q)
Line3D(P,Q,.1)
Line3D(P,Q,.1,dir.vec=FALSE)

tr<-range(vecs);
tf<-(tr[2]-tr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=3) #try also l=10, 20 or 100

lnPQ3D<-Line3D(P,Q,tsq)
lnPQ3D
print(lnPQ3D)

typeof(lnPQ3D)
attributes(lnPQ3D)
```

`print.NumArcs` *Print a* `NumArcs` `object`

Description

Prints the `call` of the `object` of class `"NumArcs"` and also the `desc` (i.e. a brief description) of the output.

Usage

```
## S3 method for class 'NumArcs'
print(x, ...)
```

Arguments

`x`	A `NumArcs` `object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"NumArcs"` and also the `desc` (i.e. a brief description) of the output: number of arcs in the proximity catch digraph (PCD) and related quantities in the induced subdigraphs for points in the Delaunay cells.

See Also

`summary.NumArcs`, `print.summary.NumArcs`, and `plot.NumArcs`

Examples

```
nx<-15; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-"CC"  #try also M<-c(1,1,1)

Narcs<-num.arcsAS(Xp,Yp,M)
Narcs
print(Narcs)

typeof(Narcs)
attributes(Narcs)
```

`print.Patterns` *Print a* `Patterns` object

Description

Prints the `call` of the `object` of class `"Patterns"` and also the `type` (or description) of the pattern).

Usage

```
## S3 method for class 'Patterns'
print(x, ...)
```

Arguments

`x`	A `Patterns object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"Patterns"` and also the `type` (or description) of the pattern).

See Also

`summary.Patterns`, `print.summary.Patterns`, and `plot.Patterns`

Examples

```
nx<-10; #try also 20, 100, and 1000
ny<-5; #try also 1
e<-.15;
Y<-cbind(runif(ny),runif(ny))
#with default bounding box (i.e., unit square)

Xdt<-rseg.circular(nx,Y,e)
Xdt
print(Xdt)

typeof(Xdt)
attributes(Xdt)
```

`print.PCDs` *Print a* `PCDs object`

Description

Prints the `call` of the `object` of class `"PCDs"` and also the `type` (i.e. a brief description) of the proximity catch digraph (PCD).

Usage

```
## S3 method for class 'PCDs'
print(x, ...)
```

Arguments

`x`	A `PCDs object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"PCDs"` and also the `type` (i.e. a brief description) of the proximity catch digraph (PCD).

See Also

`summary.PCDs`, `print.summary.PCDs`, and `plot.PCDs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10
Xp<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)
Arcs<-arcsAStri(Xp,Tr,M)
Arcs
print(Arcs)

typeof(Arcs)
attributes(Arcs)
```

`print.Planes` *Print a* `Planes object`

Description

Prints the `call` of the `object` of class `"Planes"` and also the `coefficients` of the plane (in the form: `z = A*x + B*y + C`).

Usage

```
## S3 method for class 'Planes'
print(x, ...)
```

Arguments

`x`	A `Planes object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"Planes"` and the `coefficients` of the plane (in the form: `z = A*x + B*y + C`).

See Also

`summary.Planes`, `print.summary.Planes`, and `plot.Planes`

Examples

```
P<-c(1,10,3); Q<-c(1,1,3); C<-c(3,9,12)
pts<-rbind(P,Q,C)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1
#how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5) #try also l=10, 20 or 100
y<-seq(yr[1]-yf,yr[2]+yf,l=5) #try also l=10, 20 or 100

plPQC<-Plane(P,Q,C,x,y)
plPQC
print(plPQC)

typeof(plPQC)
attributes(plPQC)
```

`print.summary.Extrema` *Print a summary of a* `Extrema` object

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.Extrema'
print(x, ...)
```

Arguments

`x`	An `object` of class "`summary.Extrema`", generated by `summary.Extrema`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.Extrema`, `summary.Extrema`, and `plot.Extrema`

`print.summary.Lines` *Print a summary of a* `Lines` object

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.Lines'
print(x, ...)
```

Arguments

`x`	An `object` of class "`summary.Lines`", generated by `summary.Lines`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.Lines`, `summary.Lines`, and `plot.Lines`

`print.summary.Lines3D` *Print a summary of a* `Lines3D` object

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.Lines3D'
print(x, ...)
```

Arguments

`x`	An object of class "`summary.Lines3D`", generated by `summary.Lines3D`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.Lines3D`, `summary.Lines3D`, and `plot.Lines3D`

`print.summary.NumArcs` *Print a summary of a* `NumArcs` object

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.NumArcs'
print(x, ...)
```

Arguments

`x`	An object of class "`summary.NumArcs`", generated by `summary.NumArcs`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.NumArcs`, `summary.NumArcs`, and `plot.NumArcs`

`print.summary.Patterns` *Print a summary of a* `Patterns` object

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.Patterns'
print(x, ...)
```

Arguments

`x`	An object of class "`summary.Patterns`", generated by `summary.Patterns`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.Patterns`, `summary.Patterns`, and `plot.Patterns`

`print.summary.PCDs` *Print a summary of a* `PCDs` object

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.PCDs'
print(x, ...)
```

Arguments

`x`	An `object` of class `"summary.PCDs"`, generated by `summary.PCDs`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.PCDs`, `summary.PCDs`, and `plot.PCDs`

`print.summary.Planes` *Print a summary of a* `Planes` `object`

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.Planes'
print(x, ...)
```

Arguments

`x`	An `object` of class `"summary.Planes"`, generated by `summary.Planes`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.Planes`, `summary.Planes`, and `plot.Planes`

`print.summary.TriLines` *Print a summary of a* `TriLines` object

Description

Prints some information about the `object`

Usage

```
## S3 method for class 'summary.TriLines'
print(x, ...)
```

Arguments

`x`	An `object` of class `"summary.TriLines"`, generated by `summary.TriLines`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.TriLines`, `summary.TriLines`, and `plot.TriLines`

`print.summary.Uniform` *Print a summary of a* `Uniform` object

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.Uniform'
print(x, ...)
```

Arguments

`x`	An `object` of class `"summary.Uniform"`, generated by `summary.Uniform`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.Uniform`, `summary.Uniform`, and `plot.Uniform`

---

print.TriLines *Print a* `TriLines object`

Description

Prints the `call` of the `object` of class `"TriLines"` and also the `coefficients` of the line (in the form: `y` = `slope` * `x` + `intercept`), and the vertices of the triangle with respect to which the line is defined.

Usage

```
## S3 method for class 'TriLines'
print(x, ...)
```

Arguments

`x`	A `TriLines object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"TriLines"` , the `coefficients` of the line (in the form: `y` = `slope` * `x` + `intercept`), and the vertices of the triangle with respect to which the line is defined.

See Also

`summary.TriLines`, `print.summary.TriLines`, and `plot.TriLines`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,l=3)

lnACM<-lineA2CMinTe(x)
lnACM
print(lnACM)

typeof(lnACM)
attributes(lnACM)
```

`print.Uniform` *Print a* `Uniform` object

Description

Prints the `call` of the `object` of class `"Uniform"` and also the `type` (i.e. a brief description) of the uniform distribution).

Usage

```
## S3 method for class 'Uniform'
print(x, ...)
```

Arguments

`x`	A `Uniform` `object`.
`...`	Additional arguments for the S3 method `'print'`.

Value

The `call` of the `object` of class `"Uniform"` and also the `type` (i.e. a brief description) of the uniform distribution).

See Also

`summary.Uniform`, `print.summary.Uniform`, and `plot.Uniform`

Examples

```
n<-10 #try also 20, 100, and 1000
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)

Xdt<-runif.tri(n,Tr)
Xdt
print(Xdt)

typeof(Xdt)
attributes(Xdt)
```

`prj.cent2edges` *Projections of a point inside a triangle to its edges*

Description

Returns the projections from a general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a triangle to the edges on the extension of the lines joining `M` to the vertices (see the examples for an illustration).

See also (Ceyhan (2005, 2010)).

Usage

```
prj.cent2edges(tri, M)
```

Arguments

`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.

Value

Three projection points (stacked row-wise) from a general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a triangle to the edges on the extension of the lines joining `M` to the vertices; row i is the projection point into edge i, for $i = 1, 2, 3$.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`prj.cent2edges.basic.tri` and `prj.nondegPEcent2edges`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

Ds<-prj.cent2edges(Tr,M)  #try also prj.cent2edges(Tr,M=c(1,1))
Ds

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",
main="Projection of Center M on the edges of a triangle",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.04,-.04,-.02)
yc<-txt[,2]+c(-.02,.04,.04,-.06)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
```

`prj.cent2edges.basic.tri`

Projections of a point inside the standard basic triangle form to its edges

---

Description

Returns the projections from a general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle form $T_b = T((0,0),(1,0),(c_1,c_2))$ to the edges on the extension of the lines joining `M` to the vertices (see the examples for an illustration). In the standard basic triangle form T_b, c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the standard basic triangle form by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle form is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
prj.cent2edges.basic.tri(c1, c2, M)
```

Arguments

`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle form adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle form.

Value

Three projection points (stacked row-wise) from a general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a standard basic triangle form to the edges on the extension of the lines joining `M` to the vertices; row i is the projection point into edge i, for $i = 1, 2, 3$.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`prj.cent2edges` and `prj.nondegPEcent2edges`

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.2)
```

```
Ds<-prj.cent2edges.basic.tri(c1,c2,M)
Ds

Xlim<-range(Tb[,1])
Ylim<-range(Tb[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tb)}
#need to run this when M is given in barycentric coordinates

plot(Tb,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.1,.1),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
L<-rbind(M,M,M); R<-Tb
segments(L[,1], L[,2], R[,1], R[,2], lty = 3,col=2)

xc<-Tb[,1]+c(-.04,.05,.04)
yc<-Tb[,2]+c(.02,.02,.03)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.03,-.03,0)
yc<-txt[,2]+c(-.02,.02,.02,-.03)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
```

`prj.nondegPEcent2edges` *Projections of Centers for non-degenerate asymptotic distribution of domination number of Proportional Edge Proximity Catch Digraphs (PE-PCDs) to its edges*

Description

Returns the projections from center `cent` to the edges on the extension of the lines joining `cent` to the vertices in the triangle, `tri`. Here M is one of the three centers which gives nondegenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for a given expansion parameter `r` in $(1, 1.5]$. The center label `cent` values `1,2,3` correspond to the vertices M_1, M_2, and M_3 (i.e., row numbers in the output of `center.nondegPE(tri,r)`); default for `cent` is 1. `cent` becomes center of mass CM for $r = 1.5$.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011)).

Usage

```
prj.nondegPEcent2edges(tri, r, cent = 1)
```

Arguments

`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ for this function.
`cent`	Index of the center (as $1, 2, 3$ corresponding to M_1, M_2, M_3) which gives nondegenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for expansion parameter `r` in $(1, 1.5]$; default `cent=1`.

Value

Three projection points (stacked row-wise) from one of the centers (as $1, 2, 3$ corresponding to M_1, M_2, M_3) which gives nondegenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for expansion parameter `r` in $(1, 1.5]$.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`prj.cent2edges.basic.tri` and `prj.cent2edges`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
r<-1.35

prj.nondegPEcent2edges(Tr,r,cent=2)

Ms<-center.nondegPE(Tr,r)
M1=Ms[1,]
```

```
Ds<-prj.nondegPEcent2edges(Tr,r,cent=1)

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",
main="Projections from a non-degeneracy center\n to the edges of the triangle",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Ms,pch=".",col=1)
polygon(Ms,lty = 2)

xc<-Tr[,1]+c(-.02,.03,.02)
yc<-Tr[,2]+c(-.02,.04,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-Ms
xc<-txt[,1]+c(-.02,.04,-.04)
yc<-txt[,2]+c(-.02,.04,.04)
txt.str<-c("M1","M2","M3")
text(xc,yc,txt.str)

points(Ds,pch=4,col=2)
L<-rbind(M1,M1,M1); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2,lwd=2,col=4)
txt<-Ds
xc<-txt[,1]+c(-.02,.04,-.04)
yc<-txt[,2]+c(-.02,.04,.04)
txt.str<-c("D1","D2","D3")
text(xc,yc,txt.str)

prj.nondegPEcent2edges(Tr,r,cent=3)
#gives an error message if center index, cent, is different from 1, 2 or 3
prj.nondegPEcent2edges(Tr,r=1.49,cent=2)
#gives an error message if r>1.5
```

`radii` *The radii of points from one class with respect to points from the other class*

Description

Returns the radii of the balls centered at x points where radius of an x point equals to the minimum distance to y points (i.e., distance to the closest y point). That is, for each x point $radius = \min_{y \in Y}(d(x,y))$. x and y points must be of the same dimension.

Usage

```
radii(x, y)
```

Arguments

`x`	A set of d-dimensional points for which the radii are computed. Radius of an `x` point equals to the distance to the closest `y` point.
`y`	A set of d-dimensional points representing the reference points for the balls. That is, radius of an `x` point is defined as the minimum distance to the `y` points.

Value

A `list` with three elements

`rad`	A `vector` whose entries are the radius values for the `x` points. Radius of an `x` point equals to the distance to the closest `y` point
`index.of.clYp`	A `vector` of indices of the closest `y` points to the `x` points. The i-th entry in this `vector` is the index of the closest `y` point to i-th `x` point.
`closest.Yp`	A `vector` of the closest `y` points to the `x` points. The i-th entry in this `vector` or i-th row in the matrix is the closest `y` point to i-th `x` point.

Author(s)

Elvan Ceyhan

See Also

`radius`

Examples

```
nx<-10
ny<-5
X<-cbind(runif(nx),runif(nx))
Y<-cbind(runif(ny),runif(ny))
Rad<-radii(X,Y)
Rad
rd<-Rad$rad

Xlim<-range(X[,1]-rd,X[,1]+rd,Y[,1])
Ylim<-range(X[,2]-rd,X[,2]+rd,Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(Y),asp=1,pch=16,col=2,xlab="",ylab="",
main="Circles Centered at Class X Points with \n Radius Equal to the Distance to Closest Y Point",
axes=TRUE, xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(rbind(X))
interp::circles(X[,1],X[,2],Rad$rad,lty=1,lwd=1,col=4)
```

```
#For 1D data
nx<-10
ny<-5
Xm<-as.matrix(X)
Ym<-as.matrix(Y)
radii(Xm,Ym)  #this works as Xm and Ym are treated as 1D data sets
#but will give error if radii(X,Y) is used
#as X and Y are treated as vectors (i.e., points)

#For 3D data
nx<-10
ny<-5
X<-cbind(runif(nx),runif(nx),runif(nx))
Y<-cbind(runif(ny),runif(ny),runif(ny))
radii(X,Y)
```

`radius` *The radius of a point from one class with respect to points from the other class*

Description

Returns the radius for the ball centered at point `p` with radius=min distance to `Y` points. That is, for the point `p` $radius = \min_{y \in Y} d(p, y)$ (i.e., distance from `p` to the closest `Y` point). The point `p` and `Y` points must be of same dimension.

Usage

```
radius(p, Y)
```

Arguments

`p`	A d-dimensional point for which radius is computed. Radius of `p` equals to the distance to the closest `Y` point to `p`.
`Y`	A set of d-dimensional points representing the reference points for the balls. That is, radius of the point `p` is defined as the minimum distance to the `Y` points.

Value

A `list` with three elements

`rad`	Radius value for the point, `p` defined as $\min_{y in Y} d(p, y)$
`index.of.clYpnt`	Index of the closest `Y` points to the point `p`
`closest.Ypnt`	The closest `Y` point to the point `p`

Author(s)

Elvan Ceyhan

See Also

`radii`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

ny<-10
Y<-cbind(runif(ny),runif(ny))
radius(A,Y)

nx<-10
X<-cbind(runif(nx),runif(nx))
rad<-rep(0,nx)
for (i in 1:nx)
rad[i]<-radius(X[i,],Y)$rad

Xlim<-range(X[,1]-rad,X[,1]+rad,Y[,1])
Ylim<-range(X[,2]-rad,X[,2]+rad,Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(Y),asp=1,pch=16,col=2,xlab="",ylab="",
main="Circles Centered at Class X Points with \n Radius Equal to the Distance to Closest Y Point",
axes=TRUE, xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(rbind(X))
interp::circles(X[,1],X[,2],rad,lty=1,lwd=1,col=4)

#For 1D data
ny<-5
Y<-runif(ny)
Ym = as.matrix(Y)
radius(1,Ym)  #this works as Y is treated as 1D data sets
#but will give error if radius(1,Y) is used
#as Y is treated as a vector (i.e., points)

#For 3D data
ny<-5
X<-runif(3)
Y<-cbind(runif(ny),runif(ny),runif(ny))
radius(X,Y)
```

`rassoc.circular` *Generation of points associated (in a radial or circular fashion) with a given set of points*

Description

An object of class `"Patterns"`. Generates `n` 2D points uniformly in $(a_1 - e, a_1 + e) \times (a_1 - e, a_1 + e) \cap U_i B(y_i, e)$ (a_1 and $b1$ are denoted as `a1` and `b1` as arguments) where $Y_p = (y_1, y_2, \ldots, y_{n_y})$ with n_y being number of `Yp` points for various values of `e` under the association pattern and $B(y_i, e)$ is the ball centered at y_i with radius `e`.

`e` must be positive and very large values of `e` provide patterns close to CSR. `a1` is defaulted to the minimum of the x-coordinates of the `Yp` points, `a2` is defaulted to the maximum of the x-coordinates of the `Yp` points, `b1` is defaulted to the minimum of the y-coordinates of the `Yp` points, `b2` is defaulted to the maximum of the y-coordinates of the `Yp` points. This function is also very similar to `rassoc.matern`, where `rassoc.circular` needs the study window to be specified, while `rassoc.matern` does not.

Usage

```
rassoc.circular(
  n,
  Yp,
  e,
  a1 = min(Yp[, 1]),
  a2 = max(Yp[, 1]),
  b1 = min(Yp[, 2]),
  b2 = max(Yp[, 2])
)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`Yp`	A set of 2D points representing the reference points. The generated points are associated (in a circular or radial fashion) with these points.
`e`	A positive real number representing the radius of the balls centered at `Yp` points. Only these balls are allowed for the generated points (i.e., generated points would be in the union of these balls).
`a1, a2`	Real numbers representing the range of x-coordinates in the region (default is the range of x-coordinates of the `Yp` points).
`b1, b2`	Real numbers representing the range of y-coordinates in the region (default is the range of y-coordinates of the `Yp` points).

Value

A `list` with the elements

`type`	The type of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	Radial attraction parameter of the association pattern
`ref.points`	The input set of attraction points `Yp`, i.e., points with which generated points are associated.
`gen.points`	The output set of generated points associated with `Yp` points
`tri.Yp`	Logical output for triangulation based on `Yp` points should be implemented or not. if `TRUE` triangulation based on `Yp` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of attraction (i.e., `Yp`) points.
`xlimit, ylimit`	The possible range of the x- and y-coordinates of the generated points.

Author(s)

Elvan Ceyhan

See Also

`rseg.circular`, `rassoc.std.tri`, `rassocII.std.tri`, `rassoc.matern`, and `rassoc.multi.tri`

Examples

```
nx<-100; ny<-4;  #try also nx<-1000; ny<-10;

e<-.15;
#with default bounding box (i.e., unit square)
Y<-cbind(runif(ny),runif(ny))

Xdt<-rassoc.circular(nx,Y,e)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xdt<-Xdt$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",
main="Circular Association of X points with Y Points",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),
     pch=16,col=2,lwd=2)
points(Xdt)

#with default bounding box (i.e., unit square)
Xlim<-range(Xdt[,1],Y[,1]);
```

```
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",
main="Circular Association of X points with Y Points",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),pch=16,
     col=2,lwd=2)
points(Xdt)

#with a rectangular bounding box
a1<-0; a2<-10;
b1<-0; b2<-5;
e<-1.1;  #try also e<-5; #pattern very close to CSR!

Y<-cbind(runif(ny,a1,a2),runif(ny,b1,b2))
#try also Y<-cbind(runif(ny,a1,a2/2),runif(ny,b1,b2/2))

Xdt<-rassoc.circular(nx,Y,e,a1,a2,b1,b2)$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",
main="Circular Association of X points with Y Points",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),
     pch=16,col=2,lwd=2)
points(Xdt)
```

`rassoc.matern` *Generation of points associated (in a Matern-like fashion) to a given set of points*

Description

An object of class `"Patterns"`. Generates `n` 2D points uniformly in $\cup B(y_i, e)$ where $Y_p = (y_1, y_2, \ldots, y_{n_y})$ with n_y being number of `Yp` points for various values of `e` under the association pattern and $B(y_i, e)$ is the ball centered at y_i with radius `e`.

The pattern resembles the Matern cluster pattern (see `rMatClust` in the `spatstat.random` package for further information (Baddeley and Turner (2005)). `rMatClust(kappa, scale, mu, win)` in the simplest case generates a uniform Poisson point process of "parent" points with intensity `kappa`. Then each parent point is replaced by a random cluster of "offspring" points, the number of points per cluster being Poisson(`mu`) distributed, and their positions being placed and uniformly inside a disc of radius scale centered on the parent point. The resulting point pattern is a realization of the classical "stationary Matern cluster process" generated inside the window `win`.

The main difference of `rassoc.matern` and `rMatClust` is that the parent points are `Yp` points which are given beforehand and we do not discard them in the end in `rassoc.matern` and the offspring points are the points associated with the reference points, `Yp`; `e` must be positive and very large values of `e` provide patterns close to CSR.

This function is also very similar to `rassoc.circular`, where `rassoc.circular` needs the study window to be specified, while `rassoc.matern` does not.

Usage

```
rassoc.matern(n, Yp, e)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`Yp`	A set of 2D points representing the reference points. The generated points are associated (in a Matern-cluster like fashion) with these points.
`e`	A positive real number representing the radius of the balls centered at `Yp` points. Only these balls are allowed for the generated points (i.e., generated points would be in the union of these balls).

Value

A `list` with the elements

`type`	The type of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	Radial (i.e., circular) attraction parameter of the association pattern.
`ref.points`	The input set of attraction points `Yp`, i.e., points with which generated points are associated.
`gen.points`	The output set of generated points associated with `Yp` points.
`tri.Yp`	Logical output for triangulation based on `Yp` points should be implemented or not. if `TRUE` triangulation based on `Yp` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of attraction (i.e., `Yp`) points.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated points.

Author(s)

Elvan Ceyhan

References

Baddeley AJ, Turner R (2005). "spatstat: An R Package for Analyzing Spatial Point Patterns." *Journal of Statistical Software*, **12(6)**, 1-42.

See Also

rassoc.circular, rassoc.std.tri, rassocII.std.tri, rassoc.multi.tri, rseg.circular, and rMatClust in the spatstat.random package

Examples

```
nx<-100; ny<-4;  #try also nx<-1000; ny<-10;

e<-.15;
 #try also e<-1.1; #closer to CSR than association, as e is large

#Y points uniform in unit square
Y<-cbind(runif(ny),runif(ny))

Xdt<-rassoc.matern(nx,Y,e)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xdt<-Xdt$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",
main="Matern-like Association of X points with Y Points",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),
     pch=16,col=2,lwd=2)
points(Xdt)

a1<-0; a2<-10;
b1<-0; b2<-5;
e<-1.1;

#Y points uniform in a rectangle
Y<-cbind(runif(ny,a1,a2),runif(ny,b1,b2))
#try also Y<-cbind(runif(ny,a1,a2/2),runif(ny,b1,b2/2))

Xdt<-rassoc.matern(nx,Y,e)$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",
main="Matern-like Association of X points with Y Points",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),pch=16,col=2,lwd=2)
points(Xdt)
```

`rassoc.multi.tri` *Generation of points associated (in a Type I fashion) with a given set of points*

Description

An object of class `"Patterns"`. Generates `n` points uniformly in the support for Type I association in the convex hull of set of points, `Yp`. `delta` is the parameter of association (that is, only $\delta 100$ % area around each vertex in each Delaunay triangle is allowed for point generation).

`delta` corresponds to `eps` in the standard equilateral triangle T_e as $delta = 4eps^2/3$ (see `rseg.std.tri` function).

If `Yp` consists only of 3 points, then the function behaves like the function `rassoc.tri`.

`DTmesh` must be the Delaunay triangulation of `Yp` and `DTr` must be the corresponding Delaunay triangles (both `DTmesh` and `DTr` are `NULL` by default). If `NULL`, `DTmesh` is computed via `tri.mesh` and `DTr` is computed via `triangles` function in `interp` package.

`tri.mesh` function yields the triangulation nodes with their neighbours, and creates a triangulation object, and `triangles` function yields a triangulation data structure from the triangulation object created by `tri.mesh` (the first three columns are the vertex indices of the Delaunay triangles).

See (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for more on the association pattern. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
rassoc.multi.tri(n, Yp, delta, DTmesh = NULL, DTr = NULL)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`Yp`	A set of 2D points from which Delaunay triangulation is constructed.
`delta`	A positive real number in $(0, 1)$. `delta` is the parameter of association (that is, only $\delta 100$ % area around vertices of each Delaunay triangle is allowed for point generation).
`DTmesh`	Delaunay triangulation of `Yp`, default is `NULL`, which is computed via `tri.mesh` function in `interp` package. `tri.mesh` function yields the triangulation nodes with their neighbours, and creates a triangulation object.
`DTr`	Delaunay triangles based on `Yp`, default is `NULL`, which is computed via `tri.mesh` function in `interp` package. `triangles` function yields a triangulation data structure from the triangulation object created by `tri.mesh`.

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	Attraction parameter, `delta`, of the Type I association pattern. `delta` is in $(0, 1)$ and only $\delta 100$ % of the area around vertices of each Delaunay triangle is allowed for point generation.
`ref.points`	The input set of points `Yp`; reference points, i.e., points with which generated points are associated.
`gen.points`	The output set of generated points associated with `Yp` points.
`tri.Y`	Logical output, `TRUE` if triangulation based on `Yp` points should be implemented.
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., `Yp`) points.
`xlimit`, `ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`rassoc.circular`, `rassoc.std.tri`, `rassocII.std.tri`, and `rseg.multi.tri`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-100; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Yp<-cbind(runif(ny),runif(ny))
del<-.4

Xdt<-rassoc.multi.tri(nx,Yp,del)
Xdt
summary(Xdt)
plot(Xdt)

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")
#Delaunay triangulation based on Y points
TRY<-interp::triangles(DTY)[,1:3];
Xp<-rassoc.multi.tri(nx,Yp,del,DTY,TRY)$g
#data under CSR in the convex hull of Ypoints

Xlim<-range(Yp[,1])
Ylim<-range(Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot of the data in the convex hull of Y points together with the Delaunay triangulation
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")
#Delaunay triangulation based on Y points

plot(Xp,main="Points from Type I Association \n in Multipe Triangles",
xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),
ylim=Ylim+yd*c(-.05,.05),type="n")
interp::plot.triSht(DTY, add=TRUE,
do.points=TRUE,col="blue")
points(Xp,pch=".",cex=3)
```

`rassoc.std.tri` *Generation of points associated (in a Type I fashion) with the vertices of T_e*

Description

An object of class `"Patterns"`. Generates `n` points uniformly in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ under the type I association alternative for `eps` in $(0,\sqrt{3}/3 = 0.5773503]$. The allowed triangular regions around the vertices are determined by the parameter `eps`.

In the type I association, the triangular support regions around the vertices are determined by the parameter `eps` where $\sqrt{3}/3$-`eps` serves as the height of these triangles (see examples for a sample plot.)

See also (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)).

Usage

```
rassoc.std.tri(n, eps)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`eps`	A positive real number representing the parameter of type I association (where $\sqrt{3}/3$-`eps` serves as the height of the triangular support regions around the vertices).

Value

A `list` with the elements

`type`	The type of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	The attraction parameter of the association pattern, `eps`, where $\sqrt{3}/3$-`eps` serves as the height of the triangular support regions around the vertices
`ref.points`	The input set of points `Y`; reference points, i.e., points with which generated points are associated (i.e., vertices of T_e).
`gen.points`	The output set of generated points associated with `Y` points (i.e., vertices of T_e).
`tri.Y`	Logical output for triangulation based on `Y` points should be implemented or not. if `TRUE` triangulation based on `Y` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern.
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., `Y`) points.
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the vertices of T_e here

Author(s)

Elvan Ceyhan

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`rseg.circular`, `rassoc.circular`, `rsegII.std.tri`, and `rseg.multi.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-100  #try also n<-20 or n<-100 or 1000
eps<-.25  #try also .15, .5, .75

set.seed(1)
Xdt<-rassoc.std.tri(n,eps)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Xp<-Xdt$gen.points
plot(Te,pch=".",xlab="",ylab="",
main="Type I association in the \n standard equilateral triangle",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(Xp)

#The support for the Type I association alternative
sr<-(sqrt(3)/3-eps)/(sqrt(3)/2)
C1<-C+sr*(A-C); C2<-C+sr*(B-C)
A1<-A+sr*(B-A); A2<-A+sr*(C-A)
B1<-B+sr*(A-B); B2<-B+sr*(C-B)
supp<-rbind(A1,B1,B2,C2,C1,A2)

plot(Te,asp=1,pch=".",xlab="",ylab="",
main="Support of the Type I Association",
xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
if (sr<=.5)
{
  polygon(Te,col=5)
  polygon(supp,col=0)
} else
{
  polygon(Te,col=0,lwd=2.5)
  polygon(rbind(A,A1,A2),col=5,border=NA)
```

```
  polygon(rbind(B,B1,B2),col=5,border=NA)
  polygon(rbind(C,C1,C2),col=5,border=NA)
}
points(Xp)
```

`rassoc.tri` *Generation of points associated (in a Type I fashion) with the vertices of a triangle*

Description

An object of class `"Patterns"`. Generates `n` points uniformly in the support for Type I association in a given triangle, `tri`. `delta` is the parameter of association (that is, only $\delta 100$ % area around each vertex in the triangle is allowed for point generation). `delta` corresponds to `eps` in the standard equilateral triangle T_e as $delta = 4eps^2/3$ (see `rseg.std.tri` function).

See (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for more on the association pattern.

Usage

```
rassoc.tri(n, tri, delta)
```

Arguments

`n`	A positive integer representing the number of points to be generated from the association pattern in the triangle, `tri`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`delta`	A positive real number in $(0, 1)$. `delta` is the parameter of association (that is, only $\delta 100$ % area around vertices of the triangle is allowed for point generation).

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	Attraction parameter, `delta`, of the Type I association pattern. `delta` is in $(0, 1)$ and only $\delta 100$ % of the area around vertices of the triangle `tri` is allowed for point generation.
`ref.points`	The input set of points, i.e., vertices of `tri`; reference points, i.e., points with which generated points are associated.
`gen.points`	The output set of generated points associated with the vertices of `tri`.
`tri.Y`	Logical output, `TRUE` if triangulation based on `Yp` points should be implemented.
`desc.pat`	Description of the point pattern

`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., `Yp`) points.
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`rseg.tri`, `rassoc.std.tri`, `rassocII.std.tri`, and `rassoc.multi.tri`

Examples

```
n<-100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)
del<-.4

Xdt<-rassoc.tri(n,Tr,del)
Xdt
summary(Xdt)
plot(Xdt)

Xp<-Xdt$g
Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",
main="Points from Type I Association \n in one Triangle",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp)
xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.03)
txt.str<-c("A","B","C")
```

```
text(xc,yc,txt.str)
```

---

rassocII.std.tri *Generation of points associated (in a Type II fashion) with the edges of T_e*

Description

An object of class `"Patterns"`. Generates `n` points uniformly in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ under the type II association alternative for `eps` in $(0, \sqrt{3}/6 = 0.2886751]$.

In the type II association, the annular allowed regions around the edges are determined by the parameter `eps` where $\sqrt{3}/6$-`eps` is the distance from the interior triangle (i.e., forbidden region for association) to T_e (see examples for a sample plot.)

Usage

```
rassocII.std.tri(n, eps)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`eps`	A positive real number representing the parameter of type II association (where $\sqrt{3}/6$-`eps` is the distance from the interior triangle distance from the interior triangle to T_e).

Value

A `list` with the elements

`type`	The type of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	The attraction parameter, `eps`, of the association pattern, where $\sqrt{3}/6$-`eps` is the distance from the interior triangle to T_e
`ref.points`	The input set of points `Y`; reference points, i.e., points with which generated points are associated (i.e., vertices of T_e).
`gen.points`	The output set of generated points associated with `Y` points (i.e., edges of T_e).
`tri.Y`	Logical output for triangulation based on `Y` points should be implemented or not. if `TRUE` triangulation based on `Y` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., `Y`) points, which is 3 here.
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the vertices of T_e here.

Author(s)

Elvan Ceyhan

See Also

`rseg.circular`, `rassoc.circular`, `rsegII.std.tri`, and `rseg.multi.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-100  #try also n<-20 or n<-100 or 1000
eps<-.2  #try also .25, .1

set.seed(1)
Xdt<-rassocII.std.tri(n,eps)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Xp<-Xdt$gen.points
plot(Te,pch=".",xlab="",ylab="",
main="Type II association in the \n standard equilateral triangle",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(Xp)

#The support for the Type II association alternative
A1<-c(1/2-eps*sqrt(3),sqrt(3)/6-eps);
B1<-c(1/2+eps*sqrt(3),sqrt(3)/6-eps);
C1<-c(1/2,sqrt(3)/6+2*eps);
supp<-rbind(A1,B1,C1)

plot(Te,asp=1,pch=".",xlab="",ylab="",
main="Support of the Type II Association",
xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te,col=5)
polygon(supp,col=0)
points(Xp)
```

`rel.edge.basic.tri` *The index of the edge region in a standard basic triangle form that contains a point*

Description

Returns the index of the edge whose region contains point, `p`, in the standard basic triangle form $T_b = T(A = (0, 0), B = (1, 0), C = (c_1, c_2))$ and edge regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle form T_b.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B, C, M)$, edge region 2 is $T(A, C, M)$, and edge region 3 is $T(A, B, M)$. In the standard basic triangle form T_b c_1 is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the standard basic triangle form by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle form is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
rel.edge.basic.tri(p, c1, c2, M)
```

Arguments

`p`	A 2D point for which `M`-edge region it resides in is to be determined in the standard basic triangle form T_b.
`c1, c2`	Positive real numbers which constitute the upper vertex of the standard basic triangle form (i.e., the vertex adjacent to the shorter edges of T_b); c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle form T_b.

Value

A `list` with three elements

`re`	Index of the `M`-edge region that contains point, `p` in the standard basic triangle form T_b.
`tri`	The vertices of the triangle, where row labels are A, B, and C with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
`desc`	Description of the edge labels

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`rel.edge.triCM`, `rel.edge.tri`, `rel.edge.basic.tri`, `rel.edge.std.triCM`, and `edge.reg.triCM`

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);
M<-c(.6,.2)

P<-c(.4,.2)
rel.edge.basic.tri(P,c1,c2,M)

A<-c(0,0);B<-c(1,0);C<-c(c1,c2);
Tb<-rbind(A,B,C)

n<-20  #try also n<-40
Xp<-runif.basic.tri(n,c1,c2)$g

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.2)

re<-vector()
for (i in 1:n)
  re<-c(re,rel.edge.basic.tri(Xp[i,],c1,c2,M)$re)
re

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
plot(Tb,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(Xp,pch=".")
polygon(Tb)
L<-Tb; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(re))

txt<-rbind(Tb,M)
xc<-txt[,1]+c(-.03,.03,.02,0)
yc<-txt[,2]+c(.02,.02,.02,-.03)
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)
```

`rel.edge.basic.triCM` *The index of the CM-edge region in a standard basic triangle form that contains a point*

Description

Returns the index of the edge whose region contains point, `p`, in the standard basic triangle form $T_b = T(A = (0, 0), B = (1, 0), C = (c_1, c_2)$ where c_1 is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \le 1$ with edge regions based on center of mass $CM = (A + B + C)/3$.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B, C, CM)$, edge region 2 is $T(A, C, CM)$, and edge region 3 is $T(A, B, CM)$.

Any given triangle can be mapped to the standard basic triangle form by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle form is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
rel.edge.basic.triCM(p, c1, c2)
```

Arguments

`p`	A 2D point for which CM-edge region it resides in is to be determined in the standard basic triangle form T_b.
`c1, c2`	Positive real numbers which constitute the upper vertex of the standard basic triangle form (i.e., the vertex adjacent to the shorter edges of T_b); c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \le 1$.

Value

A `list` with three elements

`re`	Index of the CM-edge region that contains point, `p` in the standard basic triangle form T_b
`tri`	The vertices of the triangle, where row labels are $A = (0, 0)$, $B = (1, 0)$, and $C = (c_1, c_2)$ with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
`desc`	Description of the edge labels

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`rel.edge.triCM`, `rel.edge.tri`, `rel.edge.basic.tri`, `rel.edge.std.triCM`, and `edge.reg.triCM`

Examples

```
c1<-.4; c2<-.6
P<-c(.4,.2)
rel.edge.basic.triCM(P,c1,c2)

A<-c(0,0);B<-c(1,0);C<-c(c1,c2);
Tb<-rbind(A,B,C)
CM<-(A+B+C)/3

rel.edge.basic.triCM(A,c1,c2)
rel.edge.basic.triCM(B,c1,c2)
rel.edge.basic.triCM(C,c1,c2)
rel.edge.basic.triCM(CM,c1,c2)

n<-20  #try also n<-40
```

```
Xp<-runif.basic.tri(n,c1,c2)$g

re<-vector()
for (i in 1:n)
  re<-c(re,rel.edge.basic.triCM(Xp[i,],c1,c2)$re)
re

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(Xp,pch=".")
polygon(Tb)
L<-Tb; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(re))

txt<-rbind(Tb,CM)
xc<-txt[,1]+c(-.03,.03,.02,0)
yc<-txt[,2]+c(.02,.02,.02,-.04)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)
```

`rel.edge.std.triCM` *The index of the edge region in the standard equilateral triangle that contains a point*

Description

Returns the index of the edge whose region contains point, `p`, in the standard equilateral triangle $T_e = T(A = (0,0), B = (1,0), C = (1/2, \sqrt{3}/2))$ with edge regions based on center of mass $CM = (A + B + C)/3$.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B, C, M)$, edge region 2 is $T(A, C, M)$, and edge region 3 is $T(A, B, M)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
rel.edge.std.triCM(p)
```

Arguments

`p`	A 2D point for which CM-edge region it resides in is to be determined in the the standard equilateral triangle T_e.

Value

A `list` with three elements

`re`	Index of the CM-edge region that contains point, `p` in the standard equilateral triangle T_e
`tri`	The vertices of the standard equilateral triangle T_e, where row labels are A, B, and C with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
`desc`	Description of the edge labels

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`rel.edge.triCM`, `rel.edge.tri`, `rel.edge.basic.triCM`, `rel.edge.basic.tri`, and `edge.reg.triCM`

Examples

```
P<-c(.4,.2)
rel.edge.std.triCM(P)

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
CM<-(A+B+C)/3

n<-20  #try also n<-40
Xp<-runif.std.tri(n)$gen.points

re<-vector()
for (i in 1:n)
  re<-c(re,rel.edge.std.triCM(Xp[i,])$re)
```

```
re

Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,asp=1,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
points(Xp,pch=".")
polygon(Te)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(re))

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.03,.03,.03,-.06)
yc<-txt[,2]+c(.02,.02,.02,.03)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

p1<-(A+B+CM)/3
p2<-(B+C+CM)/3
p3<-(A+C+CM)/3

plot(Te,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Te,CM,p1,p2,p3)
xc<-txt[,1]+c(-.03,.03,.03,-.06,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.03,0,0,0)
txt.str<-c("A","B","C","CM","re=3","re=1","re=2")
text(xc,yc,txt.str)
```

`rel.edge.tri` *The index of the edge region in a triangle that contains the point*

Description

Returns the index of the edge whose region contains point, `p`, in the triangle `tri`$= T(A,B,C)$ with edge regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B,C,M)$, edge region 2 is $T(A,C,M)$, and edge region 3 is $T(A,B,M)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
rel.edge.tri(p, tri, M)
```

Arguments

`p`	A 2D point for which `M`-edge region it resides in is to be determined in the triangle `tri`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.

Value

A `list` with three elements

`re`	Index of the `M`-edge region that contains point, `p` in the triangle `tri`.
`tri`	The vertices of the triangle, where row labels are A, B, and C with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
`desc`	Description of the edge labels

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`rel.edge.triCM`, `rel.edge.basic.triCM`, `rel.edge.basic.tri`, `rel.edge.std.triCM`, and `edge.reg.triCM`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(1.4,1.2)
M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

rel.edge.tri(P,Tr,M)

n<-20  #try also n<-40
Xp<-runif.tri(n,Tr)$g

re<-vector()
for (i in 1:n)
  re<-c(re,rel.edge.tri(Xp[i,],Tr,M)$re)
re

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}

plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".")
L<-Tr; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(re))

txt<-rbind(Tr,M)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)

p1<-(A+B+M)/3
p2<-(B+C+M)/3
p3<-(A+C+M)/3

plot(Tr,xlab="",ylab="", main="Illustration of M-edge regions in a triangle",
axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-Tr; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Tr,M,p1,p2,p3)
xc<-txt[,1]+c(-.02,.02,.02,.02,.02,.02,.02)
yc<-txt[,2]+c(.02,.02,.04,.05,.02,.02,.02)
txt.str<-c("A","B","C","M","re=3","re=1","re=2")
text(xc,yc,txt.str)
```

`rel.edge.triCM` *The index of the CM-edge region in a triangle that contains the point*

Description

Returns the index of the edge whose region contains point, `p`, in the triangle `tri`$= T(A, B, C)$ with edge regions based on center of mass $CM = (A + B + C)/3$.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B, C, CM)$, edge region 2 is $T(A, C, CM)$, and edge region 3 is $T(A, B, CM)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
rel.edge.triCM(p, tri)
```

Arguments

`p`	A 2D point for which CM-edge region it resides in is to be determined in the triangle `tri`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with three elements

`re`	Index of the CM-edge region that contains point, `p` in the triangle `tri`.
`tri`	The vertices of the triangle, where row labels are A, B, and C with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
`desc`	Description of the edge labels

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`rel.edge.tri`, `rel.edge.basic.triCM`, `rel.edge.basic.tri`, `rel.edge.std.triCM`, and `edge.reg.triCM`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
P<-c(1.4,1.2)
rel.edge.triCM(P,Tr)

P<-c(1.5,1.61)
rel.edge.triCM(P,Tr)

CM<-(A+B+C)/3

n<-20  #try also n<-40
Xp<-runif.tri(n,Tr)$g

re<-vector()
for (i in 1:n)
  re<-c(re,rel.edge.triCM(Xp[i,],Tr)$re)
re

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(Xp,pch=".")
polygon(Tr)
L<-Tr; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(re))

txt<-rbind(Tr,CM)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

p1<-(A+B+CM)/3
p2<-(B+C+CM)/3
p3<-(A+C+CM)/3
```

```
plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-Tr; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Tr,CM,p1,p2,p3)
xc<-txt[,1]+c(-.02,.02,.02,.02,.02,.02,.02)
yc<-txt[,2]+c(.02,.02,.04,.05,.02,.02,.02)
txt.str<-c("A","B","C","CM","re=3","re=1","re=2")
text(xc,yc,txt.str)
```

`rel.edges.tri` *The indices of the* `M`*-edge regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the edges whose regions contain the points in data set `Xp` in a triangle `tri`$= T(A, B, C)$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as $1 = A$, $2 = B$, and $3 = C$ also according to the row number the vertex is recorded in `tri` and the corresponding edges are $1 = BC$, $2 = AC$, and $3 = AB$.

If a point in `Xp` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding edge region is the polygon with the vertex, `M`, and vertices other than the non-adjacent vertex, i.e., edge region 1 is the triangle $T(B, M, C)$, edge region 2 is $T(A, M, C)$ and edge region 3 is $T(A, B, M)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
rel.edges.tri(Xp, tri, M)
```

Arguments

`Xp`	A set of 2D points representing the set of data points for which indices of the edge regions containing them are to be determined.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.

Value

A `list` with the elements

`re`	Indices (i.e., a `vector` of indices) of the edges whose region contains points in `Xp` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index opposite to edge whose index is given in re.
`desc`	Description of the edge labels as `"Edge labels are AB=3, BC=1, and AC=2"`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`rel.edges.triCM`, `rel.verts.tri`, and `rel.verts.tri.nondegPE`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-c(1.6,1.2)

P<-c(.4,.2)
rel.edges.tri(P,Tr,M)

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

(re<-rel.edges.tri(Xp,Tr,M))
```

```
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",
main="Scatterplot of data points \n and the M-edge regions",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".",col=1)
L<-Tr; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]+c(-.02,.03,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(.05,.06,-.05,-.02)
yc<-txt[,2]+c(.03,.03,.05,-.08)
txt.str<-c("M","re=2","re=3","re=1")
text(xc,yc,txt.str)
text(Xp,labels=factor(re$re))
```

`rel.edges.triCM` *The indices of the CM-edge regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the edges whose regions contain the points in data set `Xp` in a triangle `tri`$=(A,B,C)$ and edge regions are based on the center of mass CM of `tri`. (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as $1=A$, $2=B$, and $3=C$ also according to the row number the vertex is recorded in `tri` and the corresponding edges are $1=BC$, $2=AC$, and $3=AB$.

If a point in `Xp` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding edge region is the polygon with the vertex, CM, and vertices other than the non-adjacent vertex, i.e., edge region 1 is the triangle $T(B, CM, C)$, edge region 2 is $T(A, CM, C)$ and edge region 3 is $T(A, B, CM)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
rel.edges.triCM(Xp, tri)
```

Arguments

`Xp`	A set of 2D points representing the set of data points for which indices of the edge regions containing them are to be determined.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with the elements

`re`	Indices (i.e., a `vector` of indices) of the edges whose region contains points in `Xp` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.
`desc`	Description of the edge labels as `"Edge labels are AB=3, BC=1, and AC=2"`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`rel.edges.tri`, `rel.verts.tri`, and `rel.verts.tri.nondegPE`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(.4,.2)
rel.edges.triCM(P,Tr)

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.tri(n,Tr)$g

re<-rel.edges.triCM(Xp,Tr)
re
CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".",col=1)
L<-Tr; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]+c(-.02,.03,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-rbind(CM,Ds)
xc<-txt[,1]+c(.05,.06,-.05,-.02)
yc<-txt[,2]+c(.03,.03,.05,-.08)
txt.str<-c("CM","re=2","re=3","re=1")
text(xc,yc,txt.str)
text(Xp,labels=factor(re$re))
```

`rel.vert.basic.tri` *The index of the vertex region in a standard basic triangle form that contains a given point*

Description

Returns the index of the related vertex in the standard basic triangle form whose region contains point p. The standard basic triangle form is $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$..

Vertex regions are based on the general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the standard basic triangle form T_b. Vertices of the standard basic triangle form T_b are labeled according to the row number the vertex is recorded, i.e., as 1=(0,0), 2=(1,0),and $3 = (c_1, c_2)$.

If the point, `p`, is not inside T_b, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, `M`, and projections from `M` to the edges on the lines joining vertices and `M`. That is, `rv=1` has vertices $(0,0), D_3, M, D_2$; `rv=2` has vertices $(1,0), D_1, M, D_3$; and $rv = 3$ has vertices $(c_1, c_2), D_2, M, D_1$ (see the illustration in the examples).

Any given triangle can be mapped to the standard basic triangle form by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle form is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.vert.basic.tri(p, c1, c2, M)
```

Arguments

`p`	A 2D point for which `M`-vertex region it resides in is to be determined in the standard basic triangle form T_b.
`c1, c2`	Positive real numbers which constitute the vertex of the standard basic triangle form adjacent to the shorter edges; c_1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard basic triangle form.

Value

A `list` with two elements

`rv`	Index of the vertex whose region contains point, `p`; index of the vertex is the same as the row number in the standard basic triangle form, T_b
`tri`	The vertices of the standard basic triangle form, T_b, where row number corresponds to the vertex index `rv` with `rv=1` is row $1 = (0,0)$, `rv=2` is row $2 = (1,0)$, and $rv = 3$ is row $3 = (c_1, c_2)$.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.vert.basic.triCM`, `rel.vert.tri`, `rel.vert.triCC`, `rel.vert.basic.triCC`, `rel.vert.triCM`, and `rel.vert.std.triCM`

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);
M<-c(.6,.2)

P<-c(.4,.2)
rel.vert.basic.tri(P,c1,c2,M)

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.basic.tri(n,c1,c2)$g

M<-as.numeric(runif.basic.tri(1,c1,c2)$g)  #try also M<-c(.6,.2)

Rv<-vector()
for (i in 1:n)
{ Rv<-c(Rv,rel.vert.basic.tri(Xp[i,],c1,c2,M)$rv)}
Rv

Ds<-prj.cent2edges.basic.tri(c1,c2,M)

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tb)}
#need to run this when M is given in barycentric coordinates

plot(Tb,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.1,.1),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(Xp,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tb[,1]+c(-.04,.05,.04)
yc<-Tb[,2]+c(.02,.02,.03)
```

```
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.04,-.03,0)
yc<-txt[,2]+c(-.02,.02,.02,-.03)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
text(Xp,labels=factor(Rv))
```

`rel.vert.basic.triCC` *The index of the CC-vertex region in a standard basic triangle form that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in the standard basic triangle form $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$ and vertex regions are based on the circumcenter CC of T_b. (see the plots in the example for illustrations).

The vertices of the standard basic triangle form T_b are labeled as $1 = (0,0)$, $2 = (1,0)$,and $3 = (c_1,c_2)$ also according to the row number the vertex is recorded in T_b. If the point, `p`, is not inside T_b, then the function yields `NA` as output. The corresponding vertex region is the polygon whose interior points are closest to that vertex.

Any given triangle can be mapped to the standard basic triangle form by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle form is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.vert.basic.triCC(p, c1, c2)
```

Arguments

`p`	A 2D point for which CC-vertex region it resides in is to be determined in the standard basic triangle form T_b.
`c1, c2`	Positive real numbers which constitute the upper vertex of the standard basic triangle form (i.e., the vertex adjacent to the shorter edges of T_b); c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Value

A `list` with two elements

`rv`	Index of the CC-vertex region that contains point, `p` in the standard basic triangle form T_b
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv` with row $1 = (0, 0)$, row $2 = (1, 0)$, and row $3 = (c_1, c_2)$.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.vert.triCM`, `rel.vert.tri`, `rel.vert.triCC`, `rel.vert.basic.triCM`, `rel.vert.basic.tri`, and `rel.vert.std.triCM`

Examples

```
c1<-.4; c2<-.6;  #try also c1<-.5; c2<-.5;

P<-c(.3,.2)
rel.vert.basic.triCC(P,c1,c2)

A<-c(0,0);B<-c(1,0);C<-c(c1,c2);
Tb<-rbind(A,B,C)
CC<-circumcenter.basic.tri(c1,c2)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1])
Ylim<-range(Tb[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,asp=1,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
```

```
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Tb,CC,Ds)
xc<-txt[,1]+c(-.03,.03,0.02,-.01,.05,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.03,.03,-.03)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

RV1<-(A+D3+CC+D2)/4
RV2<-(B+D3+CC+D1)/4
RV3<-(C+D2+CC+D1)/4

txt<-rbind(RV1,RV2,RV3)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

n<-20  #try also n<-40
Xp<-runif.basic.tri(n,c1,c2)$g

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rel.vert.basic.triCC(Xp[i,],c1,c2)$rv)
Rv

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,asp=1,xlab="",pch=".",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(Xp,pch=".")
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(Rv))

txt<-rbind(Tb,CC,Ds)
xc<-txt[,1]+c(-.03,.03,0.02,-.01,.05,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.03,.03,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
```

`rel.vert.basic.triCM` *The index of the CM-vertex region in a standard basic triangle form that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in the standard basic triangle form $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$ and vertex regions are based on the center of mass CM=((1+c1)/3,c2/3) of T_b. (see the plots in the example for illustrations).

The vertices of the standard basic triangle form T_b are labeled as $1 = (0,0)$, $2 = (1,0)$,and $3 = (c_1,c_2)$ also according to the row number the vertex is recorded in T_b. If the point, `p`, is not inside T_b, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, CM, and midpoints of the edges adjacent to the vertex.

Any given triangle can be mapped to the standard basic triangle form by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle form is useful for simulation studies under the uniformity hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2006))

Usage

```
rel.vert.basic.triCM(p, c1, c2)
```

Arguments

`p`	A 2D point for which CM-vertex region it resides in is to be determined in the standard basic triangle form T_b.
`c1, c2`	Positive real numbers which constitute the upper vertex of the standard basic triangle form (i.e., the vertex adjacent to the shorter edges of T_b); c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Value

A `list` with two elements

`rv`	Index of the CM-vertex region that contains point, `p` in the standard basic triangle form T_b
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv` with row $1 = (0,0)$, row $2 = (1,0)$, and row $3 = (c_1,c_2)$.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

#' @author Elvan Ceyhan

See Also

`rel.vert.triCM`, `rel.vert.tri`, `rel.vert.triCC`, `rel.vert.basic.triCC`, `rel.vert.basic.tri`, and `rel.vert.std.triCM`

Examples

```
c1<-.4; c2<-.6
P<-c(.4,.2)
rel.vert.basic.triCM(P,c1,c2)

A<-c(0,0);B<-c(1,0);C<-c(c1,c2);
Tb<-rbind(A,B,C)
CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

n<-20  #try also n<-40
Xp<-runif.basic.tri(n,c1,c2)$g

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rel.vert.basic.triCM(Xp[i,],c1,c2)$rv)
Rv

Xlim<-range(Tb[,1],Xp[,1])
Ylim<-range(Tb[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,xlab="",ylab="",axes="T",pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(Xp,pch=".")
polygon(Tb)
L<-Ds; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(Rv))

txt<-rbind(Tb,CM,Ds)
xc<-txt[,1]+c(-.03,.03,.02,-.01,.06,-.05,.0)
yc<-txt[,2]+c(.02,.02,.02,.04,.02,.02,-.03)
txt.str<-c("A","B","C","CM","D1","D2","D3")
text(xc,yc,txt.str)

plot(Tb,xlab="",ylab="",axes="T",pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-Ds; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
```

```
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

RV1<-(A+D3+CM+D2)/4
RV2<-(B+D3+CM+D1)/4
RV3<-(C+D2+CM+D1)/4

txt<-rbind(RV1,RV2,RV3)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(Tb,CM,Ds)
xc<-txt[,1]+c(-.03,.03,.02,-.01,.04,-.03,.0)
yc<-txt[,2]+c(.02,.02,.02,.04,.02,.02,-.03)
txt.str<-c("A","B","C","CM","D1","D2","D3")
text(xc,yc,txt.str)
```

`rel.vert.end.int` *The index of the vertex region in an end-interval that contains a given point*

Description

Returns the index of the vertex in the interval, `int`, whose end interval contains the 1D point `p`, that is, it finds the index of the vertex for the point, `p`, outside the interval `int`$=(a,b)=$(vertex 1,vertex 2); vertices of interval are labeled as 1 and 2 according to their order in the interval.

If the point, `p`, is inside `int`, then the function yields `NA` as output. The corresponding vertex region is an interval as $(-\infty,a)$ or (b,∞) for the interval (a,b). Then if $p<a$, then `rv=1` and if $p>b$, then `rv=2`. Unlike `rel.vert.mid.int`, centrality parameter (i.e., center of the interval is not relevant for `rel.vert.end.int`.)

See also (Ceyhan (2012, 2016)).

Usage

```
rel.vert.end.int(p, int)
```

Arguments

`p`	A 1D point whose end interval region is provided by the function.
`int`	A `vector` of two real numbers representing an interval.

Value

A `list` with two elements

`rv`	Index of the end vertex whose region contains point, `p`.
`int`	The vertices of the interval as a `vector` where position of the vertex corresponds to the vertex index as `int=(rv=1,rv=2)`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`rel.vert.mid.int`

Examples

```
a<-0; b<-10; int<-c(a,b)

rel.vert.end.int(-6,int)
rel.vert.end.int(16,int)

n<-10
xf<-(int[2]-int[1])*.5
XpL<-runif(n,a-xf,a)
XpR<-runif(n,b,b+xf)
Xp<-c(XpL,XpR)
rel.vert.end.int(Xp[1],int)

Rv<-vector()
for (i in 1:length(Xp))
  Rv<-c(Rv,rel.vert.end.int(Xp[i],int)$rv)
Rv

Xlim<-range(a,b,Xp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),xlab="",pch=".",xlim=Xlim+xd*c(-.05,.05))
abline(h=0)
abline(v=c(a,b),col=1,lty = 2)
points(cbind(Xp,0))
text(cbind(Xp,0.1),labels=factor(Rv))
text(cbind(c(a,b),-0.1),c("rv=1","rv=2"))
```

```
jit<-.1
yjit<-runif(length(Xp),-jit,jit)

Xlim<-range(a,b,Xp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),
main="vertex region indices for the points\n in the end intervals",
     xlab=" ", ylab=" ",pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=3*range(yjit))
points(Xp, yjit,xlim=Xlim+xd*c(-.05,.05),pch=".",cex=3)
abline(h=0)
abline(v=c(a,b),lty = 2)
text(Xp,yjit,labels=factor(Rv))
text(cbind(c(a,b),-.01),c("rv=1","rv=2"))
```

`rel.vert.mid.int` *The index of the vertex region in a middle interval that contains a given point*

Description

Returns the index of the vertex whose region contains point `p` in the interval `int`$=(a,b)=$(vertex 1,vertex 2) with (parameterized) center M_c associated with the centrality parameter $c \in (0,1)$; vertices of interval are labeled as 1 and 2 according to their order in the interval `int`. If the point, `p`, is not inside `int`, then the function yields `NA` as output. The corresponding vertex region is the interval (a,b) as (a,M_c) and (M_c,b) where $M_c = a + c(b-a)$.

See also (Ceyhan (2012, 2016)).

Usage

```
rel.vert.mid.int(p, int, c = 0.5)
```

Arguments

`p`	A 1D point. The vertex region `p` resides is to be found.
`int`	A `vector` of two real numbers representing an interval.
`c`	A positive real number in $(0,1)$ parameterizing the center inside `int`$=(a,b)$ with the default `c=.5`. For the interval, `int`$=(a,b)$, the parameterized center is $M_c = a + c(b-a)$.

Value

A `list` with two elements

`rv`	Index of the vertex in the interval `int` whose region contains point, `p`.
`int`	The vertices of the interval as a `vector` where position of the vertex corresponds to the vertex index as `int=(rv=1,rv=2)`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2012). “The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data.” *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). “Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity.” *REVSTAT*, **14(4)**, 349-394.

See Also

`rel.vert.end.int`

Examples

```
c<-.4
a<-0; b<-10; int = c(a,b)

Mc<-centerMc(int,c)

rel.vert.mid.int(6,int,c)

n<-20  #try also n<-40
xr<-range(a,b,Mc)
xf<-(int[2]-int[1])*.5
Xp<-runif(n,a,b)

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rel.vert.mid.int(Xp[i],int,c)$rv)
Rv

jit<-.1
yjit<-runif(n,-jit,jit)

Xlim<-range(a,b,Xp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(Mc,0),main="vertex region indices for the points", xlab=" ",
ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*range(yjit),pch=".",cex=3)
abline(h=0)
points(Xp,yjit)
abline(v=c(a,b,Mc),lty = 2,col=c(1,1,2))
text(Xp,yjit,labels=factor(Rv))
text(cbind(c(a,b,Mc),.02),c("rv=1","rv=2","Mc"))
```

`rel.vert.std.tri` *The index of the vertex region in the standard equilateral triangle that contains a given point*

Description

Returns the index of the vertex whose region contains point `p` in standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with vertex regions are constructed with center $M = (m_1,m_2)$ in Cartesian coordinates or $M = (\alpha,\beta,\gamma)$ in barycentric coordinates in the interior of T_e. (see the plots in the example for illustrations).

The vertices of triangle, T_e, are labeled as $1,2,3$ according to the row number the vertex is recorded in T_e. If the point, `p`, is not inside T_e, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, `M`, and projections from `M` to the edges on the lines joining vertices and `M`.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.vert.std.tri(p, M)
```

Arguments

`p`	A 2D point for which `M`-vertex region it resides in is to be determined in the standard equilateral triangle T_e.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e.

Value

A `list` with two elements

`rv`	Index of the vertex whose region contains point, `p`.
`tri`	The vertices of the triangle, T_e, where row number corresponds to the vertex index in `rv` with row $1 = (0,0)$, row $2 = (1,0)$, and row $3 = (1/2,\sqrt{3}/2)$.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.vert.std.triCM`, `rel.vert.tri`, `rel.vert.triCC`, `rel.vert.basic.triCC`, `rel.vert.triCM`, and `rel.vert.basic.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
n<-20  #try also n<-40

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

M<-as.numeric(runif.std.tri(1)$g)  #try also M<-c(.6,.2)

rel.vert.std.tri(Xp[1,],M)

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rel.vert.std.tri(Xp[i,],M)$rv)
Rv

Ds<-prj.cent2edges(Te,M)

Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Te)}
#need to run this when M is given in barycentric coordinates

plot(Te,asp=1,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(Xp,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Te,M)
xc<-txt[,1]+c(-.02,.03,.02,0)
yc<-txt[,2]+c(.02,.02,.03,.05)
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)
text(Xp,labels=factor(Rv))
```

`rel.vert.std.triCM` *The index of the CM-vertex region in the standard equilateral triangle that contains a given point*

Description

Returns the index of the vertex whose region contains point `p` in standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with vertex regions are constructed with center of mass CM (see the plots in the example for illustrations).

The vertices of triangle, T_e, are labeled as $1,2,3$ according to the row number the vertex is recorded in T_e. If the point, `p`, is not inside T_e, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, CM, and midpoints of the edges adjacent to the vertex.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.vert.std.triCM(p)
```

Arguments

`p`	A 2D point for which CM-vertex region it resides in is to be determined in the standard equilateral triangle T_e.

Value

A `list` with two elements

`rv`	Index of the vertex whose region contains point, `p`.
`tri`	The vertices of the triangle, T_e, where row number corresponds to the vertex index in `rv` with row $1 = (0,0)$, row $2 = (1,0)$, and row $3 = (1/2,\sqrt{3}/2)$.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.vert.basic.triCM`, `rel.vert.tri`, `rel.vert.triCC`, `rel.vert.basic.triCC`, `rel.vert.triCM`, and `rel.vert.basic.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)

n<-20  #try also n<-40

set.seed(1)
Xp<-runif.std.tri(n)$gen.points

rel.vert.std.triCM(Xp[1,])

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rel.vert.std.triCM(Xp[i,])$rv)
Rv

CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,asp=1,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(Xp,pch=".",col=1)
L<-matrix(rep(CM,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.02,.03,.02,0)
yc<-txt[,2]+c(.02,.02,.03,.05)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)
text(Xp,labels=factor(Rv))
```

`rel.vert.tetraCC` *The index of the CC-vertex region in a tetrahedron that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in a tetrahedron $th = T(A, B, C, D)$ and vertex regions are based on the circumcenter CC of `th`. (see the plots in the example for illustrations).

The vertices of the tetrahedron `th` are labeled as $1 = A$, $2 = B$, $3 = C$, and $4 = C$ also according to the row number the vertex is recorded in `th`.

If the point, `p`, is not inside `th`, then the function yields `NA` as output. The corresponding vertex region is the polygon whose interior points are closest to that vertex. If `th` is regular tetrahedron, then CC and CM (center of mass) coincide.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.vert.tetraCC(p, th)
```

Arguments

`p`	A 3D point for which CC-vertex region it resides in is to be determined in the tetrahedron `th`.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.

Value

A `list` with two elements

`rv`	Index of the CC-vertex region that contains point, `p` in the tetrahedron `th`
`tri`	The vertices of the tetrahedron, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`rel.vert.tetraCM` and `rel.vert.triCC`

Examples

```
set.seed(123)
A<-c(0,0,0)+runif(3,-.2,.2);
B<-c(1,0,0)+runif(3,-.2,.2);
C<-c(1/2,sqrt(3)/2,0)+runif(3,-.2,.2);
D<-c(1/2,sqrt(3)/6,sqrt(6)/3)+runif(3,-.2,.2);
tetra<-rbind(A,B,C,D)

n<-20  #try also n<-40

Xp<-runif.tetra(n,tetra)$g

rel.vert.tetraCC(Xp[1,],tetra)

Rv<-vector()
for (i in 1:n)
 Rv<-c(Rv,rel.vert.tetraCC(Xp[i,],tetra)$rv)
Rv

CC<-circumcenter.tetra(tetra)
CC

Xlim<-range(tetra[,1],Xp[,1],CC[1])
Ylim<-range(tetra[,2],Xp[,2],CC[2])
Zlim<-range(tetra[,3],Xp[,3],CC[3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(tetra[,1],tetra[,2],tetra[,3],
phi =0,theta=40, bty = "g",
main="Scatterplot of data points \n and CC-vertex regions",
xlim=Xlim+xd*c(-.05,.05), ylim=Ylim+yd*c(-.05,.05),
zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],
add=TRUE,lwd=2)
#add the data points
plot3D::points3D(Xp[,1],Xp[,2],Xp[,3],pch=".",cex=3, add=TRUE)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3],
labels=c("A","B","C","D"), add=TRUE)
plot3D::text3D(CC[1],CC[2],CC[3], labels=c("CC"), add=TRUE)

D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2;
D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-matrix(rep(CC,6),ncol=3,byrow=TRUE)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],
add=TRUE,lty = 2)

F1<-intersect.line.plane(A,CC,B,C,D)
```

```
L<-matrix(rep(F1,4),ncol=3,byrow=TRUE); R<-rbind(D4,D5,D6,CC)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=2,
add=TRUE,lty = 2)

F2<-intersect.line.plane(B,CC,A,C,D)
L<-matrix(rep(F2,4),ncol=3,byrow=TRUE); R<-rbind(D2,D3,D6,CC)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=3,
add=TRUE,lty = 2)

F3<-intersect.line.plane(C,CC,A,B,D)
L<-matrix(rep(F3,4),ncol=3,byrow=TRUE); R<-rbind(D3,D5,D6,CC)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=4,
add=TRUE,lty = 2)

F4<-intersect.line.plane(D,CC,A,B,C)
L<-matrix(rep(F4,4),ncol=3,byrow=TRUE); R<-rbind(D1,D2,D4,CC)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=5,
add=TRUE,lty = 2)

plot3D::text3D(Xp[,1],Xp[,2],Xp[,3], labels=factor(Rv), add=TRUE)
```

`rel.vert.tetraCM` *The index of the CM-vertex region in a tetrahedron that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in a tetrahedron $th = T(A, B, C, D)$ and vertex regions are based on the center of mass $CM = (A+B+C+D)/4$ of `th`. (see the plots in the example for illustrations).

The vertices of the tetrahedron `th` are labeled as $1 = A$, $2 = B$, $3 = C$, and $4 = C$ also according to the row number the vertex is recorded in `th`.

If the point, `p`, is not inside `th`, then the function yields `NA` as output. The corresponding vertex region is the simplex with the vertex, CM, and midpoints of the edges adjacent to the vertex.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.vert.tetraCM(p, th)
```

Arguments

`p`	A 3D point for which CM-vertex region it resides in is to be determined in the tetrahedron `th`.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.

Value

A `list` with two elements

`rv`	Index of the CM-vertex region that contains point, `p` in the tetrahedron `th`
`th`	The vertices of the tetrahedron, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`rel.vert.tetraCC` and `rel.vert.triCM`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0);
D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-20  #try also n<-40

Xp<-runif.std.tetra(n)$g

rel.vert.tetraCM(Xp[1,],tetra)

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv, rel.vert.tetraCM(Xp[i,],tetra)$rv )
Rv

Xlim<-range(tetra[,1],Xp[,1])
Ylim<-range(tetra[,2],Xp[,2])
Zlim<-range(tetra[,3],Xp[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

CM<-apply(tetra,2,mean)

plot3D::scatter3D(tetra[,1],tetra[,2],tetra[,3], phi =0,theta=40, bty = "g",
```

```
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
         pch = 20, cex = 1, ticktype = "detailed")
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
#add the data points
plot3D::points3D(Xp[,1],Xp[,2],Xp[,3],pch=".",cex=3, add=TRUE)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3],
labels=c("A","B","C","D"), add=TRUE)
plot3D::text3D(CM[1],CM[2],CM[3], labels=c("CM"), add=TRUE)

D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-matrix(rep(CM,6),ncol=3,byrow=TRUE)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty = 2)

F1<-intersect.line.plane(A,CM,B,C,D)
L<-matrix(rep(F1,4),ncol=3,byrow=TRUE); R<-rbind(D4,D5,D6,CM)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=2,
add=TRUE,lty = 2)

F2<-intersect.line.plane(B,CM,A,C,D)
L<-matrix(rep(F2,4),ncol=3,byrow=TRUE); R<-rbind(D2,D3,D6,CM)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=3,
add=TRUE,lty = 2)

F3<-intersect.line.plane(C,CM,A,B,D)
L<-matrix(rep(F3,4),ncol=3,byrow=TRUE); R<-rbind(D3,D5,D6,CM)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=4,
add=TRUE,lty = 2)

F4<-intersect.line.plane(D,CM,A,B,C)
L<-matrix(rep(F4,4),ncol=3,byrow=TRUE); R<-rbind(D1,D2,D4,CM)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=5,
add=TRUE,lty = 2)

plot3D::text3D(Xp[,1],Xp[,2],Xp[,3], labels=factor(Rv), add=TRUE)
```

`rel.vert.tri` *The index of the vertex region in a triangle that contains a given point*

Description

Returns the index of the related vertex in the triangle, `tri`, whose region contains point p.

Vertex regions are based on the general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`. Vertices of the triangle `tri` are labeled according to the row number the vertex is recorded.

If the point, `p`, is not inside `tri`, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, `M`, and projections from `M` to the edges on the lines joining vertices and `M` (see the illustration in the examples).

See also (Ceyhan (2005, 2010)).

Usage

```
rel.vert.tri(p, tri, M)
```

Arguments

`p`	A 2D point for which `M`-vertex region it resides in is to be determined in the triangle `tri`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.

Value

A `list` with two elements

`rv`	Index of the vertex whose region contains point, `p`; index of the vertex is the same as the row number in the triangle, `tri`
`tri`	The vertices of the triangle, `tri`, where row number corresponds to the vertex index `rv` with `rv=1` is row 1, `rv=2` is row 2, and $rv = 3$ is is row 3.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.vert.triCM`, `rel.vert.triCC`, `rel.vert.basic.triCC`, `rel.vert.basic.triCM`, `rel.vert.basic.tri`, and `rel.vert.std.triCM`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
M<-c(1.6,1.0)

P<-c(1.5,1.6)
rel.vert.tri(P,Tr,M)
#try also rel.vert.tri(P,Tr,M=c(2,2))
#center is not in the interior of the triangle

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

Rv<-vector()
for (i in 1:n)
{Rv<-c(Rv,rel.vert.tri(Xp[i,],Tr,M)$rv)}
Rv

Ds<-prj.cent2edges(Tr,M)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",
main="Illustration of M-Vertex Regions\n in a Triangle",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.04,-.04,0)
yc<-txt[,2]+c(-.02,.04,.05,-.08)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
text(Xp,labels=factor(Rv))
```

`rel.vert.triCC` *The index of the CC-vertex region in a triangle that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in a triangle `tri`$= (A, B, C)$ and vertex regions are based on the circumcenter CC of `tri`. (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as $1 = A$, $2 = B$, and $3 = C$ also according to the row number the vertex is recorded in `tri`. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. The corresponding vertex region is the polygon whose interior points are closest to that vertex. If `tri` is equilateral triangle, then CC and CM (center of mass) coincide.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.vert.triCC(p, tri)
```

Arguments

`p`	A 2D point for which CC-vertex region it resides in is to be determined in the triangle `tri`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with two elements

`rv`	Index of the CC-vertex region that contains point, `p` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.vert.tri`, `rel.vert.triCM`, `rel.vert.basic.triCM`, `rel.vert.basic.triCC`, `rel.vert.basic.tri`, and `rel.vert.std.triCM`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(1.3,1.2)
rel.vert.triCC(P,Tr)

CC<-circumcenter.tri(Tr)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],CC[1])
Ylim<-range(Tr[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,asp=1,xlab="",ylab="",pch=".",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.07,.08,.06,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.03,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

RV1<-(A+.5*(D3-A)+A+.5*(D2-A))/2
RV2<-(B+.5*(D3-B)+B+.5*(D1-B))/2
RV3<-(C+.5*(D2-C)+C+.5*(D1-C))/2

txt<-rbind(RV1,RV2,RV3)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

n<-20  #try also n<-40
Xp<-runif.tri(n,Tr)$g

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rel.vert.triCC(Xp[i,],Tr)$rv)
Rv

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
```

```
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,asp=1,xlab="",ylab="",
main="Illustration of CC-Vertex Regions\n in a Triangle",
pch=".",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".")
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(Rv))

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.07,.08,.06,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.03,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
```

`rel.vert.triCM` *The index of the CM-vertex region in a triangle that contains a given point*

Description

Returns the index of the vertex whose region contains point `p` in the triangle `tri`$= (y_1, y_2, y_3)$ with vertex regions are constructed with center of mass $CM = (y_1 + y_2 + y_3)/3$ (see the plots in the example for illustrations).

The vertices of triangle, `tri`, are labeled as $1, 2, 3$ according to the row number the vertex is recorded in `tri`. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, CM, and midpoints of the edges adjacent to the vertex.

See (Ceyhan (2005, 2010))

Usage

```
rel.vert.triCM(p, tri)
```

Arguments

`p`	A 2D point for which CM-vertex region it resides in is to be determined in the triangle `tri`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with two elements

`rv`	Index of the CM-vertex region that contains point, `p` in the triangle `tri`.
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.vert.tri`, `rel.vert.triCC`, `rel.vert.basic.triCM`, `rel.vert.basic.triCC`, `rel.vert.basic.tri`, and `rel.vert.std.triCM`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.6,2);
Tr<-rbind(A,B,C);
P<-c(1.4,1.2)
rel.vert.triCM(P,Tr)

n<-20  #try also n<-40
Xp<-runif.tri(n,Tr)$g

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rel.vert.triCM(Xp[i,],Tr)$rv)
Rv

CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
```

```
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".")
L<-Ds; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)
text(Xp,labels=factor(Rv))

txt<-rbind(Tr,CM,D1,D2,D3)
xc<-txt[,1]+c(-.02,.02,.02,-.02,.02,-.01,-.01)
yc<-txt[,2]+c(-.02,-.04,.06,-.02,.02,.06,-.06)
txt.str<-c("rv=1","rv=2","rv=3","CM","D1","D2","D3")
text(xc,yc,txt.str)
```

`rel.verts.tri` *The indices of the vertex regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the vertices whose regions contain the points in data set `Xp` in a triangle `tri`$= T(A, B, C)$.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle to the edges on the extension of the lines joining `M` to the vertices or based on the circumcenter of `tri`. Vertices of triangle `tri` are labeled as $1, 2, 3$ according to the row number the vertex is recorded.

If a point in `Xp` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding vertex region is the polygon with the vertex, `M`, and projection points from `M` to the edges crossing the vertex (as the output of `prj.cent2edges(Tr,M)`) or CC-vertex region (see the examples for an illustration).

See also (Ceyhan (2005, 2011)).

Usage

```
rel.verts.tri(Xp, tri, M)
```

Arguments

`Xp`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`.

Value

A `list` with two elements

`rv`	Indices of the vertices whose regions contains points in `Xp`.
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.verts.triCM`, `rel.verts.triCC`, and `rel.verts.tri.nondegPE`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
M<-c(1.6,1.0)

P<-c(.4,.2)
rel.verts.tri(P,Tr,M)

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also  #M<-c(1.6,1.0)

rel.verts.tri(Xp,Tr,M)
rel.verts.tri(rbind(Xp,c(2,2)),Tr,M)

rv<-rel.verts.tri(Xp,Tr,M)
```

```
rv

ifelse(identical(M,circumcenter.tri(Tr)),
Ds<-rbind((B+C)/2,(A+C)/2,(A+B)/2),Ds<-prj.cent2edges(Tr,M))

Xlim<-range(Tr[,1],M[1],Xp[,1])
Ylim<-range(Tr[,2],M[2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",
main="Scatterplot of data points \n and M-vertex regions in a triangle",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(.02,.04,-.03,0)
yc<-txt[,2]+c(.07,.04,.05,-.07)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
text(Xp,labels=factor(rv$rv))
```

`rel.verts.tri.nondegPE` *The indices of the vertex regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the vertices whose regions contain the points in data set `Xp` in a triangle `tri`$=(A,B,C)$ and vertex regions are based on the center `cent` which yields nondegenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for expansion parameter `r` in $(1,1.5]$.

Vertices of triangle `tri` are labeled as $1,2,3$ according to the row number the vertex is recorded if a point in `Xp` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding vertex region is the polygon with the vertex, `cent`, and projection points on the edges. The center label `cent` values `1,2,3` correspond to the vertices M_1, M_2, and M_3; with default 1 (see the examples for an illustration).

See also (Ceyhan (2005, 2011)).

Usage

```
rel.verts.tri.nondegPE(Xp, tri, r, cent = 1)
```

Arguments

`Xp`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ for this function.
`cent`	Index of the center (as $1, 2, 3$ corresponding to M_1, M_2, M_3) which gives non-degenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for expansion parameter `r` in $(1, 1.5]$; default `cent=1`.

Value

A `list` with two elements

`rv`	Indices (i.e., a `vector` of indices) of the vertices whose region contains points in `Xp` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). “Extension of One-Dimensional Proximity Regions to Higher Dimensions.” *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.verts.triCM`, `rel.verts.triCC`, and `rel.verts.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
r<-1.35
cent<-2

P<-c(1.4,1.0)
rel.verts.tri.nondegPE(P,Tr,r,cent)

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.tri(n,Tr)$g

rel.verts.tri.nondegPE(Xp,Tr,r,cent)
rel.verts.tri.nondegPE(rbind(Xp,c(2,2)),Tr,r,cent)

rv<-rel.verts.tri.nondegPE(Xp,Tr,r,cent)

M<-center.nondegPE(Tr,r)[cent,];
Ds<-prj.nondegPEcent2edges(Tr,r,cent)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]+c(-.03,.05,.05)
yc<-Tr[,2]+c(-.06,.02,.05)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(.02,.04,-.03,0)
yc<-txt[,2]+c(.07,.03,.05,-.07)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
text(Xp,labels=factor(rv$rv))
```

`rel.verts.triCC` *The indices of the CC-vertex regions in a triangle that contains the points in a give data set.*

Description

Returns the indices of the vertices whose regions contain the points in data set `Xp` in a triangle `tri`$=(A, B, C)$ and vertex regions are based on the circumcenter CC of `tri`. (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as $1 = A$, $2 = B$, and $3 = C$ also according to the row number the vertex is recorded in `tri`. If a point in `Xp` is not inside `tri`, then the function yields `NA` as output. The corresponding vertex region is the polygon whose interior points are closest to that vertex. If `tri` is equilateral triangle, then CC and CM (center of mass) coincide.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.verts.triCC(Xp, tri)
```

Arguments

`Xp`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with two elements

`rv`	Indices (i.e., a `vector` of indices) of the vertices whose region contains points in `Xp` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). “An investigation of new graph invariants related to the domination number of random proximity catch digraphs.” *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.verts.triCM`, `rel.verts.tri`, and `rel.verts.tri.nondegPE`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(.4,.2)
rel.verts.triCC(P,Tr)

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.tri(n,Tr)$g

rel.verts.triCC(Xp,Tr)
rel.verts.triCC(rbind(Xp,c(2,2)),Tr)

(rv<-rel.verts.triCC(Xp,Tr))

CC<-circumcenter.tri(Tr)
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1],CC[1])
Ylim<-range(Tr[,2],Xp[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",asp=1,xlab="",ylab="",
main="Scatterplot of data points \n and the CC-vertex regions",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".",col=1)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(CC,Ds)
xc<-txt[,1]+c(.04,.04,-.03,0)
yc<-txt[,2]+c(-.07,.04,.06,-.08)
txt.str<-c("CC","D1","D2","D3")
text(xc,yc,txt.str)
```

```
text(Xp,labels=factor(rv$rv))
```

`rel.verts.triCM` *The indices of the CM-vertex regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the vertices whose regions contain the points in data set `Xp` in a triangle `tri`$= (A, B, C)$ and vertex regions are based on the center of mass CM of `tri`. (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as $1 = A$, $2 = B$, and $3 = C$ also according to the row number the vertex is recorded in `tri`. If a point in `Xp` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding vertex region is the polygon with the vertex, CM, and midpoints the edges crossing the vertex.

See also (Ceyhan (2005, 2010)).

Usage

```
rel.verts.triCM(Xp, tri)
```

Arguments

`Xp`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with two elements

`rv`	Indices (i.e., a `vector` of indices) of the vertices whose region contains points in `Xp` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rel.verts.tri`, `rel.verts.triCC`, and `rel.verts.tri.nondegPE`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(.4,.2)
rel.verts.triCM(P,Tr)

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.tri(n,Tr)$g

rv<-rel.verts.triCM(Xp,Tr)
rv

CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],Xp[,1])
Ylim<-range(Tr[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".",col=1)
L<-matrix(rep(CM,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]+c(-.04,.05,.05)
yc<-Tr[,2]+c(-.05,.05,.03)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)
```

```
txt<-rbind(CM,Ds)
xc<-txt[,1]+c(.04,.04,-.03,0)
yc<-txt[,2]+c(-.07,.04,.06,-.08)
txt.str<-c("CM","D1","D2","D3")
text(xc,yc,txt.str)
text(Xp,labels=factor(rv$rv))
```

`rel.verts.triM` *The alternative function for the indices of the M-vertex regions in a triangle that contains the points in a give data set*

Description

An alternative function to the function `rel.verts.tri` when the center M is not the circumcenter falling outside the triangle. This function only works for a center M in the interior of the triangle, with the projections of M to the edges along the lines joining M to the vertices.

Usage

```
rel.verts.triM(Xp, tri, M)
```

Arguments

`Xp`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.

Value

A `list` with two elements

`rv`	Indices of the vertices whose regions contains points in `Xp`.
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

Author(s)

Elvan Ceyhan

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

rel.verts.tri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
M<-c(1.6,1.0)

P<-c(.4,.2)
rel.verts.triM(P,Tr,M)

n<-20  #try also n<-40
set.seed(1)
Xp<-runif.tri(n,Tr)$g

M<-c(1.6,1.0)  #try also M<-c(1.3,1.3)

(rv<-rel.verts.tri(Xp,Tr,M))
rel.verts.triM(rbind(Xp,c(2,2)),Tr,M)

Ds<-prj.cent2edges(Tr,M)

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty = 2)

xc<-Tr[,1]+c(-.03,.05,.05)
yc<-Tr[,2]+c(-.06,.02,.05)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(.02,.04,-.03,0)
yc<-txt[,2]+c(.07,.03,.05,-.07)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
text(Xp,labels=factor(rv$rv))
```

`rseg.circular` *Generation of points segregated (in a radial or circular fashion) from a given set of points*

Description

An object of class `"Patterns"`. Generates `n` 2D points uniformly in $(a_1 - e, a_1 + e) \times (a_1 - e, a_1 + e) \setminus B(y_i, e)$ (a_1 and $b1$ are denoted as `a1` and `b1` as arguments) where $Y_p = (y_1, y_2, \ldots, y_{n_y})$ with n_y being number of `Yp` points for various values of `e` under the segregation pattern and $B(y_i, e)$ is the ball centered at y_i with radius `e`.

Positive values of `e` yield realizations from the segregation pattern and nonpositive values of `e` provide a type of complete spatial randomness (CSR), `e` should not be too large to make the support of generated points empty, `a1` is defaulted to the minimum of the x-coordinates of the `Yp` points, `a2` is defaulted to the maximum of the x-coordinates of the `Yp` points, `b1` is defaulted to the minimum of the y-coordinates of the `Yp` points, `b2` is defaulted to the maximum of the y-coordinates of the `Yp` points.

Usage

```
rseg.circular(
  n,
  Yp,
  e,
  a1 = min(Yp[, 1]),
  a2 = max(Yp[, 1]),
  b1 = min(Yp[, 2]),
  b2 = max(Yp[, 2])
)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`Yp`	A set of 2D points representing the reference points. The generated points are segregated (in a circular or radial fashion) from these points.
`e`	A positive real number representing the radius of the balls centered at `Yp` points. These balls are forbidden for the generated points (i.e., generated points would be in the complement of union of these balls).
`a1, a2`	Real numbers representing the range of x-coordinates in the region (default is the range of x-coordinates of the `Yp` points).
`b1, b2`	Real numbers representing the range of y-coordinates in the region (default is the range of y-coordinates of the `Yp` points).

Value

A `list` with the elements

`type`	The type of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	Radial (i.e., circular) exclusion parameter of the segregation pattern
`ref.points`	The input set of reference points `Yp`, i.e., points from which generated points are segregated.
`gen.points`	The output set of generated points segregated from `Yp` points
`tri.Yp`	Logical output for triangulation based on `Yp` points should be implemented or not. if `TRUE` triangulation based on `Yp` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., `Yp`) points.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated points

Author(s)

Elvan Ceyhan

See Also

`rassoc.circular`, `rseg.std.tri`, `rsegII.std.tri`, and `rseg.multi.tri`

Examples

```
nx<-100; ny<-4;  #try also nx<-1000; ny<-10
e<-.15;  #try also e<- -.1; #a negative e provides a CSR realization
#with default bounding box (i.e., unit square)
Y<-cbind(runif(ny),runif(ny))

Xdt<-rseg.circular(nx,Y,e)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

#with default bounding box (i.e., unit square)
Y<-cbind(runif(ny),runif(ny))
Xdt<-Xdt$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,pch=16,col=2,lwd=2, xlab="x",ylab="y",
     main="Circular Segregation of X points from Y Points",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
points(Xdt)

#with a rectangular bounding box
a1<-0; a2<-10;
```

```
b1<-0; b2<-5;
e<-1.5;
Y<-cbind(runif(ny,a1,a2),runif(ny,b1,b2))
#try also Y<-cbind(runif(ny,a1,a2/2),runif(ny,b1,b2/2))

Xdt<-rseg.circular(nx,Y,e,a1,a2,b1,b2)$gen.points
Xlim<-range(Xdt[,1],Y[,1]); Ylim<-range(Xdt[,2],Y[,2])

plot(Y,pch=16,asp=1,col=2,lwd=2, xlab="x",ylab="y",
     main="Circular Segregation of X points from Y Points",
     xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(Xdt)
```

`rseg.multi.tri` *Generation of points segregated (in a Type I fashion) from a given set of points*

Description

An object of class `"Patterns"`. Generates `n` points uniformly in the support for Type I segregation in the convex hull of set of points, `Yp`.

`delta` is the parameter of segregation (that is, $\delta 100$ % of the area around each vertex in each Delaunay triangle is forbidden for point generation). `delta` corresponds to `eps` in the standard equilateral triangle T_e as $delta = 4eps^2/3$ (see `rseg.std.tri` function).

If `Yp` consists only of 3 points, then the function behaves like the function `rseg.tri`.

`DTmesh` must be the Delaunay triangulation of `Yp` and `DTr` must be the corresponding Delaunay triangles (both `DTmesh` and `DTr` are `NULL` by default). If `NULL`, `DTmesh` is computed via `tri.mesh` and `DTr` is computed via `triangles` function in `interp` package.

`tri.mesh` function yields the triangulation nodes with their neighbours, and creates a triangulation object, and `triangles` function yields a triangulation data structure from the triangulation object created by `tri.mesh` (the first three columns are the vertex indices of the Delaunay triangles.)

See (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for more on the segregation pattern. Also, see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
rseg.multi.tri(n, Yp, delta, DTmesh = NULL, DTr = NULL)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`Yp`	A set of 2D points from which Delaunay triangulation is constructed.

`delta`	A positive real number in $(0, 1)$. `delta` is the parameter of segregation (that is, $\delta 100$ each Delaunay triangle is forbidden for point generation).
`DTmesh`	Delaunay triangulation of `Yp`, default is `NULL`, which is computed via `tri.mesh` function in `interp` package. `tri.mesh` function yields the triangulation nodes with their neighbours, and creates a triangulation object.
`DTr`	Delaunay triangles based on `Yp`, default is `NULL`, which is computed via `tri.mesh` function in `interp` package. `triangles` function yields a triangulation data structure from the triangulation object created by `tri.mesh`.

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	Exclusion parameter, `delta`, of the Type I segregation pattern. `delta` is in $(0, 1)$ and $\delta 100$ % area around vertices of each Delaunay triangle is forbidden for point generation.
`ref.points`	The input set of points `Yp`; reference points, i.e., points from which generated points are segregated.
`gen.points`	The output set of generated points segregated from `Yp` points.
`tri.Y`	Logical output, `TRUE`, if triangulation based on `Yp` points should be implemented.
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., `Yp`) points.
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). “S-hull: a fast radial sweep-hull routine for Delaunay triangulation.” 1604.01428.

See Also

`rseg.circular`, `rseg.std.tri`, `rsegII.std.tri`, and `rassoc.multi.tri`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-100; ny<-4;  #try also nx<-1000; ny<-10;

set.seed(1)
Yp<-cbind(runif(ny),runif(ny))
del<-.4

Xdt<-rseg.multi.tri(nx,Yp,del)
Xdt
summary(Xdt)
plot(Xdt)

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")
#Delaunay triangulation based on Y points
TRY<-interp::triangles(DTY)[,1:3];
Xp<-rseg.multi.tri(nx,Yp,del,DTY,TRY)$gen.points
#data under CSR in the convex hull of Ypoints

Xlim<-range(Yp[,1])
Ylim<-range(Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot of the data in the convex hull of Y points together with the Delaunay triangulation
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")
#Delaunay triangulation based on Y points

oldpar <- par(pty="s")
plot(Xp,main="Points from Type I Segregation \n in Multipe Triangles",
xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),
ylim=Ylim+yd*c(-.05,.05),type="n")
interp::plot.triSht(DTY, add=TRUE,
do.points=TRUE,col="blue")
points(Xp,pch=".",cex=3)
par(oldpar)
```

`rseg.std.tri` *Generation of points segregated (in a Type I fashion) from the vertices of T_e*

Description

An object of class `"Patterns"`. Generates `n` points uniformly in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ under the type I segregation alternative for `eps` in $(0, \sqrt{3}/3 = 0.5773503]$.

In the type I segregation, the triangular forbidden regions around the vertices are determined by the parameter `eps` which serves as the height of these triangles (see examples for a sample plot.)

See also (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)).

Usage

```
rseg.std.tri(n, eps)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`eps`	A positive real number representing the parameter of type I segregation (which is the height of the triangular forbidden regions around the vertices).

Value

A `list` with the elements

`type`	The type of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	The exclusion parameter, `eps`, of the segregation pattern, which is the height of the triangular forbidden regions around the vertices
`ref.points`	The input set of points `Y`; reference points, i.e., points from which generated points are segregated (i.e., vertices of T_e).
`gen.points`	The output set of generated points segregated from `Y` points (i.e., vertices of T_e).
`tri.Y`	Logical output for triangulation based on `Y` points should be implemented or not. if `TRUE` triangulation based on `Y` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., `Y`) points, which is 3 here.
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the vertices of T_e here.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2011). “Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family.” *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Marchette DJ (2007). “A new family of random graphs for testing spatial segregation.” *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). “Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association.” *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`rseg.circular`, `rassoc.circular`, `rsegII.std.tri`, and `rseg.multi.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-100
eps<-.3  #try also .15, .5, .75

set.seed(1)
Xdt<-rseg.std.tri(n,eps)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Xp<-Xdt$gen.points

plot(Te,asp=1,pch=".",xlab="",ylab="",
main="Type I segregation in the \n standard equilateral triangle",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(Xp)

#The support for the Type I segregation alternative
sr<-eps/(sqrt(3)/2)
C1<-C+sr*(A-C); C2<-C+sr*(B-C)
A1<-A+sr*(B-A); A2<-A+sr*(C-A)
B1<-B+sr*(A-B); B2<-B+sr*(C-B)
supp<-rbind(A1,B1,B2,C2,C1,A2)
```

```
plot(Te,asp=1,pch=".",xlab="",ylab="",
main="Support of the Type I Segregation",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
if (sr<=.5)
{
  polygon(Te)
  polygon(supp,col=5)
} else
{
  polygon(Te,col=5,lwd=2.5)
  polygon(rbind(A,A1,A2),col=0,border=NA)
  polygon(rbind(B,B1,B2),col=0,border=NA)
  polygon(rbind(C,C1,C2),col=0,border=NA)
}
points(Xp)
```

`rseg.tri` *Generation of points segregated (in a Type I fashion) from the vertices of a triangle*

Description

An object of class `"Patterns"`. Generates `n` points uniformly in the support for Type I segregation in a given triangle, `tri`.

`delta` is the parameter of segregation (that is, $\delta 100$ % of the area around each vertex in the triangle is forbidden for point generation). `delta` corresponds to `eps` in the standard equilateral triangle T_e as $delta = 4eps^2/3$ (see `rseg.std.tri` function).

See (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for more on the segregation pattern.

Usage

```
rseg.tri(n, tri, delta)
```

Arguments

`n`	A positive integer representing the number of points to be generated from the segregation pattern in the triangle, `tri`.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.
`delta`	A positive real number in $(0, 1)$. `delta` is the parameter of segregation (that is, $\delta 100$ % area around vertices of each Delaunay triangle is forbidden for point generation).

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	Exclusion parameter, `delta`, of the Type I segregation pattern. `delta` is in $(0, 1)$ and $\delta 100$ % area around vertices of the triangle `tri` is forbidden for point generation.
`ref.points`	The input set of points, i.e., vertices of `tri`; reference points, i.e., points from which generated points are segregated.
`gen.points`	The output set of generated points segregated from the vertices of `tri`.
`tri.Y`	Logical output, if `TRUE` the triangle `tri` is also plotted when the corresponding plot function from the `Patterns` object is called.
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., vertex of `tri`, which is 3 here).
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the vertices of the triangle `tri`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`rassoc.tri`, `rseg.std.tri`, `rsegII.std.tri`, and `rseg.multi.tri`

Examples

```
n<-100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)
del<-.4

Xdt<-rseg.tri(n,Tr,del)
```

```
Xdt
summary(Xdt)
plot(Xdt)

Xp<-Xdt$g
Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",
main="Points from Type I Segregation \n in one Triangle",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp)
xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.03)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)
```

`rsegII.std.tri` *Generation of points segregated (in a Type II fashion) from the vertices of T_e*

Description

An object of class "Patterns". Generates n points uniformly in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ under the type II segregation alternative for eps in $(0, \sqrt{3}/6 = 0.2886751]$.

In the type II segregation, the annular forbidden regions around the edges are determined by the parameter eps which is the distance from the interior triangle (i.e., support for the segregation) to T_e (see examples for a sample plot.)

Usage

```
rsegII.std.tri(n, eps)
```

Arguments

n	A positive integer representing the number of points to be generated.
eps	A positive real number representing the parameter of type II segregation (which is the distance from the interior triangle points to the boundary of T_e).

Value

A `list` with the elements

`type`	The type of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`parameters`	The exclusion parameter, `eps`, of the segregation pattern, which is the distance from the interior triangle to T_e
`ref.points`	The input set of points `Y`; reference points, i.e., points from which generated points are segregated (i.e., vertices of T_e).
`gen.points`	The output set of generated points segregated from `Y` points (i.e., vertices of T_e).
`tri.Y`	Logical output for triangulation based on `Y` points should be implemented or not. if `TRUE` triangulation based on `Y` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of reference (i.e., `Y`) points, which is 3 here.
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the vertices of T_e here

Author(s)

Elvan Ceyhan

See Also

`rseg.circular`, `rassoc.circular`, `rseg.std.tri`, and `rseg.multi.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10  #try also n<-20 or n<-100 or 1000
eps<-.15  #try also .2

set.seed(1)
Xdt<-rsegII.std.tri(n,eps)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Xp<-Xdt$gen.points

plot(Te,pch=".",xlab="",ylab="",
```

```
main="Type II segregation in the \n standard equilateral triangle",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(Xp)

#The support for the Type II segregation alternative
C1<-c(1/2,sqrt(3)/2-2*eps);
A1<-c(eps*sqrt(3),eps); B1<-c(1-eps*sqrt(3),eps);
supp<-rbind(A1,B1,C1)

plot(Te,asp=1,pch=".",xlab="",ylab="",
main="Support of the Type II Segregation",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
  polygon(Te)
  polygon(supp,col=5)
points(Xp)
```

`runif.basic.tri` *Generation of Uniform Points in the standard basic triangle*

Description

An object of class "Uniform". Generates `n` points uniformly in the standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan et al. (2006)). Hence, standard basic triangle is useful for simulation studies under the uniformity hypothesis.

Usage

```
runif.basic.tri(n, c1, c2)
```

Arguments

`n`	A positive integer representing the number of uniform points to be generated in the standard basic triangle.
`c1, c2`	Positive real numbers representing the top vertex in standard basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, c_1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The "`main`" title for the plot of the point pattern

`tess.points`	The vertices of the support of the uniformly generated points, it is the standard basic triangle T_b for this function
`gen.points`	The output set of generated points uniformly in the standard basic triangle
`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 3).
`txt4pnts`	Description of the two numbers in `num.points`.
`xlimit`, `ylimit`	The ranges of the x- and y-coordinates of the support, Tb

Author(s)

Elvan Ceyhan

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe CE, Marchette DJ (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe CE, Wierman JC (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics & Data Analysis*, **50(8)**, 1925-1964.

See Also

`runif.std.tri`, `runif.tri`, and `runif.multi.tri`

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);
n<-100

set.seed(1)
runif.basic.tri(1,c1,c2)
Xdt<-runif.basic.tri(n,c1,c2)
Xdt
summary(Xdt)
plot(Xdt)

Xp<-runif.basic.tri(n,c1,c2)$g

Xlim<-range(Tb[,1])
Ylim<-range(Tb[,2])
```

```
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,xlab="",ylab="",xlim=Xlim+xd*c(-.01,.01),
ylim=Ylim+yd*c(-.01,.01),type="n")
polygon(Tb)
points(Xp)
```

`runif.multi.tri` *Generation of Uniform Points in the Convex Hull of Points*

Description

An object of class `"Uniform"`. Generates `n` points uniformly in the Convex Hull of set of points, `Yp`. That is, generates uniformly in each of the triangles in the Delaunay triangulation of `Yp`, i.e., in the multiple triangles partitioning the convex hull of `Yp`.

If `Yp` consists only of 3 points, then the function behaves like the function `runif.tri`.

`DTmesh` is the Delaunay triangulation of `Yp`, default is `DTmesh=NULL`. `DTmesh` yields triangulation nodes with neighbours (result of `tri.mesh` function from `interp` package).

See (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
runif.multi.tri(n, Yp, DTmesh = NULL)
```

Arguments

`n`	A positive integer representing the number of uniform points to be generated in the convex hull of the point set `Yp`.
`Yp`	A set of 2D points whose convex hull is the support of the uniform points to be generated.
`DTmesh`	Triangulation nodes with neighbours (result of `tri.mesh` function from `interp` package).

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`tess.points`	The points which constitute the vertices of the triangulation and whose convex hull determines the support of the generated points.
`gen.points`	The output set of generated points uniformly in the convex hull of `Yp`

`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of vertices in the triangulation (i.e., size of `Yp`) points.
`txt4pnts`	Description of the two numbers in `num.points`
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the points in Yp

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`runif.tri`, `runif.std.tri`, and `runif.basic.tri`,

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-100; ny<-4;  #try also nx<-1000; ny<-10;
set.seed(1)
Yp<-cbind(runif(ny,0,10),runif(ny,0,10))

Xdt<-runif.multi.tri(nx,Yp)
#data under CSR in the convex hull of Ypoints
Xdt
summary(Xdt)
plot(Xdt)

Xp<-Xdt$g
#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")
#Delaunay triangulation based on Y points
Xp<-runif.multi.tri(nx,Yp,DTY)$g
#data under CSR in the convex hull of Ypoints

Xlim<-range(Yp[,1])
Ylim<-range(Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
#plot of the data in the convex hull of Y points together with the Delaunay triangulation
plot(Xp, xlab=" ", ylab=" ",
main="Uniform Points in Convex Hull of Y Points",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),type="n")
interp::plot.triSht(DTY, add=TRUE,
do.points = TRUE,pch=16,col="blue")
points(Xp,pch=".",cex=3)

Yp<-rbind(c(.3,.2),c(.4,.5),c(.14,.15))
runif.multi.tri(nx,Yp)
```

`runif.std.tetra` *Generation of Uniform Points in the Standard Regular Tetrahedron T_h*

Description

An object of class `"Uniform"`. Generates `n` points uniformly in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3))$.

Usage

```
runif.std.tetra(n)
```

Arguments

`n`	A positive integer representing the number of uniform points to be generated in the standard regular tetrahedron T_h.

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`tess.points`	The vertices of the support region of the uniformly generated points, it is the standard regular tetrahedron T_h for this function
`gen.points`	The output set of generated points uniformly in the standard regular tetrahedron T_h.
`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 4).
`txt4pnts`	Description of the two numbers in `num.points`
`xlimit`, `ylimit`, `zlimit`	The ranges of the x-, y-, and z-coordinates of the support, T_h

Author(s)

Elvan Ceyhan

See Also

`runif.tetra`, `runif.tri`, and `runif.multi.tri`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-100

set.seed(1)
Xdt<-runif.std.tetra(n)
Xdt
summary(Xdt)
plot(Xdt)

Xp<-runif.std.tetra(n)$g

Xlim<-range(tetra[,1])
Ylim<-range(tetra[,2])
Zlim<-range(tetra[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Xp[,1],Xp[,2],Xp[,3],
phi =20,theta=15, bty = "g", pch = 20, cex = 1,
ticktype = "detailed",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),
zlim=Zlim+zd*c(-.05,.05))
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],
add=TRUE,lwd=2)

plot3D::text3D(tetra[,1]+c(.05,0,0,0),tetra[,2],tetra[,3],
labels=c("A","B","C","D"), add=TRUE)

s3d<-scatterplot3d::scatterplot3d(Xp, highlight.3d=TRUE,xlab="x",
ylab="y",zlab="z", col.axis="blue", col.grid="lightblue",
               main="3D Scatterplot of the data", pch=20)
s3d$points3d(tetra,pch=20,col="blue")
```

`runif.std.tri` *Generation of Uniform Points in the Standard Equilateral Triangle*

Description

An object of class `"Uniform"`. Generates `n` points uniformly in the standard equilateral triangle $T_e = T(A, B, C)$ with vertices $A = (0, 0)$, $B = (1, 0)$, and $C = (1/2, \sqrt{3}/2)$.

Usage

```
runif.std.tri(n)
```

Arguments

`n`	A positive integer representing the number of uniform points to be generated in the standard equilateral triangle T_e.

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`tess.points`	The vertices of the support region of the uniformly generated points, it is the standard equilateral triangle T_e for this function
`gen.points`	The output set of generated points uniformly in the standard equilateral triangle T_e.
`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 3).
`txt4pnts`	Description of the two numbers in `num.points`
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the support, T_e

Author(s)

Elvan Ceyhan

See Also

`runif.basic.tri`, `runif.tri`, and `runif.multi.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-100

set.seed(1)
Xdt<-runif.std.tri(n)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Xp<-runif.std.tri(n)$gen.points
plot(Te,asp=1,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.01,.01),
ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(Xp)
```

`runif.std.tri.onesixth` *Generation of Uniform Points in the first one-sixth of standard equilateral triangle*

Description

An object of class `"Uniform"`. Generates `n` points uniformly in the first 1/6th of the standard equilateral triangle $T_e = (A, B, C)$ with vertices with $A = (0, 0)$; $B = (1, 0)$, $C = (1/2, \sqrt{3}/2)$ (see the examples below). The first 1/6th of the standard equilateral triangle is the triangle with vertices $A = (0, 0)$, $(1/2, 0)$, $C = (1/2, \sqrt{3}/6)$.

Usage

```
runif.std.tri.onesixth(n)
```

Arguments

`n`	a positive integer representing number of uniform points to be generated in the first one-sixth of T_e.

Value

A `list` with the elements

`type`	The type of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`support`	The vertices of the support of the uniformly generated points
`gen.points`	The output set of uniformly generated points in the first 1/6th of the standard equilateral triangle.
`out.region`	The outer region for the one-sixth of T_e, which is just T_e here.
`desc.pat`	Description of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of vertices of the support (i.e., `Y`) points.
`txt4pnts`	Description of the two numbers in `num.points`.
`xlimit`, `ylimit`	The ranges of the x- and y-coordinates of the generated, support and outer region points

Author(s)

Elvan Ceyhan

See Also

`runif.std.tri`, `runif.basic.tri`, `runif.tri`, and `runif.multi.tri`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
CM<-(A+B+C)/3;
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
nx<-100  #try also nx<-1000

#data generation step
set.seed(1)
Xdt<-runif.std.tri.onesixth(nx)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xd<-Xdt$gen.points

#plot of the data with the regions in the equilateral triangle
Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
plot(Te,asp=1,pch=".",xlim=Xlim+xd*c(-.01,.01),
ylim=Ylim+yd*c(-.01,.01),xlab=" ",ylab=" ",
     main="first 1/6th of the \n standard equilateral triangle")
polygon(Te)
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
polygon(rbind(A,D3,CM),col=5)
points(Xd)

#letter annotation of the plot
txt<-rbind(A,B,C,CM,D1,D2,D3)
xc<-txt[,1]+c(-.02,.02,.02,.04,.05,-.03,0)
yc<-txt[,2]+c(.02,.02,.02,.03,0,.03,-.03)
txt.str<-c("A","B","C","CM","D1","D2","D3")
text(xc,yc,txt.str)
```

`runif.tetra` *Generation of Uniform Points in a tetrahedron*

Description

An object of class `"Uniform"`. Generates `n` points uniformly in the general tetrahedron `th` whose vertices are stacked row-wise.

Usage

```
runif.tetra(n, th)
```

Arguments

`n`	A positive integer representing the number of uniform points to be generated in the tetrahedron.
`th`	A 4×3 matrix with each row representing a vertex of the tetrahedron.

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`tess.points`	The vertices of the support of the uniformly generated points, it is the tetrahedron' `th` for this function
`gen.points`	The output set of generated points uniformly in the tetrahedron, `th`.
`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated

`num.points`	The `vector` of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 4).
`txt4pnts`	Description of the two numbers in `num.points`
`xlimit`, `ylimit`, `zlimit`	The ranges of the x-, y-, and z-coordinates of the support, `th`

Author(s)

Elvan Ceyhan

See Also

`runif.std.tetra` and `runif.tri`

Examples

```
A<-sample(1:12,3); B<-sample(1:12,3);
C<-sample(1:12,3); D<-sample(1:12,3)
tetra<-rbind(A,B,C,D)

n<-100

set.seed(1)
Xdt<-runif.tetra(n,tetra)
Xdt
summary(Xdt)
plot(Xdt)

Xp<-Xdt$g

Xlim<-range(tetra[,1],Xp[,1])
Ylim<-range(tetra[,2],Xp[,2])
Zlim<-range(tetra[,3],Xp[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Xp[,1],Xp[,2],Xp[,3],
theta =225, phi = 30, bty = "g",
main="Uniform Points in a Tetrahedron",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),
zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],
add=TRUE,lwd=2)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3],
labels=c("A","B","C","D"), add=TRUE)
```

```
s3d<-scatterplot3d::scatterplot3d(Xp, highlight.3d=TRUE,
xlab="x",ylab="y",zlab="z", col.axis="blue", col.grid="lightblue",
                    main="3D Scatterplot of the data", pch=20)
 s3d$points3d(tetra,pch=20,col="blue")
```

`runif.tri` *Generation of Uniform Points in a Triangle*

Description

An object of class `"Uniform"`. Generates `n` points uniformly in a given triangle, `tri`

Usage

```
runif.tri(n, tri)
```

Arguments

`n`	A positive integer representing the number of uniform points to be generated in the triangle.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The `"main"` title for the plot of the point pattern
`tess.points`	The vertices of the support of the uniformly generated points, it is the triangle `tri` for this function
`gen.points`	The output set of generated points uniformly in the triangle, `tri`.
`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated
`num.points`	The `vector` of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 3).
`txt4pnts`	Description of the two numbers in `num.points`
`xlimit, ylimit`	The ranges of the x- and y-coordinates of the support, `tri`

Author(s)

Elvan Ceyhan

See Also

`runif.std.tri`, `runif.basic.tri`, and `runif.multi.tri`

Examples

```
n<-100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)

Xdt<-runif.tri(n,Tr)
Xdt
summary(Xdt)
plot(Xdt)

Xp<-Xdt$g
Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
plot(Tr,pch=".",xlab="",ylab="",main="Uniform Points in One Triangle",
     xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Xp)
xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)
```

`seg.tri.support` *The auxiliary triangle to define the support of type I segregation*

Description

Returns the triangle whose intersection with a general triangle gives the support for type I segregation given the `delta` (i.e., $\delta 100$ % area of a triangle around the vertices is chopped off). See the plot in the examples.

Caveat: the vertices of this triangle may be outside the triangle, `tri`, depending on the value of `delta` (i.e., for small values of `delta`).

Usage

```
seg.tri.support(delta, tri)
```

Arguments

`delta`	A positive real number between 0 and 1 that determines the percentage of area of the triangle around the vertices forbidden for point generation.
`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

the vertices of the triangle (stacked row-wise) whose intersection with a general triangle gives the support for type I segregation for the given `delta`

Author(s)

Elvan Ceyhan

See Also

`rseg.std.tri` and `rseg.multi.tri`

Examples

```
#for a general triangle
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
delta<-.3  #try also .5,.75,.85
Tseg<-seg.tri.support(delta,Tr)

Xlim<-range(Tr[,1],Tseg[,1])
Ylim<-range(Tr[,2],Tseg[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

oldpar <- par(pty="s")
plot(Tr,pch=".",xlab="",ylab="",
main="segregation support is the intersection\n of these two triangles",
axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
polygon(Tseg,lty=2)

txt<-rbind(Tr,Tseg)
xc<-txt[,1]+c(-.03,.03,.03,.06,.04,-.04)
yc<-txt[,2]+c(.02,.02,.04,-.03,0,0)
txt.str<-c("A","B","C","T1","T2","T3")
text(xc,yc,txt.str)
par(oldpar)
```

`six.extremaTe` *The closest points among a data set in the standard equilateral triangle to the median lines in the six half edge regions*

Description

An object of class `"Extrema"`. Returns the six closest points among the data set, `Xp`, in the standard equilateral triangle $T_e = T(A = (0,0), B = (1,0), C = (1/2, \sqrt{3}/2))$ in half edge regions. In particular, in regions r_1 and r_6, it finds the closest point in each region to the line segment $[A, CM]$ in regions r_2 and r_3, it finds the closest point in each region to the line segment $[B, CM]$ and in regions r_4 and r_5, it finds the closest point in each region to the line segment $[C, CM]$ where $CM = (A + B + C)/3$ is the center of mass.

See the example for this function or example for `index.six.Te` function. If there is no data point in region r_i, then it returns "`NA NA`" for i-th row in the extrema. `ch.all.intri` is for checking whether all data points are in T_e (default is `FALSE`).

Usage

```
six.extremaTe(Xp, ch.all.intri = FALSE)
```

Arguments

`Xp`	A set of 2D points among which the closest points in the standard equilateral triangle to the median lines in 6 half edge regions.
`ch.all.intri`	A logical argument for checking whether all data points are in T_e (default is `FALSE`).

Value

A `list` with the elements

`txt1`	Region labels as r1-r6 (correspond to row number in Extremum Points).
`txt2`	A short description of the distances as `"Distances to Line Segments (A,CM), (B,CM),and (C,CM) in the six regions r1-r6"`.
`type`	Type of the extrema points
`mtitle`	The `"main"` title for the plot of the extrema
`ext`	The extrema points, here, closest points in each of regions `r1-r6` to the line segments joining vertices to the center of mass, CM.
`X`	The input data, `Xp`, can be a `matrix` or `data frame`
`num.points`	The number of data points, i.e., size of `Xp`
`supp`	Support of the data points, here, it is T_e.
`cent`	The center point used for construction of edge regions.
`ncent`	Name of the center, `cent`, it is center of mass `"CM"` for this function.
`regions`	The six regions, `r1-r6` and edge regions inside the triangle, T_e, provided as a list.
`region.names`	Names of the regions as `"r1"-"r6"` and names of the edge regions as `"er=1"`, `"er=2"`, and `"er=3"`.
`region.centers`	Centers of mass of the regions `r1-r6` and of edge regions inside T_e.
`dist2ref`	Distances from closest points in each of regions `r1-r6` to the line segments joining vertices to the center of mass, CM.

Author(s)

Elvan Ceyhan

See Also

`index.six.Te` and `cl2edges.std.tri`

Examples

```
n<-20  #try also n<-100
Xp<-runif.std.tri(n)$gen.points

Ext<-six.extremaTe(Xp)
Ext
summary(Ext)
plot(Ext)

sixt<-Ext

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

h1<-c(1/2,sqrt(3)/18); h2<-c(2/3, sqrt(3)/9); h3<-c(2/3, 2*sqrt(3)/9);
h4<-c(1/2, 5*sqrt(3)/18); h5<-c(1/3, 2*sqrt(3)/9); h6<-c(1/3, sqrt(3)/9);

r1<-(h1+h6+CM)/3;r2<-(h1+h2+CM)/3;r3<-(h2+h3+CM)/3;
r4<-(h3+h4+CM)/3;r5<-(h4+h5+CM)/3;r6<-(h5+h6+CM)/3;

Xlim<-range(Te[,1],Xp[,1])
Ylim<-range(Te[,2],Xp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),
ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
polygon(rbind(h1,h2,h3,h4,h5,h6))
points(Xp)
points(sixt$ext,pty=2,pch=4,col="red")

txt<-rbind(Te,r1,r2,r3,r4,r5,r6)
xc<-txt[,1]+c(-.02,.02,.02,0,0,0,0,0,0)
yc<-txt[,2]+c(.02,.02,.03,0,0,0,0,0,0)
txt.str<-c("A","B","C","1","2","3","4","5","6")
text(xc,yc,txt.str)
```

`slope` *The slope of a line*

Description

Returns the slope of the line joining two distinct 2D points `a` and `b`.

Usage

```
slope(a, b)
```

Arguments

`a, b`	2D points that determine the straight line (i.e., through which the straight line passes).

Value

Slope of the line joining 2D points `a` and `b`

Author(s)

Elvan Ceyhan

See Also

`Line`, `paraline`, and `perpline`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)
slope(A,B)

slope(c(1,2),c(2,3))
```

`summary.Extrema` *Return a summary of a* `Extrema` *object*

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the `type` of the extrema (i.e. the description of the extrema), extrema points, distances from extrema to the reference `object` (e.g. boundary of a triangle), some of the data points (from which extrema is found).

Usage

```
## S3 method for class 'Extrema'
summary(object, ...)
```

Arguments

`object`	An `object` of class `Extrema`.
`...`	Additional parameters for `summary`.

Value

The `call` of the `object` of class `"Extrema"`, the `type` of the extrema (i.e. the description of the extrema), extrema points, distances from extrema to the reference `object` (e.g. boundary of a triangle), some of the data points (from which extrema is found).

See Also

`print.Extrema`, `print.summary.Extrema`, and `plot.Extrema`

Examples

```
n<-10
Xp<-runif.std.tri(n)$gen.points
Ext<-cl2edges.std.tri(Xp)
Ext
summary(Ext)
```

`summary.Lines` *Return a summary of a* `Lines` object

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the defining `points`, selected x and y points on the line, equation of the line, and `coefficients` of the line.

Usage

```
## S3 method for class 'Lines'
summary(object, ...)
```

Arguments

`object`	An `object` of class `Lines`.
`...`	Additional parameters for `summary`.

Value

The `call` of the `object` of class `"Lines"`, the defining `points`, selected x and y points on the line, equation of the line, and `coefficients` of the line (in the form: `y = slope * x + intercept`).

See Also

`print.Lines`, `print.summary.Lines`, and `plot.Lines`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)
xr<-range(A,B);
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=3) #try also l=10, 20 or 100

lnAB<-Line(A,B,x)
lnAB
summary(lnAB)
```

`summary.Lines3D` *Return a summary of a* `Lines3D` object

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the defining vectors (i.e., initial and direction vectors), selected x, y, and z points on the line, equation of the line (in parametric form), and `coefficients` of the line.

Usage

```
## S3 method for class 'Lines3D'
summary(object, ...)
```

Arguments

`object`	An `object` of class `Lines3D`.
`...`	Additional parameters for `summary`.

Value

`call` of the function defining the `object`, the defining vectors (i.e., initial and direction vectors), selected x, y, and z points on the line, equation of the line (in parametric form), and `coefficients` of the line (for the form: `x=x0 + A*t`, `y=y0 + B*t`, and `z=z0 + C*t`).

See Also

`print.Lines3D`, `print.summary.Lines3D`, and `plot.Lines3D`

Examples

```
P<-c(1,10,3); Q<-c(1,1,3);
vecs<-rbind(P,Q)
Line3D(P,Q,.1)
Line3D(P,Q,.1,dir.vec=FALSE)

tr<-range(vecs);
tf<-(tr[2]-tr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=3) #try also l=10, 20 or 100

lnPQ3D<-Line3D(P,Q,tsq)
lnPQ3D
summary(lnPQ3D)
```

`summary.NumArcs` *Return a summary of a* `NumArcs` object

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the description of the output, `desc`: number of arcs in the proximity catch digraph (PCD) and related quantities in the induced subdigraphs for points in the Delaunay cells. In the one Delaunay cell case, the function provides the total number of arcs in the digraph, vertices of Delaunay cell, and indices of target points in the Delaunay cell.

In the multiple Delaunay cell case, the function provides total number of arcs in the digraph, number of arcs for the induced digraphs for points in the Delaunay cells, vertices of Delaunay cells or indices of points that form the the Delaunay cells, indices of target points in the convex hull of nontarget points, indices of Delaunay cells in which points reside, and area or length of the the Delaunay cells.

Usage

```
## S3 method for class 'NumArcs'
summary(object, ...)
```

Arguments

`object`	An `object` of class `NumArcs`.
`...`	Additional parameters for `summary`.

Value

The `call` of the `object` of class `"NumArcs"`, the `desc` of the output: total number of arcs in the digraph. Moreover, in the one Delaunay cell case, the function also provides vertices of Delaunay cell, and indices of target points in the Delaunay cell; and in the multiple Delaunay cell case, it also provides number of arcs for the induced subdigraphs for points in the Delaunay cells, vertices of Delaunay cells or indices of points that form the the Delaunay cells, indices of target points in the convex hull of nontarget points, indices of Delaunay cells in which points reside, and area or length of the the Delaunay cells.

See Also

`print.NumArcs`, `print.summary.NumArcs`, and `plot.NumArcs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10
Xp<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)
Arcs<-arcsAStri(Xp,Tr,M)
Arcs
summary(Arcs)
```

`summary.Patterns` *Return a summary of a* `Patterns` object

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the `type` of the pattern, `parameters` of the pattern, study window, some sample points from the generated pattern, reference points (if any for the bivariate pattern), and number of points for each class

Usage

```
## S3 method for class 'Patterns'
summary(object, ...)
```

Arguments

`object`	An `object` of class `Patterns`.
`...`	Additional parameters for `summary`.

Value

The `call` of the `object` of class `"Patterns"`, the `type` of the pattern, `parameters` of the pattern, study window, some sample points from the generated pattern, reference points (if any for the bivariate pattern), and number of points for each class

See Also

`print.Patterns`, `print.summary.Patterns`, and `plot.Patterns`

Examples

```
nx<-10; #try also 10, 100, and 1000
ny<-5; #try also 1
e<-.15;
Y<-cbind(runif(ny),runif(ny))
#with default bounding box (i.e., unit square)

Xdt<-rseg.circular(nx,Y,e)
Xdt
summary(Xdt)
```

`summary.PCDs` *Return a summary of a* `PCDs` `object`

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the `type` of the proximity catch digraph (PCD), (i.e. the description of the PCD), some of the partition (i.e. intervalization in the 1D case and triangulation in the 2D case) points (i.e., vertices of the intervals or the triangles), parameter(s) of the PCD, and various quantities (number of vertices, number of arcs and arc density of the PCDs, number of vertices for the partition and number of partition cells (i.e., intervals or triangles)).

Usage

```
## S3 method for class 'PCDs'
summary(object, ...)
```

Arguments

`object`	An `object` of class `PCDs`.
`...`	Additional parameters for `summary`.

Value

The `call` of the `object` of class `"PCDs"`, the `type` of the proximity catch digraph (PCD), (i.e. the description of the PCD), some of the partition (i.e. intervalization in the 1D case and triangulation in the 2D case) points (i.e., vertices of the intervals or the triangles), parameter(s) of the PCD, and various quantities (number of vertices, number of arcs and arc density of the PCDs, number of vertices for the partition and number of partition cells (i.e., intervals or triangles)).

See Also

`print.PCDs`, `print.summary.PCDs`, and `plot.PCDs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10
Xp<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)
Arcs<-arcsAStri(Xp,Tr,M)
Arcs
summary(Arcs)
```

`summary.Planes` *Return a summary of a* `Planes` `object`

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the defining 3D `points`, selected x, y, and z points on the plane, equation of the plane, and `coefficients` of the plane.

Usage

```
## S3 method for class 'Planes'
summary(object, ...)
```

Arguments

`object`	An `object` of class `Planes`.
`...`	Additional parameters for `summary`.

Value

The `call` of the `object` of class `"Planes"`, the defining 3D `points`, selected x, y, and z points on the plane, equation of the plane, and `coefficients` of the plane (in the form: `z = A*x + B*y + C`).

See Also

`print.Planes`, `print.summary.Planes`, and `plot.Planes`

Examples

```
P<-c(1,10,3); Q<-c(1,1,3); C<-c(3,9,12)
pts<-rbind(P,Q,C)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1
#how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1
#how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=5) #try also l=10, 20 or 100
y<-seq(yr[1]-yf,yr[2]+yf,l=5) #try also l=10, 20 or 100

plPQC<-Plane(P,Q,C,x,y)
plPQC
summary(plPQC)
```

`summary.TriLines` *Return a summary of a* `TriLines` object

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the defining `points`, selected x and y points on the line, equation of the line, together with the vertices of the triangle, and `coefficients` of the line.

Usage

```
## S3 method for class 'TriLines'
summary(object, ...)
```

Arguments

`object`	An `object` of class `TriLines`.
`...`	Additional parameters for `summary`.

Value

The `call` of the `object` of class `"TriLines"`, the defining `points`, selected x and y points on the line, equation of the line, together with the vertices of the triangle, and `coefficients` of the line (in the form: `y` = `slope` * `x` + `intercept`).

See Also

`print.TriLines`, `print.summary.TriLines`, and `plot.TriLines`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25
#how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,l=3)

lnACM<-lineA2CMinTe(x)
lnACM
summary(lnACM)
```

`summary.Uniform` *Return a summary of a* `Uniform` object

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the `type` of the pattern (i.e. the description of the uniform distribution), study window, vertices of the support of the Uniform distribution, some sample points generated from the uniform distribution, and the number of points (i.e., number of generated points and the number of vertices of the support of the uniform distribution.)

Usage

```
## S3 method for class 'Uniform'
summary(object, ...)
```

Arguments

`object`	An `object` of class `Uniform`.
`...`	Additional parameters for `summary`.

Value

The `call` of the `object` of class `"Uniform"`, the `type` of the pattern (i.e. the description of the uniform distribution), study window, vertices of the support of the Uniform distribution, some sample points generated from the uniform distribution, and the number of points (i.e., number of generated points and the number of vertices of the support of the uniform distribution.)

See Also

`print.Uniform`, `print.summary.Uniform`, and `plot.Uniform`

Examples

```
n<-10 #try also 20, 100, and 1000
A<-c(1,1); B<-c(2,0); R<-c(1.5,2);
Tr<-rbind(A,B,R)

Xdt<-runif.tri(n,Tr)
Xdt
summary(Xdt)
```

`swamptrees` *Tree Species in a Swamp Forest*

Description

Locations and species classification of trees in a plot in the Savannah River, SC, USA. Locations are given in meters, rounded to the nearest 0.1 decimal. The data come from a one-hectare (200-by-50m) plot in the Savannah River Site. The 734 mapped stems included 156 Carolina ashes (Fraxinus caroliniana), 215 water tupelos (Nyssa aquatica), 205 swamp tupelos (Nyssa sylvatica), 98 bald cypresses (Taxodium distichum) and 60 stems from 8 additional three species (labeled as Others (OT)). The plots were set up by Bill Good and their spatial patterns described in (Good and Whipple (1982)), the plots have been maintained and resampled by Rebecca Sharitz and her colleagues of the Savannah River Ecology Laboratory. The data and some of its description are borrowed from the swamp data entry in the `dixon` package in the CRAN repository.

See also (Good and Whipple (1982); Jones et al. (1994); Dixon (2002)).

Usage

```
data(swamptrees)
```

Format

A data frame with 734 rows and 4 variables

Details

Text describing the variable (i.e., column) names in the data set.

- x,y: x and y (i.e., Cartesian) coordinates of the trees
- live: a categorical variable that indicates the tree is alive (labeled as 1) or dead (labeled as 0)
- sp: species label of the trees:

  FX: Carolina ash (Fraxinus caroliniana)
  NS: Swamp tupelo (Nyssa sylvatica)
  NX: Water tupelo (Nyssa aquatica)
  TD: Bald cypress (Taxodium distichum)
  OT: Other species

Source

Prof. Philip Dixon's website

References

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

Good BJ, Whipple SA (1982). “Tree spatial patterns: South Carolina bottomland and swamp forests.” *Bulletin of the Torrey Botanical Club*, **109(4)**, 529-536.

Jones RH, Sharitz RR, James SM, Dixon PM (1994). “Tree population dynamics in seven South Carolina mixed-species forests.” *Bulletin of the Torrey Botanical Club*, **121(4)**, 360-368.

Examples

```
data(swamptrees)
plot(swamptrees$x,swamptrees$y, col=as.numeric(swamptrees$sp),pch=19,
xlab='',ylab='',main='Swamp Trees')
```

`tri2std.basic.tri` *Converting a triangle to the standard basic triangle form form*

Description

This function transforms any triangle, `tri`, to the standard basic triangle form.

The standard basic triangle form is $T_b = T((0,0),(1,0),(c_1,c_2))$ where c_1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the standard basic triangle form by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence, standard basic triangle form is useful for simulation studies under the uniformity hypothesis.

Usage

```
tri2std.basic.tri(tri)
```

Arguments

`tri`	A 3×2 matrix with each row representing a vertex of the triangle.

Value

A `list` with two elements

`Cvec`	The nontrivial vertex $C = (c_1, c_2)$ in the standard basic triangle form T_b.
`orig.order`	Row order of the input triangle, `tri`, when converted to the standard basic triangle form T_b

Author(s)

Elvan Ceyhan

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);

tri2std.basic.tri(rbind(A,B,C))
tri2std.basic.tri(rbind(B,C,A))

A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
tri2std.basic.tri(rbind(A,B,C))
tri2std.basic.tri(rbind(A,C,B))
tri2std.basic.tri(rbind(B,A,C))
```

`Xin.convex.hullY` *Points from one class inside the convex hull of the points from the other class*

Description

Given two 2D data sets, `Xp` and `Yp`, it returns the `Xp` points inside the convex hull of `Yp` points.

See (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
Xin.convex.hullY(Xp, Yp)
```

Arguments

`Xp`	A set of 2D points which constitute the data set.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.

Value

`Xp` points inside the convex hull of `Yp` points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`plotDelaunay.tri`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-5;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,.25),runif(ny,0,.25))+cbind(c(0,0,0.5,1,1),c(0,1,.5,0,1))
#try also Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

DT<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Xch<-Xin.convex.hullY(Xp,Yp)

plot(Xp,main=" ", xlab=" ", ylab=" ",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),pch=".",cex=3)
interp::convex.hull(DT,plot.it = TRUE, add = TRUE)  # or try polygon(Yp[ch$i,])
points(Xch,pch=4,col="red")
```

Index