Package 'baseverse'

April 13, 2026

Title Modern Base-R Functions

Version 0.1.0

Description Includes modern base-R functions. Functions beginning with p_ are wrapper functions to existing base-R functions, supporting native piping. base_match() and base_when() mimic case_match() and case_when() from 'dplyr' but return a factor by default with levels ordered according to user input. et() mimics count() from 'dplyr'.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

LazyData true

NeedsCompilation no

Author Yea-Hung Chen [aut, cre]

Maintainer Yea-Hung Chen <yea-hung.chen@ucsf.edu>

Depends R (>= 4.1.0)

Repository CRAN

Date/Publication 2026-04-13 14:30:08 UTC

Contents

base_match 2
base_when 2
et 3
nhanes 4
p_cor 5
p_glm 5
p_lm 6
p_t.test 7
p_table 8
p_wilcox.test 9

Index **11**

base_match *Defines factors using value-label mapping*

Description

A base-R approximation of `case_match()` (from 'dplyr'). Unlike `case_match()`, `base_when()` returns a factor. The levels will be ordered according to the order included in `...` (see below).

Usage

```
base_match(original_variable, ..., as_factor = TRUE, string_for_na = "")
```

Arguments

`original_variable`	the original variable
`...`	the codebook, specified as a named vector, with each element in `'label'=level` format, with the levels listed in the desired order
`as_factor`	logical, controlling whether the function should return a factor
`string_for_na`	string value that will be converted to NA

Value

a factor

Examples

```
# load data
data(nhanes)

# define country
nhanes<-nhanes |> transform(
  country=base_match(dmdborn4,'USA'=1,'Other'=2)
)
```

base_when *Defines factors using logical objects*

Description

A base-R approximation of `case_when()` (from 'dplyr'). Unlike `case_when()`, `base_when()` returns a factor. The levels will be ordered according to the order included in `...` (see below).

Usage

```
base_when(..., as_factor = TRUE, string_for_na = "")
```

Arguments

...	conditions for defining the replacement values, specified as a named list, with each element in 'label'=logical_vector format, with the levels listed in the desired order
as_factor	logical, controlling whether the function should return a factor
string_for_na	string value that will be converted to NA

Value

a factor

Examples

```
# load data
data(nhanes)

nhanes<-nhanes |>
  transform(
    cholesterol=base_when(
      'Desirable' = (lbxtc<200),
      'Borderline high' = (lbxtc>=200)&(lbxtc<240),
      'High' = (lbxtc>=240)
    )
  )
```

---

et *Tables variables*

Description

Creates a table for a variable, mimicking count() (from 'dplyr'). ET stands for exploratory table.

Usage

```
et(data_frame, variable_name)
```

Arguments

data_frame	the data.frame
variable_name	the variable name

Value

a data frame

Examples

```
# load data
data(nhanes)

# define country
nhanes<-nhanes |> transform(
  country=base_match(nhanes$dmdborn4,'USA'=1,'Other'=2)
)

# examine variable, with native piping
nhanes |> et(country)

# examine variable, with dollar-sign notation
et(nhanes$country)
```

---

nhanes *National Health and Nutrition Examination Survey data*

Description

A subset of data from the National Health and Nutrition Examination Survey

Usage

```
nhanes
```

Format

nhanes**:**

A data frame with 8,153 rows and 3 columns:

seqn a unique identifier

riagendr gender

ridageyr age

dmqmiliz military status

dmdborn4 country of birth

lbxtc total cholesterol

bpxosy1 systolic blood pressure

bpxosy2 systolic blood pressure

bpxodi1 diastolic blood pressure

bpxodi2 diastolic blood pressure

mcq010 asthma

smq020 smoking status

Source

https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?Cycle=2021-2023

p_cor *Finds correlation coefficients, with support for piping*

Description

A wrapper function to `cor()`, with support for piping.

Usage

```
p_cor(data, x, y, ...)
```

Arguments

`data`	the data
`x`	one of the two variables
`y`	one of the two variables
`...`	additional arguments passed to `stats::cor()`.

Value

a numeric object

Examples

```
# load the data
data(nhanes)

# get correlation coefficient for systolic and diastolic
nhanes |> p_cor(bpxosy1,bpxodi1,use='complete.obs')
```

p_glm *Fits generalized linear models, with support for piping*

Description

A wrapper function to `glm()`, with the `data` argument listed first to support piping.

Usage

```
p_glm(data, formula, ...)
```

Arguments

`data`	the data
`formula`	the formula
`...`	additional arguments passed to `stats::glm()`.

Value

a glm object

Examples

```
# load the data
data(nhanes)

# define asthma
nhanes<-nhanes |> transform(
  asthma=base_match(mcq010,'No'=2,'History of asthma'=1)
)

# define smoking
nhanes<-nhanes |> transform(
  smoking=base_match(smq020,'No'=2,'History of smoking'=1)
)

# fit a model
mymodel<-nhanes |> p_glm(asthma~smoking,family=binomial(link='logit'))

# obtain model details
mymodel |> summary()

# obtain confidence interval
mymodel |> confint() |> exp()
```

---

p_lm *Fits linear models, with support for piping*

Description

A wrapper function to lm(), with the data argument listed first to support native piping.

Usage

```
p_lm(data, formula, ...)
```

Arguments

data	the data
formula	the formula
...	additional arguments passed to stats::lm().

Value

an lm object

Examples

```
# load the data
data(nhanes)

# define country
nhanes<-nhanes |> transform(
  country=base_match(dmdborn4,'USA'=1,'Other'=2)
)

# fit a model
mymodel<-nhanes |> p_lm(bpxosy1~country)

# obtain model details
mymodel |> summary()

# obtain confidence interval
mymodel |> confint()
```

`p_t.test` *Conducts t-tests, with support for piping*

Description

A wrapper function to `t.test()`, with the `data` argument listed first to support piping.

Usage

```
p_t.test(data, x = NULL, y = NULL, formula = NULL, ...)
```

Arguments

`data`	the data
`x`	one of two variables
`y`	one of two variables
`formula`	a formula
`...`	additional arguments passed to `stats::t.test()`.

Value

an `htest` object

Examples

```
# load the data
data(nhanes)

# define smoking
nhanes<-nhanes |> transform(
  smoking=base_match(smq020,'No'=2,'History of smoking'=1)
)

# conduct a one-sample t-test
nhanes |> p_t.test(bpxosy1)

# conduct a two-sample t-test, using formula notation
nhanes |> p_t.test(bpxosy1~smoking)

# conduct a two-sample t-test, using formula notation
nhanes |> p_t.test(formula=bpxosy1~smoking)

# conduct a paired t-test, using x and y
nhanes |> p_t.test(bpxosy1,bpxosy2,paired=TRUE)
```

---

p_table *Tables variables, with support for piping*

Description

A wrapper function to `table()`, with support for piping.

Usage

```
p_table(data, x, y = NULL, ...)
```

Arguments

`data`	the data
`x`	one variable
`y`	another variable
`...`	additional arguments passed to `table()`.

Value

a `table` object

Examples

```
# load the data
data(nhanes)

# define smoking
nhanes<-nhanes |> transform(
  smoking=base_match(smq020,'No'=2,'History of smoking'=1)
)

# define asthma
nhanes<-nhanes |> transform(
  asthma=base_match(mcq010,'Asthma'=1,'No'=2)
)

# table smoking
nhanes |> p_table(smoking)

# table smoking and asthma
nhanes |> p_table(smoking,asthma)
```

`p_wilcox.test` *Conducts Wilcoxon rank-sum tests, with support for piping*

Description

A wrapper function to `wilcox.test()`, with the `data` argument listed first to support piping.

Usage

```
p_wilcox.test(data, x = NULL, y = NULL, formula = NULL, ...)
```

Arguments

`data`	the data
`x`	one of two variables
`y`	one of two variables
`formula`	a formula
`...`	additional arguments passed to `stats::wilcox.test()`.

Value

an `htest` object

Examples

```
# load the data
data(nhanes)

# define smoking
nhanes<-nhanes |> transform(
  smoking=base_match(smq020,'No'=2,'History of smoking'=1)
)

# conduct a one-sample wilcoxon test
nhanes |> p_wilcox.test(bpxosy1)

# conduct a two-sample wilcoxon test, using formula notation
nhanes |> p_wilcox.test(bpxosy1~smoking)

# conduct a two-sample wilcoxon test, using formula notation
nhanes |> p_wilcox.test(formula=bpxosy1~smoking)

# conduct a paired wilcoxon test, using x and y
nhanes |> p_wilcox.test(bpxosy1,bpxosy2,paired=TRUE)
```

Index

∗ **datasets**
nhanes, 4

base_match, 2
base_when, 2

et, 3

nhanes, 4

p_cor, 5
p_glm, 5
p_lm, 6
p_t.test, 7
p_table, 8
p_wilcox.test, 9

stats::cor(), *5*
stats::glm(), *5*
stats::lm(), *6*
stats::t.test(), *7*
stats::wilcox.test(), *9*

table(), *8*