Package 'gdtools'

August 26, 2025

Title Utilities for Graphical Rendering and Fonts Management

Version 0.4.3

Description Tools are provided to compute metrics of formatted strings and to check the availability of a font. Another set of functions is provided to support the collection of fonts from 'Google Fonts' in a cache. Their use is simple within 'R Markdown' documents and 'shiny' applications but also with graphic productions generated with the 'ggiraph', 'ragg' and 'svglite' packages or with tabular productions from the 'flextable' package.

License GPL-3 | file LICENSE

URL https://davidgohel.github.io/gdtools/

BugReports https://github.com/davidgohel/gdtools/issues

Depends R (>= 4.0.0)

Imports fontquiver (>= 0.2.0), htmltools, Rcpp (>= 0.12.12), systemfonts (>= 1.1.0), tools

Suggests curl, gfonts, methods, testthat

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.3.2

SystemRequirements cairo, freetype2, fontconfig

NeedsCompilation yes

Author David Gohel [aut, cre],
Hadley Wickham [aut],
Lionel Henry [aut],
Jeroen Ooms [aut] (ORCID: <https://orcid.org/0000-0002-4035-0289>),
Yixuan Qiu [ctb],
R Core Team [cph] (Cairo code from X11 device),
ArData [cph],
RStudio [cph]

Maintainer David Gohel <david.gohel@ardata.fr>

Repository CRAN

Date/Publication 2025-08-26 07:40:02 UTC

Contents

`addGFontHtmlDependency` *Use a font in Shiny or Markdown*

Description

Add an empty HTML element attached to an 'HTML Dependency' containing the css and the font files so that the font is available in the HTML page. Multiple families are supported.

The htmlDependency is defined with function `gfontHtmlDependency()`.

Usage

```
addGFontHtmlDependency(family = "Open Sans", subset = c("latin", "latin-ext"))
```

Arguments

`family` family name of a 'Google Fonts', for example, "Open Sans", "Roboto", "Fira Code" or "Fira Sans Condensed". Complete list is available with the following command:

```
gfonts::get_all_fonts()$family |>
  unlist() |>
  unique() |>
  sort()
```

subset — font subset, a character vector, it defaults to only "latin" and "latin-ext" and can contains values such as "greek", "emoji", "chinese-traditional",
Run the following code to get a complete list:

```
gfonts::get_all_fonts()$subsets |> unlist() |> unique() |> sort()
```

Details

It allows users to use fonts from 'Google Fonts' in an HTML page generated by 'shiny' or 'R Markdown'. At the first request, the font files will be downloaded and stored in a cache on the user's machine, thus avoiding many useless downloads or allowing to work with these fonts afterwards without an Internet connection, in a docker image for example. See `fonts_cache_dir()`.

The server delivering the font files should not be too busy. That's why a one second pause is added after each download to respect the server's limits. This time can be set with the option `GFONTS_DOWNLOAD_SLEEPTIME` which must be a number of seconds.

Value

an HTML object

See Also

Other functions for font management: `fonts_cache_dir()`, `gfontHtmlDependency()`, `install_gfont_script()`, `installed_gfonts()`, `liberationsansHtmlDependency()`, `register_gfont()`, `register_liberationsans()`

Examples

```
## Not run:
if (check_gfonts()) {
  dummy_setup()
  addGFontHtmlDependency(family = "Open Sans")
}

## End(Not run)
```

---

fontconfig_reinit *reload Fontconfig configuration*

Description

This function can be used to make fontconfig reload font configuration files.

Usage

```
fontconfig_reinit()
```

Note

Fontconfig is not used anymore and that function will be deprecated in the next release.

Author(s)

Paul Murrell

`fonts_cache_dir` *manage font working directory*

Description

Initialize or remove font directory used to store downloaded font files.

This directory is managed by R function `R_user_dir()` but can also be defined in a non-user location by setting ENV variable `GDTOOLS_CACHE_DIR` or by setting R option `GDTOOLS_CACHE_DIR`.

Its value can be read with the `fonts_cache_dir()` function.

The directory can be deleted with `rm_fonts_cache()` and created with `init_fonts_cache()`.

Usage

```
fonts_cache_dir()

rm_fonts_cache()

init_fonts_cache()
```

See Also

Other functions for font management: `addGFontHtmlDependency()`, `gfontHtmlDependency()`, `install_gfont_script()`, `installed_gfonts()`, `liberationsansHtmlDependency()`, `register_gfont()`, `register_liberationsans()`

Examples

```
fonts_cache_dir()

options(GDTOOLS_CACHE_DIR = tempdir())
fonts_cache_dir()
options(GDTOOLS_CACHE_DIR = NULL)

Sys.setenv(GDTOOLS_CACHE_DIR = tempdir())
fonts_cache_dir()
Sys.setenv(GDTOOLS_CACHE_DIR = "")



init_fonts_cache()
dir.exists(fonts_cache_dir())

rm_fonts_cache()
dir.exists(fonts_cache_dir())
```

font_family_exists *Check if font family exists.*

Description

Check if a font family exists in system fonts.

Usage

```
font_family_exists(font_family = "sans")
```

Arguments

font_family	font family name (case sensitive)

Value

A logical value

Examples

```
font_family_exists("sans")
font_family_exists("Arial")
font_family_exists("Courier")
```

gfontHtmlDependency *'Google Font' HTML dependency*

Description

Create an HTML dependency ready to be used in 'Shiny' or 'R Markdown'.

Usage

```
gfontHtmlDependency(family = "Open Sans", subset = c("latin", "latin-ext"))
```

Arguments

family	family name of a 'Google Fonts', for example, "Open Sans", "Roboto", "Fira Code" or "Fira Sans Condensed". Complete list is available with the following command:
	`gfonts::get_all_fonts()$family |>`
	`  unlist() |>`
	`  unique() |>`
	`  sort()`

subset — font subset, a character vector, it defaults to only "latin" and "latin-ext" and can contains values such as "greek", "emoji", "chinese-traditional",
Run the following code to get a complete list:

```
gfonts::get_all_fonts()$subsets |> unlist() |> unique() |> sort()
```

Details

It allows users to use fonts from 'Google Fonts' in an HTML page generated by 'shiny' or 'R Markdown'. At the first request, the font files will be downloaded and stored in a cache on the user's machine, thus avoiding many useless downloads or allowing to work with these fonts afterwards without an Internet connection, in a docker image for example. See `fonts_cache_dir()`.

The server delivering the font files should not be too busy. That's why a one second pause is added after each download to respect the server's limits. This time can be set with the option `GFONTS_DOWNLOAD_SLEEPTIME` which must be a number of seconds.

Value

an object defined with `htmltools::htmlDependency()`.

See Also

Other functions for font management: `addGFontHtmlDependency()`, `fonts_cache_dir()`, `install_gfont_script()`, `installed_gfonts()`, `liberationsansHtmlDependency()`, `register_gfont()`, `register_liberationsans()`

Examples

```
## Not run:
if (check_gfonts()) {
  dummy_setup()
  gfontHtmlDependency(family = "Open Sans")
}

## End(Not run)
```

---

glyphs_match *Validate glyph entries*

Description

Determines if strings contain glyphs not part of a font.

Usage

```
glyphs_match(x, fontname = "sans", bold = FALSE, italic = FALSE, fontfile = "")
```

Arguments

x	Character vector of strings
fontname	Font name
bold, italic	Is text bold/italic?
fontfile	Font file

Value

a logical vector, if a character element is containing at least a glyph that can not be matched in the font table, FALSE is returned.

Examples

```
glyphs_match(letters)
glyphs_match("\u265E", bold = TRUE)
```

---

installed_gfonts *List installed 'Google Fonts'*

Description

List installed 'Google Fonts' that can be found in the user cache directory.

Usage

```
installed_gfonts()
```

Value

families names as a character vector

See Also

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), liberationsansHtmlDependency(), register_gfont(), register_liberationsans()

Examples

```
## Not run:
if (check_gfonts()) {
  dummy_setup()
  register_gfont(family = "Roboto")
  installed_gfonts()
}

## End(Not run)
```

`install_gfont_script` *Shell command to install a font from 'Google Fonts'*

Description

Create a string containing a system command to execute so that the font from 'Google Fonts' is installed on the system. Its execution may require root permissions, in dockerfile for example.

Usage

```
install_gfont_script(
  family = "Open Sans",
  subset = c("latin", "latin-ext"),
  platform = c("debian", "windows", "macos"),
  file = NULL
)
```

Arguments

`family`	family name of a 'Google Fonts', for example, "Open Sans", "Roboto", "Fira Code" or "Fira Sans Condensed". Complete list is available with the following command:
	`gfonts::get_all_fonts()$family |>`
	`  unlist() |>`
	`  unique() |>`
	`  sort()`

`subset`	font subset, a character vector, it defaults to only "latin" and "latin-ext" and can contains values such as "greek", "emoji", "chinese-traditional", Run the following code to get a complete list: `gfonts::get_all_fonts()$subsets |> unlist() |> unique() |> sort()`
`platform`	"debian" and "windows" and "macos" are supported.
`file`	script file to generate, optional. If the parameter is specified, a file will be generated ready for execution. If the platform is Windows, administration rights are required to run the script.

Details

It allows users to use fonts from 'Google Fonts' in an HTML page generated by 'shiny' or 'R Markdown'. At the first request, the font files will be downloaded and stored in a cache on the user's machine, thus avoiding many useless downloads or allowing to work with these fonts afterwards without an Internet connection, in a docker image for example. See `fonts_cache_dir()`.

The server delivering the font files should not be too busy. That's why a one second pause is added after each download to respect the server's limits. This time can be set with the option `GFONTS_DOWNLOAD_SLEEPTIME` which must be a number of seconds.

Value

the 'shell' or 'PowerShell' command as a string

See Also

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), installed_gfonts(), liberationsansHtmlDependency(), register_gfont(), register_liberationsans()

Examples

```
## Not run:
if (check_gfonts()) {
  dummy_setup()
  install_gfont_script(family = "Roboto", platform = "macos")
}

## End(Not run)
```

---

liberationsansHtmlDependency

'Liberation Sans' Font HTML dependency

Description

Create an HTML dependency ready to be used in 'Shiny' or 'R Markdown' with 'Liberation Sans' Font.

Usage

```
liberationsansHtmlDependency()
```

See Also

gfontHtmlDependency()

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), installed_gfonts(), register_gfont(), register_liberationsans()

match_family *Find best family match with systemfonts*

Description

`match_family()` returns the best font family match.

Usage

```
match_family(font = "sans", bold = TRUE, italic = TRUE, debug = NULL)
```

Arguments

font	family or face to match.
bold	Wheter to match a font featuring a bold face.
italic	Wheter to match a font featuring an italic face.
debug	deprecated

Examples

```
match_family("sans")
match_family("serif")
```

m_str_extents *Compute string extents for a vector of string.*

Description

For each x element, determines the width and height of a bounding box that's big enough to (just) enclose the provided text. Unit is pixel.

Usage

```
m_str_extents(
  x,
  fontname = "sans",
  fontsize = 10,
  bold = FALSE,
  italic = FALSE,
  fontfile = NULL
)
```

Arguments

x	Character vector of strings to measure
fontname	Font name. A vector of character to match with x.
fontsize	Font size. A vector of numeric to match with x.
bold, italic	Is text bold/italic?. A vector of logical to match with x.
fontfile	Font file. A vector of character to match with x.

Examples

```
# The first run can be slow when font caches are missing
# as font files are then being scanned to build those font caches.
m_str_extents(letters, fontsize = 1:26)
m_str_extents(letters[1:3],
  bold = c(TRUE, FALSE, TRUE),
  italic = c(FALSE, TRUE, TRUE),
  fontname = c("sans", "sans", "sans") )
```

---

raster_str *Draw/preview a raster into a string*

Description

raster_view is a helper function for testing. It uses htmltools to render a png as an image with base64 encoded data image.

Usage

```
raster_str(x, width = 480, height = 480, interpolate = FALSE)

raster_view(code)
```

Arguments

x	A raster object
width, height	Width and height in pixels.
interpolate	A logical value indicating whether to linearly interpolate the image.
code	base64 code of a raster

Examples

```
r <- as.raster(matrix(hcl(0, 80, seq(50, 80, 10)),
 nrow = 4, ncol = 5))
code <- raster_str(r, width = 50, height = 50)
if (interactive() && require("htmltools")) {
  raster_view(code = code)
}
```

raster_write *Draw/preview a raster to a png file*

Description

Draw/preview a raster to a png file

Usage

```
raster_write(x, path, width = 480, height = 480, interpolate = FALSE)
```

Arguments

x	A raster object
path	name of the file to create
width, height	Width and height in pixels.
interpolate	A logical value indicating whether to linearly interpolate the image.

Examples

```
r <- as.raster(matrix(hcl(0, 80, seq(50, 80, 10)),
 nrow = 4, ncol = 5))
filepng <- tempfile(fileext = ".png")
raster_write(x = r, path = filepng, width = 50, height = 50)
```

register_gfont *Register a 'Google Fonts'*

Description

Register a font from 'Google Fonts' so that it can be used with devices using the 'systemfonts' package, i.e. the 'flextable' package and graphic outputs generated with the 'ragg', 'svglite' and 'ggiraph' packages.

Usage

```
register_gfont(family = "Open Sans", subset = c("latin", "latin-ext"))
```

Arguments

`family` family name of a 'Google Fonts', for example, "Open Sans", "Roboto", "Fira Code" or "Fira Sans Condensed". Complete list is available with the following command:

```
gfonts::get_all_fonts()$family |>
  unlist() |>
  unique() |>
  sort()
```

`subset` font subset, a character vector, it defaults to only "latin" and "latin-ext" and can contains values such as "greek", "emoji", "chinese-traditional",

Run the following code to get a complete list:

```
gfonts::get_all_fonts()$subsets |> unlist() |> unique() |> sort()
```

Details

It allows users to use fonts from 'Google Fonts' in an HTML page generated by 'shiny' or 'R Markdown'. At the first request, the font files will be downloaded and stored in a cache on the user's machine, thus avoiding many useless downloads or allowing to work with these fonts afterwards without an Internet connection, in a docker image for example. See `fonts_cache_dir()`.

The server delivering the font files should not be too busy. That's why a one second pause is added after each download to respect the server's limits. This time can be set with the option `GFONTS_DOWNLOAD_SLEEPTIME` which must be a number of seconds.

Value

TRUE if the operation went ok.

See Also

Other functions for font management: `addGFontHtmlDependency()`, `fonts_cache_dir()`, `gfontHtmlDependency()`, `install_gfont_script()`, `installed_gfonts()`, `liberationsansHtmlDependency()`, `register_liberationsans()`

Examples

```
## Not run:
if (check_gfonts()) {
  dummy_setup()
  register_gfont(family = "Roboto")
}

## End(Not run)
```

register_liberationsans *Register font 'Liberation Sans'*

Description

Register font 'Liberation Sans' so that it can be used with devices using the 'systemfonts' package, i.e. the 'flextable' package and graphic outputs generated with the 'ragg', 'svglite' and 'ggiraph' packages.

Usage

```
register_liberationsans()
```

Value

TRUE if the operation went ok.

See Also

Other functions for font management: `addGFontHtmlDependency()`, `fonts_cache_dir()`, `gfontHtmlDependency()`, `install_gfont_script()`, `installed_gfonts()`, `liberationsansHtmlDependency()`, `register_gfont()`

set_dummy_conf *Set and unset a minimalistic Fontconfig configuration*

Description

Set and unset a minimalistic Fontconfig configuration

Usage

```
set_dummy_conf()

unset_dummy_conf()
```

Note

Fontconfig is not used anymore and these functions will be deprecated in the next release.

str_extents *Compute string extents.*

Description

Determines the width and height of a bounding box that's big enough to (just) enclose the provided text.

Usage

```
str_extents(
  x,
  fontname = "sans",
  fontsize = 12,
  bold = FALSE,
  italic = FALSE,
  fontfile = ""
)
```

Arguments

x	Character vector of strings to measure
fontname	Font name
fontsize	Font size
bold, italic	Is text bold/italic?
fontfile	Font file

Examples

```
str_extents(letters)
str_extents("Hello World!", bold = TRUE, italic = FALSE,
  fontname = "sans", fontsize = 12)
```

str_metrics *Get font metrics for a string.*

Description

Get font metrics for a string.

Usage

```
str_metrics(
  x,
  fontname = "sans",
  fontsize = 12,
  bold = FALSE,
  italic = FALSE,
  fontfile = ""
)
```

Arguments

x	Character vector of strings to measure
fontname	Font name
fontsize	Font size
bold, italic	Is text bold/italic?
fontfile	Font file

Value

A named numeric vector

Examples

```
str_metrics("Hello World!")
```

---

sys_fonts *List fonts for 'systemfonts'.*

Description

List system and registryfonts details into a data.frame containing columns foundry, family, file, slant and weight.

Usage

```
sys_fonts()
```

Examples

```
sys_fonts()
```

version_freetype *Version numbers of C libraries*

Description

version_cairo() and version_freetype() return the runtime version. These helpers return version objects as with packageVersion().

Usage

```
version_freetype()

version_cairo()
```

Index