

# Package ‘gdttools’

October 14, 2019

**Version** 0.2.1

**License** GPL-3 | file LICENSE

**Title** Utilities for Graphical Rendering

**Description** Useful tools for writing vector graphics devices.

**LazyData** TRUE

**Imports** Rcpp (>= 0.12.12), systemfonts (>= 0.1.1)

**Suggests** htmltools, testthat, fontquiver (>= 0.2.0), curl

**Encoding** UTF-8

**LinkingTo** Rcpp

**SystemRequirements** cairo, freetype2

**BugReports** <https://github.com/davidgohel/gdttools/issues>

**RoxygenNote** 6.1.1

**NeedsCompilation** yes

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

**Maintainer** David Gohel <[david.gohel@ardata.fr](mailto:david.gohel@ardata.fr)>

**Repository** CRAN

**Date/Publication** 2019-10-14 15:30:02 UTC

## R topics documented:

fontconfig_reinit	2
font_family_exists	2
glyphs_match	3
match_family	4

m_str_extents . . . . .	4
raster_str . . . . .	5
raster_write . . . . .	6
set_dummy_conf . . . . .	6
str_extents . . . . .	7
str_metrics . . . . .	7
sys_fonts . . . . .	8
version_freetype . . . . .	8

## Index 9

---

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

---

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")
```

---

glyphs_match	<i>Validate glyph entries</i>
--------------	-------------------------------

---

**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)
```

---

match_family	<i>Find best family match with systemfonts</i>
--------------	--

---

### 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	<i>Compute string extents for a vector of string.</i>
---------------	---

---

### 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	<i>Draw/preview a raster to a png file</i>
--------------	--

---

**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)
```

---

set_dummy_conf	<i>Set and unset a minimalistic Fontconfig configuration</i>
----------------	--

---

**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	Is text bold/italic?
italic	Is text bold/italic?
fontfile	Font file

**Value**

A named numeric vector

**Examples**

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

---

sys_fonts	<i>List system fonts.</i>
-----------	---------------------------

---

**Description**

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

**Usage**

```
sys_fonts()
```

**Examples**

```
sys_fonts()
```

---

version_freetype	<i>Version numbers of C libraries</i>
------------------	---------------------------------------

---

**Description**

version\_cairo() and version\_freetype() return the runtime version. These helpers return version objects as with [packageVersion\(\)](#).

**Usage**

```
version_freetype()
```

```
version_cairo()
```



# Index

font\_family\_exists, 2  
fontconfig\_reinit, 2  
  
glyphs\_match, 3  
  
m\_str\_extents, 4  
match\_family, 4  
  
packageVersion, 8  
  
raster\_str, 5  
raster\_view(raster\_str), 5  
raster\_write, 6  
  
set\_dummy\_conf, 6  
str\_extents, 7  
str\_metrics, 7  
sys\_fonts, 8  
  
unset\_dummy\_conf(set\_dummy\_conf), 6  
  
version\_cairo(version\_freetype), 8  
version\_freetype, 8