Package: gdtools (via r-universe)

October 15, 2026

Title Font Metrics and Font Management Utilities for R Graphics

Version 0.5.2

Description Compute text metrics (width, ascent, descent) using 'Cairo' and 'FreeType', independently of the active graphic device. Font lookup is delegated to 'systemfonts'. Additional utilities let users register 'Google Fonts' or bundled 'Liberation' fonts, check font availability, and assemble 'htmlDependency' objects so that fonts are correctly embedded in 'Shiny' applications, 'R Markdown' documents, and 'htmlwidgets' outputs such as 'ggiraph'.

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 (>= 1.3.1), tools

Suggests curl, gfonts, testthat

LinkingTo Rcpp

Config/roxygen2/version 8.1.1

Encoding UTF-8

Roxygen list(markdown = TRUE)

SystemRequirements cairo, freetype2, fontconfig

Config/pak/sysreqs libcairo2-dev libfontconfig1-dev libfreetype6-dev

Repository https://davidgohel.r-universe.dev

Date/Publication 2026-10-15 16:51:29 UTC

RemoteUrl https://github.com/davidgohel/gdtools

RemoteRef HEAD

RemoteSha 73eee055a87e827f0bdd2d4b5ab6a4ee5989acea

Contents

addGFontHtmlDependency . . . 2
font_family_exists . . . 4
font_google . . . 4
font_liberation . . . 5
font_set . . . 6
font_set_auto . . . 7
font_set_liberation . . . 7
fonts_cache_dir . . . 8
gfontHtmlDependency . . . 9
install_gfont_script . . . 10
installed_gfonts . . . 12
liberationmonoHtmlDependency . . . 12
liberationsansHtmlDependency . . . 13
liberationserifHtmlDependency . . . 13
register_gfont . . . 14
register_liberationmono . . . 15
register_liberationsans . . . 16
register_liberationserif . . . 17
strings_sizes . . . 18
sys_fonts . . . 19
sys_fonts_reset . . . 19
version_freetype . . . 20

Index **21**

`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()`, `liberationmonoHtmlDependency()`, `liberationsansHtmlDependency()`, `liberationserifHtmlDependency()`, `register_gfont()`, `register_liberationmono()`, `register_liberationsans()`, `register_liberationserif()`, `sys_fonts_reset()`

Examples

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

## End(Not run)
```

font_family_exists *Check if font family exists.*

Description

Check if a font family exists in available fonts.

Usage

```
font_family_exists(font_family = "sans", system_only = FALSE)
```

Arguments

font_family	font family name (case sensitive)
system_only	If TRUE, only look in system-installed fonts (as returned by systemfonts::system_fonts()). This is useful to verify that a font will be found by fontconfig, which is the font resolution mechanism used by Cairo devices at rendering time. When FALSE (the default), both system and registered fonts are considered.

Value

A logical value

Examples

```
font_family_exists("sans")
font_family_exists("Arial")
font_family_exists("Courier")
font_family_exists("Arial", system_only = TRUE)
```

font_google *Google Font specification*

Description

Create a font specification for a Google Font, to be used with font_set().

Usage

```
font_google(family, subset = c("latin", "latin-ext"))
```

Arguments

family	family name of a 'Google Fonts', for example, "Open Sans", "Roboto", "Fira Code" or "Fira Sans Condensed".
subset	font subset, a character vector, it defaults to only "latin" and "latin-ext".

Value

An object of class font_spec.

See Also

Other font set functions: font_liberation(), font_set(), font_set_auto(), font_set_liberation()

Examples

```
font_google("Roboto")
```

---

font_liberation *Liberation Font specification*

Description

Create a font specification for a Liberation font, to be used with font_set().

Usage

```
font_liberation(variant = c("sans", "serif", "mono"))
```

Arguments

variant	one of "sans", "serif", or "mono".

Value

An object of class font_spec.

See Also

Other font set functions: font_google(), font_set(), font_set_auto(), font_set_liberation()

Examples

```
font_liberation("sans")
font_liberation("mono")
```

font_set *Create a font set*

Description

Bundle font registration, HTML dependencies, and font family names into a single object. Each argument accepts a font_google() specification, a font_liberation() specification, or a plain character string naming a system font already available.

The returned object provides fields ready to use with ggplot2::theme() ($sans, $serif, $mono), girafe(fonts = ...) or dsvg(fonts = ...) ($dsvg_fonts), and girafe(dependencies = ...) ($dependencies).

Usage

```
font_set(sans = NULL, serif = NULL, mono = NULL, symbol = NULL)
```

Arguments

sans	font for sans-serif text.
serif	font for serif text.
mono	font for monospace text.
symbol	font for symbol text.

Value

An object of class font_set with elements:

sans character, the sans font family name (or NULL)

serif character, the serif font family name (or NULL)

mono character, the mono font family name (or NULL)

symbol character, the symbol font family name (or NULL)

dependencies list of htmlDependency objects

dsvg_fonts named list of family names for dsvg()

sources named list of source labels

See Also

font_set_liberation(), font_set_auto()

Other font set functions: font_google(), font_liberation(), font_set_auto(), font_set_liberation()

Examples

```
fonts <- font_set(sans = font_liberation("sans"))
fonts$sans
fonts$dsvg_fonts
fonts$dependencies
```

font_set_auto *Automatic font set*

Description

Build a font_set() by detecting the best available system fonts for each role (sans, serif, mono, symbol).

For each role a list of well-known fonts is tried in order. When none is found on the system, the corresponding Liberation font is used as a guaranteed offline fallback.

Role	Candidates (in order)	Fallback
sans	Arial, Helvetica, DejaVu Sans	Liberation Sans
serif	Times New Roman, Times, DejaVu Serif	Liberation Serif
mono	Courier New, Courier, DejaVu Sans Mono	Liberation Mono
symbol	Symbol, Apple Symbols	Liberation Sans

Usage

```
font_set_auto()
```

Value

An object of class font_set.

See Also

font_set(), font_set_liberation()

Other font set functions: font_google(), font_liberation(), font_set(), font_set_liberation()

Examples

```
fonts <- font_set_auto()
fonts
```

font_set_liberation *Liberation font set*

Description

Shortcut to create a font_set() with all four font roles using Liberation fonts (Sans, Serif, Mono) and Liberation Sans as a fallback for symbols.

Usage

```
font_set_liberation()
```

Value

An object of class font_set.

See Also

font_set(), font_set_auto()

Other font set functions: font_google(), font_liberation(), font_set(), font_set_auto()

Examples

```
fonts <- font_set_liberation()
fonts
```

---

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(), liberationmonoHtmlDependency(), liberationsansHtmlDependenc liberationserifHtmlDependency(), register_gfont(), register_liberationmono(), register_liberationsans(register_liberationserif(), sys_fonts_reset()

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())
```

---

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()`, `liberationmonoHtmlDependency()`, `liberationsansHtmlDependency()`, `liberationserifHtmlDependency()`, `register_gfont()`, `register_liberationmono()`, `register_liberationsans()`, `register_liberationserif()`, `sys_fonts_reset()`

Examples

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

## 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()`, `liberationmonoHtmlDependency()`, `liberationsansHtmlDependency()`, `liberationserifHtmlDependency()`, `register_gfont()`, `register_liberationmono()`, `register_liberationsans()`, `register_liberationserif()`, `sys_fonts_reset()`

Examples

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

## End(Not run)
```

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(), liberationmonoHtmlDependency(), liberationsansHtmlDependency(), liberationserifHtmlDependency(), register_gfont(), register_liberationmono(), register_liberationsans(), register_liberationserif(), sys_fonts_reset()

Examples

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

## End(Not run)
```

liberationmonoHtmlDependency *'Liberation Mono' Font HTML dependency*

Description

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

Usage

```
liberationmonoHtmlDependency()
```

See Also

font_set_liberation(), font_set(), gfontHtmlDependency()

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), installed_gfonts(), liberationsansHtmlDependency(), liberationserifHtmlDepender register_gfont(), register_liberationmono(), register_liberationsans(), register_liberationserif(), sys_fonts_reset()

---

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

font_set_liberation(), font_set(), gfontHtmlDependency()

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), installed_gfonts(), liberationmonoHtmlDependency(), liberationserifHtmlDepender register_gfont(), register_liberationmono(), register_liberationsans(), register_liberationserif(), sys_fonts_reset()

---

liberationserifHtmlDependency

'Liberation Serif' Font HTML dependency

Description

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

Usage

```
liberationserifHtmlDependency()
```

See Also

font_set_liberation(), font_set(), gfontHtmlDependency()

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), installed_gfonts(), liberationmonoHtmlDependency(), liberationsansHtmlDependenc register_gfont(), register_liberationmono(), register_liberationsans(), register_liberationserif(), sys_fonts_reset()

---

register_gfont *Register a 'Google Font'*

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(), liberationmonoHtmlDependency(), liberationsansHtmlDependenc liberationserifHtmlDependency(), register_liberationmono(), register_liberationsans(), register_liberationserif(), sys_fonts_reset()

Examples

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

## End(Not run)
```

---

register_liberationmono

Register font 'Liberation Mono'

Description

Register font 'Liberation Mono' 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_liberationmono(name = "Liberation Mono")
```

Arguments

name	the name to use for the font family when registering with 'systemfonts'. Using a custom name (e.g. "sans", "serif", "mono") allows devices like 'ragg' to resolve generic family names to this font.

Value

TRUE if the operation went ok.

See Also

font_set_liberation(), font_set()

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), installed_gfonts(), liberationmonoHtmlDependency(), liberationsansHtmlDependenc liberationserifHtmlDependency(), register_gfont(), register_liberationsans(), register_liberationserif sys_fonts_reset()

Examples

```
register_liberationmono()
register_liberationmono(name = "mono")
```

---

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(name = "Liberation Sans")
```

Arguments

name	the name to use for the font family when registering with 'systemfonts'. Using a custom name (e.g. "sans", "serif", "mono") allows devices like 'ragg' to resolve generic family names to this font.

Value

TRUE if the operation went ok.

See Also

font_set_liberation(), font_set()

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), installed_gfonts(), liberationmonoHtmlDependency(), liberationsansHtmlDependenc liberationserifHtmlDependency(), register_gfont(), register_liberationmono(), register_liberationserif sys_fonts_reset()

Examples

```
register_liberationsans()
register_liberationsans(name = "sans")
```

register_liberationserif *Register font 'Liberation Serif'*

Description

Register font 'Liberation Serif' 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_liberationserif(name = "Liberation Serif")
```

Arguments

name	the name to use for the font family when registering with 'systemfonts'. Using a custom name (e.g. "sans", "serif", "mono") allows devices like 'ragg' to resolve generic family names to this font.

Value

TRUE if the operation went ok.

See Also

font_set_liberation(), font_set()

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), installed_gfonts(), liberationmonoHtmlDependency(), liberationsansHtmlDependenc liberationserifHtmlDependency(), register_gfont(), register_liberationmono(), register_liberationsans() sys_fonts_reset()

Examples

```
register_liberationserif()
register_liberationserif(name = "serif")
```

strings_sizes *Compute strings sizes*

Description

Determines widths, ascent and descent in inches. Font lookup is performed by 'systemfonts' (so any font registered via `systemfonts::register_font()`, `register_gfont()`, or `font_set()` is found), then Cairo computes the actual metrics. The results are accurate for devices whose rendering finds the same font – this is guaranteed for 'systemfonts'-based devices (ragg, svglite, ggiraph) and true for Cairo devices (`cairo_pdf()`, ...) when the font is also installed at the system level. For devices with their own font engine (`pdf()`, `png()`, ...) the metrics may not match the rendering.

Usage

```
strings_sizes(
  x,
  fontname = "sans",
  fontsize = 10,
  bold = FALSE,
  italic = FALSE
)
```

Arguments

`x`	A character vector of strings to measure. All arguments are vectorized and recycled to match the length of `x`.
`fontname`	A character vector specifying the font family name (e.g., "sans", "serif", "mono"). Default is "sans". This argument is vectorized.
`fontsize`	A numeric vector specifying the font size in points. Default is 10. This argument is vectorized.
`bold`	A logical vector indicating whether the text should be bold. Default is FALSE. This argument is vectorized.
`italic`	A logical vector indicating whether the text should be italic. Default is FALSE. This argument is vectorized.

See Also

Other functions for font metrics: `m_str_extents()`, `str_metrics()`

Examples

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

sys_fonts — *List fonts for 'systemfonts'.*

Description

List system and registry fonts into a data.frame containing columns such as path, family, style, weight and italic.

Listing the system fonts is slow when many fonts are installed, so the list is read once per R session; fonts registered with 'systemfonts' (e.g. with `register_liberationsans()` or `register_gfont()`) are always included. Call `sys_fonts_reset()` after installing a font on the system during the session.

Usage

```
sys_fonts()
```

Value

A data.frame of font information.

Examples

```
sys_fonts()
```

sys_fonts_reset — *Refresh the list of system fonts*

Description

The list of system fonts used by `sys_fonts()` and `font_family_exists()` is read once per R session, as reading it is slow when many fonts are installed. Call this function after installing a font on the system during the session so that it is found.

It also resets the font cache of 'systemfonts' (`systemfonts::reset_font_cache()`).

Usage

```
sys_fonts_reset()
```

Value

`NULL`, invisibly.

See Also

Other functions for font management: addGFontHtmlDependency(), fonts_cache_dir(), gfontHtmlDependency(), install_gfont_script(), installed_gfonts(), liberationmonoHtmlDependency(), liberationsansHtmlDependenc liberationserifHtmlDependency(), register_gfont(), register_liberationmono(), register_liberationsans() register_liberationserif()

Examples

```
sys_fonts_reset()
```

---

version_freetype *Version numbers of C libraries*

Description

Return the runtime version of the Cairo and FreeType libraries linked to the package.

Usage

```
version_freetype()

version_cairo()
```

Value

An object of class "numeric_version".

Examples

```
version_cairo()
version_freetype()
```

Index

∗ **font set functions**
- font_google, 4
- font_liberation, 5
- font_set, 6
- font_set_auto, 7
- font_set_liberation, 7

∗ **functions for font management**
- addGFontHtmlDependency, 2
- fonts_cache_dir, 8
- gfontHtmlDependency, 9
- install_gfont_script, 10
- installed_gfonts, 12
- liberationmonoHtmlDependency, 12
- liberationsansHtmlDependency, 13
- liberationserifHtmlDependency, 13
- register_gfont, 14
- register_liberationmono, 15
- register_liberationsans, 16
- register_liberationserif, 17
- sys_fonts_reset, 19

∗ **functions for font metrics**
- strings_sizes, 18

addGFontHtmlDependency, 2
addGFontHtmlDependency(), *8*, *10–17*, *20*

font_family_exists, 4
font_family_exists(), *19*
font_google, 4
font_google(), *5–8*
font_liberation, 5
font_liberation(), *5–8*
font_set, 6
font_set(), *4*, *5*, *7*, *8*, *13*, *14*, *16–18*
font_set_auto, 7
font_set_auto(), *5*, *6*, *8*
font_set_liberation, 7
font_set_liberation(), *5–7*, *13*, *14*, *16*, *17*
fonts_cache_dir, 8
fonts_cache_dir(), *3*, *10–17*, *20*

gfontHtmlDependency, 9
gfontHtmlDependency(), *2*, *3*, *8*, *11–17*, *20*
ggplot2::theme(), *6*

htmltools::htmlDependency(), *10*

init_fonts_cache *(*fonts_cache_dir*)*, 8
install_gfont_script, 10
install_gfont_script(), *3*, *8*, *10*, *12–17*, *20*
installed_gfonts, 12
installed_gfonts(), *3*, *8*, *10*, *11*, *13–17*, *20*

liberationmonoHtmlDependency, 12
liberationmonoHtmlDependency(), *3*, *8*, *10–17*, *20*
liberationsansHtmlDependency, 13
liberationsansHtmlDependency(), *3*, *8*, *10–17*, *20*
liberationserifHtmlDependency, 13
liberationserifHtmlDependency(), *3*, *8*, *10–13*, *15–17*, *20*

m_str_extents(), *18*

R_user_dir(), *8*
register_gfont, 14
register_gfont(), *3*, *8*, *10–14*, *16–20*
register_liberationmono, 15
register_liberationmono(), *3*, *8*, *10–17*, *20*
register_liberationsans, 16
register_liberationsans(), *3*, *8*, *10–17*, *19*, *20*
register_liberationserif, 17
register_liberationserif(), *3*, *8*, *10–16*, *20*
rm_fonts_cache *(*fonts_cache_dir*)*, 8

str_metrics(), *18*
strings_sizes, 18

sys_fonts, 19
sys_fonts(), *19*
sys_fonts_reset, 19
sys_fonts_reset(), *3*, *8*, *10–17*, *19*
systemfonts::register_font(), *18*
systemfonts::reset_font_cache(), *19*
systemfonts::system_fonts(), *4*

version_cairo *(*version_freetype*)*, 20
version_freetype, 20