Package: thematic (via r-universe)

October 30, 2026

Title Unified and Automatic 'Theming' of 'ggplot2', 'lattice', and 'base' R Graphics

Version 0.1.7.9000

Description Theme 'ggplot2', 'lattice', and 'base' graphics based on a few choices, including foreground color, background color, accent color, and font family. Fonts that aren't available on the system, but are available via download on 'Google Fonts', can be automatically downloaded, cached, and registered for use with the 'showtext' and 'ragg' packages.

License MIT + file LICENSE

URL https://rstudio.github.io/thematic/, https://github.com/rstudio/thematic

BugReports https://github.com/rstudio/thematic/issues

Depends R (>= 3.0.0)

Imports farver, ggplot2 (>= 3.5.2), graphics, grDevices, grid, rappdirs, rlang, rstudioapi (>= 0.8), scales, utils

Suggests bslib, Cairo, curl, gganimate, ggiraph, htmltools, jsonlite, knitr, lattice, ragg, rmarkdown, shiny (>= 1.5.0), showtext, stats, svglite, sysfonts, systemfonts, testthat, vdiffr (>= 1.0.0), withr

Config/Needs/check shinytest2, callr, sf, ggthemes, patchwork, gridExtra, tinytex, devtools, rversions

Config/Needs/routine ggrepel, r-lib/evaluate

Config/Needs/website GGally, RColorBrewer, patchwork, apreshill/quillt, r-lib/evaluate

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Collate 'auto.R' 'base.R' 'cache.R' 'gfonts.R' 'ggplot.R' 'globals.R' 'hooks.R' 'knitr.R' 'lattice.R' 'onLoad.R' 'thematic-package.R' 'thematic-save-plot.R' 'thematic.R' 'utils.R'

Repository https://posit-dev-shinycoreci.r-universe.dev

Date/Publication 2025-09-29 22:03:23 UTC

RemoteUrl https://github.com/rstudio/thematic

RemoteRef HEAD

RemoteSha e082b8c7107c81e9f31da8338e4e8625ffc81bac

Contents

auto_config 2
auto_resolve_theme 4
font_cache_set 5
font_spec 5
okabe_ito 6
sequential_gradient 7
thematic_on 8
thematic_save_plot 10
thematic_with_theme 11

Index **14**

`auto_config` *Configure auto theming behavior*

Description

Auto theming is really only "guaranteed" to work inside of a **shiny** runtime. In any other context, auto theming is based on a set of heuristics, which won't fit every use case. As a workaround, this function allows one to configure both a preference for specific auto values (e.g., `bg`, `fg`, etc) as well as the `priority` that certain information should receive.

Usage

```
auto_config(
  bg = NULL,
  fg = NULL,
  accent = NULL,
  font = NULL,
  priority = c("shiny", "config", "bslib", "rstudio")
)

auto_config_set(config)

auto_config_get()
```

Arguments

bg	a background color.
fg	a foreground color.
accent	a color for making certain graphical markers 'stand out' (e.g., the fitted line color for ggplot2::geom_smooth()). Can be 2 colors for lattice (stroke vs fill accent).
font	a font_spec() object. If missing, font defaults are not altered.
priority	the order of priority to use when resolving auto values. Possible values include: • "shiny": use shiny::getCurrentOutputInfo() values (if any) to resolve auto values. • "config": use the values provided to this function (if any) to resolve auto values. • "bslib": use bslib::bs_get_variables() values (if any) to resolve auto values (only relevant when knitr is in progress). • "rstudio": use rstudioapi::getThemeInfo() values (if any) to resolve auto values.
config	a auto_config() object.

Details

Configuring auto theming behavior is especially useful for developers of a custom rmarkdown output document that wish to have more sensible auto theming behavior for users of the document. In particular, by having the output document call auto_config_set() "pre-knit" with the document's styling preferences (and restoring the old defaults "post-knit"), users of the output document can then simply call thematic_on() within their document to use those preferences.

Call this function with no arguments to get the current auto defaults.

Value

a config (list-like) object.

Examples

```
old_config <- auto_config_set(auto_config("black", "white"))
thematic_with_theme(
  thematic_theme(), {
    plot(1:10, 1:10)
 })
auto_config_set(old_config)
```

`auto_resolve_theme` *Resolve auto values*

Description

Resolves `'auto'` values based on the current execution environment and configuration (i.e., `auto_config_get()`).

Usage

```
auto_resolve_theme(theme)
```

Arguments

`theme`	a `thematic_theme()` object.

Value

The `theme` object with resolved `'auto'` values.

See Also

`auto_config_set()`

Examples

```
old_config <- auto_config_set(auto_config(bg = "black", fg = "white"))

# Resolving auto values in local theme objects
theme <- thematic_theme()
theme[c("bg", "fg")]
theme <- auto_resolve_theme(theme)
theme[c("bg", "fg")]

# By default, auto values are resolved when accessing
# global theme options
thematic_on()
thematic_get_option("bg", resolve = FALSE)
thematic_get_option("bg")
thematic_off()

auto_config_set(old_config)
```

font_cache_set *Control the directory used for font caching*

Description

The default directory used for font caching is system dependent; and thus, not very portable from machine to machine. Use this function to move thematic's cache to a new path. This is primarily useful for making font cache relative to a shiny app directory, so that, when the app is deployed, the cache deploys with it.

Usage

```
font_cache_set(path, cleanup = FALSE)
```

Arguments

path	a filepath for the new cachine directory.
cleanup	whether or not to remove font files from the previously used caching directory (after copying to the new location).

Value

Returns the previously used caching directory.

See Also

thematic_on(), font_spec()

Examples

```
## Not run:
  font_cache_set("my_app")
  shiny::runApp("my_app")

## End(Not run)
```

font_spec *Font specification*

Description

Specify a collection of font families. The first font family supported by the relevant device (i.e., the device that is open, or will be opened, at plotting time) is used by thematic. If a given font family is not supported by the default, but is a Google Font and install = TRUE, the font will be downloaded, cached, and registered for use the **showtext** and **ragg** packages.

Usage

```
font_spec(
  families = "",
  scale = 1,
  install = is_installed("ragg") || is_installed("showtext"),
  update = FALSE,
  quiet = TRUE
)
```

Arguments

families	a character vector of font families.
scale	numerical constant applied to font sizes.
install	whether to download and register font families available via Google Fonts (but unavailable to R). After a successful download, fonts are cached (in a directory which can be managed via font_cache_set()), and registered for use with the **showtext** and **ragg** packages. If installation fails with a valid internet connection, you may need to fetch the latest Google Font information prior to installation (i.e., set update = TRUE).
update	if TRUE, the latest Google Fonts are fetched and any out-dated font cache is updated. Fetching the latest fonts requires a Google Font API key (one is bundled with the package, but you can set your own via an environment variable, GFONT_KEY).
quiet	whether to suppress download messages.

Value

the input arguments as a list.

See Also

thematic_save_plot(), thematic_on(), font_cache_set()

---

okabe_ito *A color-blind safe qualitative colorscale (Okabe-Ito)*

Description

This is the default qualitative colorscale in thematic_on()

Usage

```
okabe_ito(n = NULL)
```

Arguments

n	number of colors.

Value

a vector of color codes.

References

https://jfly.uni-koeln.de/color/

See Also

thematic_on()

`sequential_gradient` *Control parameters of the sequential colorscale*

Description

Controls the default weighting and direction of the color gradient derived from the `fg`, `bg`, and `accent` color (defined in `thematic_on()`).

Usage

```
sequential_gradient(fg_weight = 0.9, bg_weight = 0, fg_low = TRUE, n = 30)
```

Arguments

`fg_weight`	a number (between 0 and 1) defining much of the `fg` color should be mixed into the colorscale.
`bg_weight`	a number (between 0 and 1) defining much of the `bg` color should be mixed into the colorscale.
`fg_low`	if `TRUE` (the default), the `fg` color is used for the low end of the colorscale (rather than the high end).
`n`	number of color codes.

Value

a list of options for passing to the `sequential` argument of `thematic_on()`.

Examples

```
# Gradient from fg to accent
fg <- sequential_gradient(1, 0)
thematic_on("black", "white", "salmon", sequential = fg)
ggplot2::qplot(1:10, 1:10, color = 1:10)

# Gradient from accent -> bg
bg <- sequential_gradient(0, 1)
thematic_on("black", "white", "salmon", sequential = bg)
```

```
ggplot2::qplot(1:10, 1:10, color = 1:10)

# Gradient from mix(accent, fg, 0.5) -> mix(accent, bg, 0.5)
mix <- sequential_gradient(0.5, 0.5)
thematic_on("black", "white", "salmon", sequential = mix)
ggplot2::qplot(1:10, 1:10, color = 1:10)

# Use fg (instead of bg) for high end of scale
mix_flip <- sequential_gradient(0.5, 0.5, fg_low = FALSE)
thematic_on("black", "white", "salmon", sequential = mix_flip)
ggplot2::qplot(1:10, 1:10, color = 1:10)
```

---

thematic_on *Enable (or disable) simplified theming of R graphics.*

Description

A unified interface for theming **ggplot2**, **base**, and **lattice** graphics based on a handful of styling options. In some cases (most notably in a **shiny** runtime), these options can automatically resolve to relevant CSS styles (see the "Auto theming" section below).

Usage

```
thematic_on(
  bg = "auto",
  fg = "auto",
  accent = "auto",
  font = NA,
  sequential = sequential_gradient(),
  qualitative = okabe_ito(),
  inherit = FALSE
)

thematic_off()

thematic_theme(
  bg = "auto",
  fg = "auto",
  accent = "auto",
  font = NA,
  sequential = sequential_gradient(),
  qualitative = okabe_ito(),
  inherit = FALSE
)

thematic_shiny(
  bg = "auto",
```

```
  fg = "auto",
  accent = "auto",
  font = NA,
  sequential = sequential_gradient(),
  qualitative = okabe_ito(),
  inherit = FALSE,
  session = shiny::getDefaultReactiveDomain()
)

thematic_rmd(
  bg = "auto",
  fg = "auto",
  accent = "auto",
  font = NA,
  sequential = sequential_gradient(),
  qualitative = okabe_ito(),
  inherit = FALSE
)
```

Arguments

bg	a background color.
fg	a foreground color.
accent	a color for making certain graphical markers 'stand out' (e.g., the fitted line color for `ggplot2::geom_smooth()`). Can be 2 colors for lattice (stroke vs fill accent).
font	a `font_spec()` object. If missing, font defaults are not altered.
sequential	a color palette for graphical markers that encode numeric values. Can be a vector of color codes or a `sequential_gradient()` object.
qualitative	a color palette for graphical markers that encode qualitative values (won't be used in ggplot2 when the number of data levels exceeds the max allowed colors). Defaults to `okabe_ito()`.
inherit	should non-specified values inherit from the previous theme?
session	see `shiny::onStop()`.

Value

`thematic_theme()` returns a theme object as a list (which can be activated with `thematic_with_theme()` or `thematic_set_theme()`).

`thematic_on()`, `thematic_off()`, and `thematic_shiny()` all return the previous global theme.

Auto theming

The `bg`, `fg`, `accent`, and `font` arguments all support a value of `'auto'`, which are all resolved, at plot time, based on the execution environment. In a **shiny** runtime, resolution of auto values should always work as expect; but in other contexts, auto values may lead to wrong or surprising results. In that case, auto resolution logic can be customized (see `auto_config_set()` for more details).

Global vs. local theming

thematic_on() enables thematic in a global fashion (that is, it impacts all future plots, up until thematic_off() is called). To use thematic in local fashion, first create a theme with thematic_theme(), then provide it to thematic_with_theme() (or similar). To use thematic in a global fashion up until a **shiny** app exits, use thematic_shiny() (which cleans up after itself once the next shiny app that exits using shiny::onStop()). To use thematic in a global fashion up until a **rmarkdown** document finishes rendering, use thematic_rmd().

Color values

Colors (e.g., bg, fg, accent) may be any value understood by col2rgb() or htmltools::parseCssColors() (i.e., may be any valid R or CSS color string).

See Also

sequential_gradient(), thematic_with_theme(), thematic_save_plot()

Examples

```
# simple dark mode
thematic_on("black", "white")
plot(1:10)
plot(1:10, col = 1:10)
lattice::show.settings()

# use any hex color string
thematic_on("#444444", "#e4e4e4")
plot(1:10)
plot(1:10, col = 1:10)
lattice::show.settings()

# disables thematic (also restores global state)
thematic_off()
plot(1:10)
lattice::show.settings()

thematic_on("darkblue", "skyblue", "orange")
image(volcano)
image(volcano, col = thematic_get_option("sequential"))
lattice::show.settings()
thematic_off()
```

---

thematic_save_plot *Capture a thematic plot as a saved file*

Description

Uses a `device` to capture the result of an expression (`expr`) that produces a plot. If `default_device()` is used, custom fonts (specified through `font_spec()`) are guaranteed to work, as long as one of either the showtext or ragg package(s) are installed.

Usage

```
thematic_save_plot(
  expr,
  device = default_device(),
  filename = tempfile(fileext = ".png"),
  ...
)

default_device(type = c("png", "svg", "pdf", "tiff", "jpeg"))
```

Arguments

`expr`	an expression that produces a plot.
`device`	a graphics device to use for capturing the plot.
`filename`	a filename for the produced plot. The file extension should match the relevant `device`.
`...`	arguments passed along to the graphics `device`.
`type`	the type of output format

Value

`thematic_save_plot()` returns the `filename` of the produced plot and `default_device()` returns a graphics device function.

Examples

```
library(thematic)
font <- font_spec("Rock Salt", scale = 1.25)
thematic_on("black", "white", font = font)
file <- thematic_save_plot(plot(1:10), res = 144)
if (interactive()) browseURL(file)
```

---

thematic_with_theme *Tools for getting and restoring global state*

Description

These functions are helpful for getting and/or temporarily activating a `thematic_theme()`.

Usage

```
thematic_with_theme(theme, expr)

thematic_local_theme(theme, .local_envir = parent.frame())

thematic_set_theme(theme)

thematic_get_theme(resolve = TRUE)

thematic_get_option(name = "", default = NULL, resolve = TRUE)

thematic_get_mixture(amounts = 0.5, default = NULL)
```

Arguments

theme	a thematic_theme() object (or a return value of thematic_on/thematic_get_theme()) or NULL (in which case thematic_off() is called).
expr	R code that produces a plot.
.local_envir	The environment to use for scoping.
resolve	whether or not 'auto' values should be resolved before returning
name	a theme element name (e.g., fg, bg, etc.)
default	a default value to return in the event no thematic theme is active.
amounts	value(s) between 0 and 1 specifying how much to mix bg (0) and fg (1).

Value

the result of expr.

Functions

- thematic_with_theme(): similar to thematic_on(), but for an single plot.
- thematic_local_theme(): similar to thematic_with_theme(), but de-couples the theme from the plot expression.
- thematic_set_theme(): set a given theme object as the current theme.
- thematic_get_theme(): obtain the current theme.
- thematic_get_option(): obtain a particular theme option (and provide a default if no theme is active).
- thematic_get_mixture(): obtain a mixture of the current theme's bg and fg.

Examples

```
# Use thematic_with_theme() for a one-time use of thematic
thematic_with_theme(
  thematic_theme("darkblue", "skyblue", accent = "red"),
  plot(1:10, col = thematic_get_option("accent"), pch = 19)
)
```

```
# Use thematic_set_theme() if doing something more complicated
# like programming on top thematic (without causing side effects)
my_plot <- function(expr, las = 3, ...) {
  old_theme <- thematic_on("black", "white")
  on.exit(thematic_set_theme(old_theme), add = TRUE)
  opts <- par(las = las)
  on.exit(par(opts), add = TRUE)
  # Imagine some more customization with ...
  force(expr)
}
my_plot(plot(1:10))

thematic_off()
thematic_get_option("bg", "white")
thematic_on(bg = "red")
thematic_get_option("bg", "white")
thematic_off()

thematic_with_theme(
  thematic_theme("darkblue", "skyblue"),
  scales::show_col(thematic_get_mixture(seq(0, 1, by = 0.1)))
)
```

Index

auto_config, 2
auto_config_get *(*auto_config*)*, 2
auto_config_get(), *4*
auto_config_set *(*auto_config*)*, 2
auto_config_set(), *4*, *9*
auto_resolve_theme, 4

col2rgb(), *10*

default_device *(*thematic_save_plot*)*, 10

font_cache_set, 5
font_cache_set(), *6*
font_spec, 5
font_spec(), *3*, *5*, *9*, *11*

ggplot2::geom_smooth(), *3*, *9*

okabe_ito, 6
okabe_ito(), *9*

rstudioapi::getThemeInfo(), *3*

sequential_gradient, 7
sequential_gradient(), *9*, *10*
shiny::getCurrentOutputInfo(), *3*
shiny::onStop(), *9*, *10*

thematic_get_mixture
        *(*thematic_with_theme*)*, 11
thematic_get_option
        *(*thematic_with_theme*)*, 11
thematic_get_theme
        *(*thematic_with_theme*)*, 11
thematic_get_theme(), *12*
thematic_local_theme
        *(*thematic_with_theme*)*, 11
thematic_off *(*thematic_on*)*, 8
thematic_off(), *9*
thematic_on, 8, *12*
thematic_on(), *5–7*, *9*, *12*
thematic_rmd *(*thematic_on*)*, 8
thematic_save_plot, 10
thematic_save_plot(), *6*, *10*
thematic_set_theme
        *(*thematic_with_theme*)*, 11
thematic_set_theme(), *9*
thematic_shiny *(*thematic_on*)*, 8
thematic_shiny(), *9*
thematic_theme *(*thematic_on*)*, 8
thematic_theme(), *9–11*
thematic_with_theme, 11
thematic_with_theme(), *9*, *10*, *12*