Package: benviplot (via r-universe)

June 19, 2026

Title Color Palettes and 'ggplot2' Extensions for Data Visualization

Version 1.2.0

Description Provides 'ggplot2' scale functions and a custom theme derived from the 'Benvi' visual identity of QuintoAndar Group (discontinued in 2024). Offers qualitative, sequential, and diverging color palettes with both discrete and continuous scale variants.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports cli, dplyr (>= 1.1.0), ggplot2 (>= 4.0.0), graphics, rlang

Suggests curl, ggfittext, knitr, pkgdown, ragg, rmarkdown, systemfonts, testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

VignetteBuilder knitr

URL https://viniciusoike.github.io/benviplot/, https://github.com/viniciusoike/benviplot

BugReports https://github.com/viniciusoike/benviplot/issues

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

Date/Publication 2026-06-02 17:19:14 UTC

RemoteUrl https://github.com/viniciusoike/benviplot

RemoteRef HEAD

RemoteSha a618e0c388724498b7792fa9be8bc78441d08c00

Contents

`benvi_palette` *A color palette for Benvi.*

Description

Construction of the package is inspired by the ghibli package.

Usage

```
benvi_palette(
  pal_name = "qual_2",
  n,
  direction = 1,
  type = c("discrete", "continuous")
)
```

Arguments

`pal_name`	Name of the palette. Defaults to "qual_2".
`n`	Number of colors desired. Sets have 4 colors, Qual have 8 colors.
`direction`	Either `1` or `-1`. If `-1` the palette will be reversed.
`type`	Either "continuous" or "discrete". Continuous automatically interpolates between the colors.

Value

A vector of characters with color attribute

Examples

```
# Use default palette
benvi_palette()

# Specify palette name
benvi_palette("greens")
benvi_palette("greens", n = 20, type = "continuous")
benvi_palette("greens", n = 2, type = "discrete")
```

`font_status` *Report benviplot font status*

Description

Reports whether Poppins is available and whether the `ragg` graphics device is installed.

Usage

```
font_status()
```

Value

Invisibly returns a list with `poppins_available` and `ragg_available`.

Examples

```
font_status()
```

`format_num_br` *Format numbers using Brazilian conventions*

Description

Formats numbers using Brazilian locale conventions: period (.) as thousands separator and comma (,) as decimal separator. This is a convenient wrapper to both the `format` and `round` base functions to convert numeric vectors into character labels for tables or plots.

Usage

```
format_num_br(x, digits = 1, percent = FALSE)
```

Arguments

`x`	A numeric vector
`digits`	Number of decimal places to include. Can be negative to round to tens, hundreds, etc.
`percent`	Logical indicating if % symbol should be appended

Details

Brazilian number formatting uses:

- Thousands separator: . (period)
- Decimal separator: , (comma)

For example, 1234567.89 becomes "1.234.567,9" (with digits = 1).

Value

A character vector with formatted numbers

Examples

```
# Basic formatting
x <- 1235134.123
format_num_br(x)

# Different decimal places
format_num_br(x, digits = 3)
format_num_br(x, digits = 0)

# With percentage
format_num_br(12.5, digits = 1, percent = TRUE)

# Negative digits round to tens, hundreds, etc.
format_num_br(1234567, digits = -3)

# Works with vectors
format_num_br(c(100, 1000, 10000))
```

---

ggplot2-scales-continuous

Continuous scales to use for ggplot2

Description

These functions provide the option to use Benvi colors inside continuous palettes with the `ggplot2` package.

Usage

```
scale_colour_benvi_c(pal_name = "benvi_blue", direction = 1, ...)

scale_color_benvi_c(pal_name = "benvi_blue", direction = 1, ...)

scale_fill_benvi_c(pal_name = "benvi_blue", direction = 1, ...)
```

Arguments

pal_name	Name of the palette. Defaults to "qual_2".
direction	Either 1 or -1. If -1 the palette will be reversed.
...	Arguments to pass on to ggplot2::scale_colour_gradientn() or ggplot2::scale_fill_gradientn

Value

A ScaleContinuous object that can be added to a ggplot object

Examples

```
library(ggplot2)

ggplot(mtcars, aes(x = wt, y = mpg, color = hp)) +
  geom_point() +
  scale_color_benvi_c("benvi_blue")
```

---

ggsave_benvi *Save a ggplot with benviplot optimizations*

Description

A wrapper around ggplot2::ggsave() with smart defaults optimized for benviplot graphics. Automatically uses the ragg graphics device for PNG output when available, ensuring high-quality rendering with proper font support and no DPI issues.

If ragg is not installed, falls back to the default graphics device.

Usage

```
ggsave_benvi(
  filename,
  plot = ggplot2::last_plot(),
  device = NULL,
  width = 7,
  height = 5,
  units = "in",
  dpi = 300,
  ...
)
```

Arguments

`filename`	File name to create on disk. The file extension determines the graphics device (e.g., ".png", ".pdf", ".svg").
`plot`	Plot to save. Defaults to the last plot displayed.
`device`	Device to use. Defaults to "ragg" for PNG files if ragg package is installed, otherwise uses ggplot2's default device selection.
`width, height`	Plot size in units. Defaults to 7 x 5 inches.
`units`	Units for width and height ("in", "cm", "mm", "px"). Default is "in".
`dpi`	DPI to use for raster graphics. Default is 300 for high quality.
`...`	Additional arguments passed to `ggplot2::ggsave()`.

Value

Invisibly returns the filename.

See Also

`ggplot2::ggsave()`

Examples

```
p <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) + ggplot2::geom_point()
ggsave_benvi(tempfile(fileext = ".png"), plot = p)
```

---

iqa *QuintoAndar Index (IQA) - Rental Price Index*

Description

Historical rental price index data from QuintoAndar. This is the legacy IQA index, which has been superseded by the IQAIW (see `iqaiw`).

Usage

```
iqa
```

Format

A data frame with 96 observations and 6 variables:

date Date of the observation (first day of month)

name_muni Name of the municipality (city)

index Rental price index, normalized to 100 at first observation

chg Monthly percent variation of the index (decimal form)

acum12m 12-month accumulated variation of the index (decimal form)

price_m2 Estimated rental price per square meter (R$/m²)

Source

QuintoAndar

Examples

```
# To visualize the dataset
head(iqa)
str(iqa)

# Plot index over time for all cities
library(ggplot2)
ggplot(iqa, aes(x = date, y = index, color = name_muni)) +
  geom_line() +
  scale_color_benvi_d(pal_name = "qual_9", name = "City") +
  labs(
    title = "IQAIW: Rental Price Index",
    x = "Date",
    y = "Index (base = 100)"
  ) +
  theme_benvi(base_family = "sans")
```

---

iqaiw *QuintoAndar ImovelWeb Rental Index (IQAIW)*

Description

The IQAIW (Índice QuintoAndar ImovelWeb) is a rental index for major Brazilian cities. The index is based on both new rental contracts (managed by QuintoAndar) and online listings from QuintoAndar's listings (including ImovelWeb).

Usage

```
iqaiw
```

Format

A data frame with 1,660 observations across 6 cities and multiple time periods:

date Date of the observation (first day of month)

name_muni Name of the municipality. One of: Belo Horizonte, Brasília, Curitiba, Porto Alegre, Rio de Janeiro, São Paulo

rooms Number of rooms in the property, or "Total" for city-level aggregate

index Rental price index, normalized to 100 at first observation per city

chg Monthly percent variation of the index (decimal form)

acum12m 12-month accumulated variation of the index (decimal form)

price_m2 Estimated rental price per square meter (R$/m²)

Details

The IQAIW was developed in 2023 and replaced the former IQA index. Given the change in methodology and data sources, the IQAIW is not directly comparable to the IQA index.

Methodology

Formally, the index is a hedonic double imputed index, controlling for quality changes using a flexible GAM specification with location variables. In this sense, the IQAIW is more theoretically sound than median stratified indices like FipeZap or the former IQA. The mixture of listings and contracts, however, lacks theoretical support and seems to be mainly driven by branding purposes.

The ImovelWeb brand was purchased by QuintoAndar in 2021-22 and the IQAIW symbolizes the merging of both brands. In other words, the original IQA could've been improved simply by adopting a hedonic methodology, without the need to mix data sources.

Source

https://publicfiles.data.quintoandar.com.br/indice_quintoandar_imovelweb/index_quintoandar_imovelweb_serie.csv

Examples

```
# To visualize the dataset
head(iqaiw)
str(iqaiw)

# Plot index over time for all cities
library(ggplot2)

iqaiw_rooms <- subset(iqaiw, rooms != "Total" & !is.na(acum12m))

ggplot(iqaiw_rooms, aes(x = date, y = acum12m, color = rooms)) +
  geom_line(lwd = 0.5) +
  geom_hline(yintercept = 0) +
  scale_color_benvi_d(pal_name = "qual_6", name = "Rooms") +
  facet_wrap(vars(name_muni), ncol = 3, scales = "free") +
  labs(
    title = "IQAIW: Rental Price Index",
    x = "Date",
    y = "Index (base = 100)"
  ) +
  theme_benvi(base_family = "sans")
```

---

plot_area *Plot an area chart*

Description

Plot an area chart

Usage

```
plot_area(
  data,
  x,
  y,
  fill = NULL,
  zero = TRUE,
  order = TRUE,
  pal_name = "qual_benvi",
  scale_name = "",
  scale_label = ggplot2::waiver(),
  text = FALSE,
  text_color = "gray20",
  text_family = getOption("theme_benvi.font_family", "sans"),
  text_size = 3,
  position = "stack",
  position_text = "identity"
)
```

Arguments

data	A data.frame type object
x	<data-masked> Variable to be mapped in the x-axis.
y	<data-masked> Variable to be mapped in the y-axis.
fill	Fill color for the area. Either a color string (e.g., "blue", "#021841") for a single static color, or a bare column name (without quotes) to map a grouping variable to fill color.
zero	Logical indicating whether a horizontal line crossing the y = 0 axis should be plotted.
order	Logical indicating if the stacked areas should be ordered. Default behavior (TRUE) stacks the largest groups on top.
pal_name	String indicating the name of which palette to use.
scale_name	String indicating fill legend title.
scale_label	String indicating fill legend labels.
text	Logical indicating if text labels should be plotted on column bars
text_color	Color of the text label. Default is "gray20".
text_family	Font of the text label. Defaults to getOption("theme_benvi.font_family", "sans").
text_size	Size of the text label. Default is 3.
position	Argument passed to geom_area.
position_text	Argument passed on to position in geom_text.

Value

A ggplot2 plot

Examples

```
# Simple area chart
sao_paulo <- subset(iqa, name_muni == "S\u00e3o Paulo")
plot_area(data = sao_paulo, x = date, y = index)

# Stacked area chart with fill mapping
total <- subset(iqaiw, rooms == "Total")
plot_area(data = total, x = date, y = index, fill = name_muni)
```

`plot_column` *Plot a column chart*

Description

Plot a column chart

Usage

```
plot_column(
  data,
  x,
  y,
  fill = NULL,
  zero = TRUE,
  text = FALSE,
  text_inside = FALSE,
  text_place = NULL,
  text_padding = NULL,
  pal_name = "qual_benvi",
  scale_name = "",
  scale_label = ggplot2::waiver(),
  digits = 0,
  percent = FALSE,
  text_color = "gray20",
  text_family = getOption("theme_benvi.font_family", "sans"),
  text_size = 3,
  position_col = "stack",
  position_text = position_col,
  ...
)
```

Arguments

`data`	A data.frame type object
`x`	<data-masked> Variable to be mapped in the x-axis.
`y`	<data-masked> Variable to be mapped in the y-axis.

fill	Fill color for the columns. Either a color string (e.g., "blue", "#021841") for a single static color, or a bare column name (without quotes) to map a grouping variable to fill color.
zero	Logical indicating whether a horizontal line crossing the y = 0 axis should be plotted.
text	Logical indicating if text labels should be plotted on column bars
text_inside	Logical indicating if text labels should be placed inside bars (using ggfittext). When TRUE, text is auto-sized to fit inside bars. When FALSE (default), text appears above/beside bars at fixed size.
text_place	Placement of inside text. One of "top", "bottom", "left", "right", "centre"/"center". Only used when text_inside = TRUE. Defaults to "centre".
text_padding	Padding around inside text as grid::unit(). Only used when text_inside = TRUE. Defaults to 1mm.
pal_name	String indicating the name of which palette to use.
scale_name	String indicating fill legend title.
scale_label	String indicating fill legend labels.
digits	Number of digits to show in text labels.
percent	Logical indicating if a % should be appended to text labels
text_color	Color of the text label. Default is "gray20".
text_family	Font of the text label. Defaults to getOption("theme_benvi.font_family", "sans").
text_size	Size of the text label. Default is 3.
position_col	Argument passed on to position in geom_col.
position_text	Argument passed on to position in geom_text.
...	Further arguments for geom_text

Value

A ggplot2 plot

Examples

```
# Column chart by city at the latest date
latest <- subset(iqa, date == max(iqa$date))
plot_column(data = latest, x = name_muni, y = index)

# With text labels above bars
plot_column(data = latest, x = name_muni, y = index, text = TRUE)



# With text labels inside bars
latest <- subset(iqa, date == max(iqa$date))
plot_column(data = latest, x = name_muni, y = index, text = TRUE, text_inside = TRUE)
```

`plot_histogram` *Plot a histogram chart*

Description

Plot a histogram chart

Usage

```
plot_histogram(
  data,
  x,
  color = "#FFFFFF",
  fill = NULL,
  pal_name = "qual_benvi",
  scale_name = "",
  zero = TRUE,
  bins = NULL,
  method = "fd",
  density = FALSE,
  facet = FALSE,
  ...
)
```

Arguments

`data`	A data.frame type object
`x`	<data-masked> Indicates the numeric variable to be mapped
`color`	Color of the column border. Defaults to `"#FFFFFF"` (white).
`fill`	Fill color for the columns. Either a color string (e.g., `"blue"`, `"#021841"`) for a single static color, or a bare column name (without quotes) to map a grouping variable to fill color.
`pal_name`	String indicating the name of which palette to use when `fill` is a variable mapping.
`scale_name`	String indicating fill legend title.
`zero`	Logical indicating if a horizontal (y = 0) line should be drawn on the plot.
`bins`	Number of bins. When specified, overrides `method`.
`method`	Character specifying the binning algorithm. Must be one of: `"fd"` (default), `"FD"`, `"Scott"`, `"Sturges"`, `"Rice"`, or `"sqrt"`. See Details for algorithm descriptions. Ignored when `bins` is specified.
`density`	Logical indicating if density should be plotted on y-axis.
`facet`	<data-masked> Optional variable to facet the graphics.
`...`	Additional parameters to `facet_wrap()`

Details

Binning Methods:

The `method` parameter controls which algorithm is used to compute the optimal bin width. Available methods:

`"fd"` **or** `"FD"` **Freedman-Diaconis rule** (default). Robust to outliers, uses IQR. Formula: $2 * IQR/n^{1/3}$. Best for most distributions.

`"Scott"` **Scott's rule**. Uses standard deviation. Formula: $3 * sd/n^{1/3}$. Works well for normal-like distributions.

`"Sturges"` **Sturges' formula**. Simple logarithmic rule. Formula: $k = \lceil log_2(n) \rceil$ bins. Good for roughly normal data.

`"Rice"` **Rice rule**. Cube root based. Formula: $k = \lceil 2n^{1/3} \rceil$ bins. General purpose rule.

`"sqrt"` **Square root rule**. Formula: $k = \lceil \sqrt{n} \rceil$ bins. Simple, tends to oversmooth.

When in doubt, use the default `"fd"` (Freedman-Diaconis), which is robust and works well across different distributions.

Value

A ggplot2 object

Examples

```
set.seed(5)
tbl <- data.frame(x = rnorm(n = 1000))

# Default parameters use Freedman-Diaconis
plot_histogram(data = tbl, x = x)
# Use bins to manually choose number of bins
plot_histogram(data = tbl, x = x, bins = 50)
# Example of alternative methods: square root and Rice
plot_histogram(data = tbl, x = x, method = "sqrt")
plot_histogram(data = tbl, x = x, method = "Rice")

# Facet by rooms category
spo <- subset(iqaiw, name_muni == "S\u00e3o Paulo" & rooms != "Total")
plot_histogram(data = spo, x = index, facet = rooms)
```

`plot_line` *Title*

Description

Title

Usage

```
plot_line(
  data,
  x,
  y,
  color = NULL,
  zero = TRUE,
  point = FALSE,
  pal_name,
  scale_name = "",
  scale_label = ggplot2::waiver(),
  ...
)
```

Arguments

data	A data.frame type object.
x	<data-masked> Variable to be mapped on the x-axis.
y	<data-masked> Variable to be mapped on the y-axis.
color	Color of the line. Either a color string (e.g., "blue", "#021841") for a single static color, or a bare column name (without quotes) to map a grouping variable to color.
zero	Logical indicating if a horizontal line (y = 0) should be drawn on the plot.
point	Logical indicating if points should be drawn on top of line.
pal_name	String indicating which color palette to use.
scale_name	String indicating color legend title.
scale_label	String indicating color legend labels.
...	Other arguments to ggplot2 function.

Value

A ggplot2 plot

Examples

```
# Single series
sao_paulo <- subset(iqa, name_muni == "S\u00e3o Paulo")
plot_line(data = sao_paulo, x = date, y = index)

# Multiple series with color mapping
total <- subset(iqaiw, rooms == "Total")
plot_line(data = total, x = date, y = index, color = name_muni)
```

`plot_scatter` *Plot a scatter chart*

Description

Plot a scatter chart

Usage

```
plot_scatter(
  data,
  x,
  y,
  color = NULL,
  fit = FALSE,
  fit_variable = FALSE,
  fit_method = "auto",
  fit_formula = NULL,
  fit_color = NULL,
  fit_ci = FALSE,
  zero = "none",
  pal_name = "qual_benvi",
  scale_name = "",
  scale_label = ggplot2::waiver(),
  ...
)
```

Arguments

`data`	A data.frame type object
`x`	<data-masked> Variable to be mapped in the x-axis.
`y`	<data-masked> Variable to be mapped in the y-axis.
`color`	Color of the points. Either a color string (e.g., `"blue"`, `"#021841"`) for a single static color, or a bare column name (without quotes) to map a grouping variable to color. Continuous numeric variables automatically use a continuous color scale.
`fit`	Logical indicating if a regression line should be plotted on top of the chart.
`fit_variable`	Logical indicating if regression should be grouped. Defaults to `FALSE`.
`fit_method`	Type of model to generate regression line. See `geom_smooth` for more control and details. Defaults to `"auto"`.
`fit_formula`	A formula for fit_method. See `geom_smooth`.
`fit_color`	Color of the fitted regression line. Only applied when `fit_variable = FALSE`. When `NULL` (default), uses automatic color selection. When `fit_variable = TRUE`, the fit line colors are inherited from the grouping variable and this parameter is ignored.

fit_ci	Logical indicating if confidence interval should be plotted. Defaults to FALSE for less cluttered visualization.
zero	Draws axis lines. Must be one of "x", "y", "both", or "none" (default).
pal_name	String indicating the name of which palette to use.
scale_name	String indicating fill legend title.
scale_label	String indicating fill legend labels.
...	Further arguments to geom_point

Value

A ggplot2 plot.

Examples

```
plot_scatter(data = mtcars, x = wt, y = mpg)

# With regression line
plot_scatter(data = mtcars, x = wt, y = mpg, fit = TRUE)
```

---

sales_report *QuintoAndar Sales Report*

Description

Sales price data at a region level for major Brazilian cities. Contains contract prices per square meter, allowing comparison across cities and zones.

Usage

```
sales_report
```

Format

sales_report:

A data frame with 272 observations across multiple cities and zones:

date Date of the observation (first day of month)

name_muni Name of the municipality (city). Includes: Belo Horizonte, Rio de Janeiro, and São Paulo

name_zone Name of the zone within the city

price_m2 Median contract price per square meter (R$/m²)

Details

This dataset provides zone-level granularity, showing sales prices for specific regions within cities.

Source

QuintoAndar (Sales Report 2020-Q1/2023-Q3). https://publicfiles.data.quintoandar.com.br/sale_report/RelatorioCV_4T_2022.pdf

Examples

```
# Compare contract prices across zones
library(ggplot2)

bhe_sales <- subset(sales_report, name_muni == "Belo Horizonte" & date == max(date))

bhe_sales$name_zone <- factor(
  bhe_sales$name_zone,
  levels = bhe_sales$name_zone[order(bhe_sales$price_m2)]
)

ggplot(bhe_sales, aes(x = price_m2, y = name_zone)) +
geom_col(fill = benvi_palette("benvi_blue")[3]) +
theme_benvi(base_family = "sans")
```

---

scale_colour_benvi_d *Discrete scales to use for ggplot2*

Description

Functions to use ggplot2 scales with Benvi colors.

Usage

```
scale_colour_benvi_d(pal_name = "qual_benvi", direction = 1, ...)

scale_color_benvi_d(pal_name = "qual_benvi", direction = 1, ...)

scale_fill_benvi_d(pal_name = "qual_benvi", direction = 1, ...)
```

Arguments

pal_name	Name of the palette.
direction	Either 1 or -1. If -1 the palette will be reversed.
...	additional arguments to pass to discrete_scale

Examples

```
library(ggplot2)
# Discrete color scale with rental index data
iqaiw_total <- subset(iqaiw, rooms == "Total")
ggplot(iqaiw_total, aes(x = date, y = index, colour = name_muni)) +
  geom_line() +
```

```
scale_color_benvi_d("qual_benvi")
```

---

show_palettes *Display all Benvi palettes*

Description

Visually displays all available Benvi palettes in a grid layout, modeled on `RColorBrewer::display.brewer.all()`. Optionally filter by palette type.

Usage

```
show_palettes(type = "all", n = NULL)
```

Arguments

type	Character string specifying the palette type to display. One of: `"all"` (default), `"theme"`, `"sequential"`, `"qualitative"`, `"diverging"`, `"city"`, or `"brand"`.
n	Number of colors to display from each palette. If `NULL` (default), shows all colors in each palette.

Value

Invisibly returns `NULL`. Called for its side effect of creating a plot.

Examples

```
# Display all palettes
show_palettes()

# Display only theme palettes
show_palettes("theme")

# Display sequential palettes
show_palettes("sequential")
```

theme_benvi *A theme for Benvi styled plots*

Description

A ggplot2 base theme for Benvi styled plots.

The Poppins font is bundled with the package and registered automatically on load when `systemfonts` is installed. When both `systemfonts` and `ragg` are available, the theme uses Poppins by default. Otherwise it falls back to the system's default sans-serif font.

Registered fonts only work with systemfonts-aware devices (e.g. ragg::agg_png). Base R devices (PDF, PostScript) cannot render them. If you see font warnings when saving to PDF, pass `base_family = "sans"` or set `options(theme_benvi.font_family = "sans")`. See `font_status()` to check your setup.

Usage

```
theme_benvi(
  base_family = getOption("theme_benvi.font_family", default_font_family()),
  base_size = 10,
  background = FALSE
)
```

Arguments

base_family	Argument passed to `ggplot2::theme_minimal()`. Defaults to `"Poppins"` when the bundled font is registered and `ragg` is available, `"sans"` otherwise. Override globally with `options(theme_benvi.font_family = ...)`.
base_size	Argument passed to `ggplot2::theme_minimal()`. Defaults to 10.
background	Logical. Adds an offwhite (creme) background to the plot.

Value

A ggplot2 theme object

Examples

```
library(ggplot2)
series <- subset(iqaiw, name_muni == "S\u00e3o Paulo" & rooms == "Total")

# Base theme (using "sans" for portability)
ggplot(series, aes(date, index)) +
  geom_line(color = benvi_palette("benvi_blue")[1], lwd = 1) +
  labs(x = NULL, y = "Index (base = 100)", title = "IQAIW") +
  theme_benvi(base_family = "sans")

# Optional offwhite (creme) background
ggplot(series, aes(date, index)) +
```

```
  geom_line(color = benvi_palette("benvi_blue")[1], lwd = 1) +
  labs(x = NULL, y = "Index (base = 100)", title = "IQAIW") +
  theme_benvi(base_family = "sans", background = TRUE)

## Not run:
# Use Poppins with a ragg device (requires systemfonts and ragg)
ggplot(series, aes(date, index)) +
  geom_line(color = benvi_palette("benvi_blue")[1], lwd = 1) +
  labs(x = NULL, y = "Index (base = 100)", title = "IQAIW") +
  theme_benvi(base_family = "Poppins")

## End(Not run)
```

Index