Package: ggpop (via r-universe)

May 18, 2026

Type Package

Title Icon-Based Population Charts and Plots for 'ggplot2'

Version 1.7.1

Date 2026-03-18

Description Create engaging population and point plots charts in R. 'ggpop' allows users to represent population data and points proportionally using customizable icons, facilitating the creation of circular representative population charts as well as any point-plots.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0), knitr, rmarkdown, cowplot, ggforce, gganimate, ggrepel, ggtext, scales, reactable

Config/Needs/website sf, geofacet, ggtext, quarto, kcuilla/reactablefmtr

Config/testthat/edition 3

Depends R (>= 4.0.5)

Imports ggplot2, dplyr, ggimage, magick, rlang, tidyr, purrr, fontawesome, rsvg, cli, tibble

VignetteBuilder knitr

ByteCompile true

BugReports https://github.com/jurjoroa/ggpop/issues

URL https://jurjoroa.github.io/ggpop/

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev make libmagick++-dev gsfonts libicu-dev libpng-dev librsvg2-dev libuv1-dev libssl-dev

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

Date/Publication 2026-03-18 23:58:58 UTC

RemoteUrl https://github.com/jurjoroa/ggpop

RemoteRef HEAD

RemoteSha 4b1ed537da316b28e42119835d2384b4e1f2e589

Contents

`ggpop-package` *ggpop: Icon-Based Population Charts for R*

Description

ggpop is a ggplot2 extension for creating icon-based population charts and pictogram plots. Use `geom_pop()` and `geom_icon_point()` to visualize proportion and population data with 2,000+ Font Awesome icons.

Main functions

- `geom_pop()` – proportional icon grids
- `geom_icon_point()` – icon scatter plots
- `process_data()` – prepare count data for plotting
- `fa_icons()` – search Font Awesome icon names
- `theme_pop()`, `theme_pop_dark()`, `theme_pop_minimal()` – built-in themes

process_data()

Converts count data to one row per icon. `group_var` and `sum_var` are unquoted; `high_group_var` takes a character string for faceted charts.

```
df_plot <- process_data(
  data        = data.frame(sex = c("Female", "Male"), n = c(55, 45)),
  group_var   = sex,
  sum_var     = n,
  sample_size = 20
)
```

geom_pop()

Draws icon grids. Add an `icon` column, map `icon` and `color` in `aes()`. Do not map `x` or `y`.

```
ggplot() +
  geom_pop(data = df_plot, aes(icon = icon, color = type), size = 2) +
  scale_color_manual(values = c(Female = "#C0392B", Male = "#2980B9")) +
  theme_pop()
```

geom_icon_point()

Drop-in replacement for `geom_point()` using Font Awesome icons.

```
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_icon_point(icon = "seedling", size = 1)
```

fa_icons()

Search the bundled Font Awesome icon list by keyword.

```
fa_icons(query = "person")
```

Themes

Three built-in themes optimized for icon charts: `theme_pop()`, `theme_pop_dark()`, `theme_pop_minimal()`.

Author(s)

Maintainer: Jorge A. Roa-Contreras <jorgeroa@stanford.edu> (ORCID)

Authors:

- Ralitza Soultanova <Ralitza.soultanova@gmail.com> (ORCID)
- Fernando Alarid-Escudero <falarid@stanford.edu> (ORCID)
- Carlos Pineda-Antunez <cpinedaa@uw.edu> (ORCID)

See Also

Useful links:

- https://jurjoroa.github.io/ggpop/
- Report bugs at https://github.com/jurjoroa/ggpop/issues

Examples

```
library(ggplot2)
library(dplyr)

## ------------------------------------------------------
## geom_pop(): population icon grid
## ------------------------------------------------------
```

```
df_plot <- process_data(
  data        = data.frame(sex = c("Female", "Male"), n = c(55, 45)),
  group_var   = sex,
  sum_var     = n,
  sample_size = 20
) %>%
  mutate(icon = ifelse(type == "Female", "person-dress", "person"))

ggplot() +
  geom_pop(data = df_plot, aes(icon = icon, color = type), size = 2) +
  scale_color_manual(values = c(Female = "#C0392B", Male = "#2980B9")) +
  theme_pop() +
  labs(title = "Population by sex", color = NULL)

## ------------------------------------------------------
## geom_icon_point(): icon scatter plot
## ------------------------------------------------------
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_icon_point(icon = "seedling", size = 1) +
  scale_color_manual(values = c(
    setosa     = "#43A047",
    versicolor = "#1E88E5",
    virginica  = "#E53935"
  )) +
  labs(title = "Iris dataset", x = "Sepal Length", y = "Petal Length")
```

`fa_icons` *Search and list Font Awesome icons*

Description

Retrieves Font Awesome icon names, optionally filtered by a search query or category. Results can be returned as a plain character vector or as a tibble with category classification.

Usage

```
fa_icons(
  query = NULL,
  category = NULL,
  regex = FALSE,
  classify = TRUE,
  include_unclassified = TRUE,
  class_map = NULL,
  primary_only = TRUE,
  as_vector = FALSE
)
```

Arguments

query	Character string. Filter icons whose names contain query. Set to NULL (default) to return all icons. If regex = TRUE, query is treated as a Perl-compatible regular expression.
category	Character vector. One or more category names to filter by. Run fa_categories() to see valid options. Setting category implies classify = TRUE.
regex	Logical. When TRUE, query is interpreted as a Perl-compatible regular expression. Default FALSE (fixed-string match).
classify	Logical. When TRUE (default), each icon is classified into categories using class_map and a primary_class column is included in the returned tibble. Ignored when as_vector = TRUE and category = NULL.
include_unclassified	Logical. When FALSE, icons that do not match any category pattern are dropped. Default TRUE.
class_map	A named list mapping category names to regex patterns. Defaults to the internal .fa_default_class_map().
primary_only	Logical. When TRUE (default), the tibble contains only the primary_class column and omits all_classes.
as_vector	Logical. When TRUE, return a plain sorted character vector of icon names instead of a tibble. If category = NULL, classification is skipped entirely. Default FALSE.

Value

When as_vector = TRUE, a sorted character vector of icon names. Otherwise a tibble with columns:

icon Icon name (character).

primary_class Primary category the icon belongs to, or NA when unclassified (character).

all_classes All matching categories (list-column of character vectors). Only present when primary_only = FALSE.

Examples

```
# All icons as a classified tibble
fa_icons()

# Quick lookup -- plain sorted vector
head(fa_icons(as_vector = TRUE), 10)

# Search for icons whose name contains "heart"
fa_icons(query = "heart")

# Filter by category
fa_icons(category = "animals")

# Regex search -- all icons starting with "arrow"
```

```
fa_icons(query = "^arrow", regex = TRUE)
```

`fetch_df_coordinates` *Fetches the* `df_coordinates_final` *Dataset*

Description

Downloads and caches the `df_coordinates_final` dataset if it is not already cached locally. This function ensures that the dataset is downloaded only once and loaded into memory without cluttering the global environment. The dataset is stored in a package-specific cache directory and retrieved efficiently for subsequent uses.

Usage

```
fetch_df_coordinates()
```

Details

The dataset is downloaded from GitHub The file is cached in a directory specific to the package, which is determined using `R_user_dir`. If the dataset is already cached, it will be loaded directly from the cache instead of downloading again.

Value

A data frame containing the `df_coordinates_final` dataset.

Examples

```
df <- fetch_df_coordinates()
head(df)
```

`geom_icon_point` *Create a scatter plot with Font Awesome icons instead of points*

Description

Works exactly like geom_point(), but renders Font Awesome icons instead of dots. Pass any data with x and y variables - no special formatting required.

Usage

```
geom_icon_point(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  icon = NULL,
  size = 1,
  dpi = 50,
  legend_icons = TRUE,
  stroke_width = NULL,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.

- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`na.rm`	logical, whether remove NA values
`show.legend`	Logical. Should this layer be included in the legends? `NA` (default) includes the layer if any aesthetics are mapped. `FALSE` suppresses the layer's legend entries entirely.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.
`icon`	Default Font Awesome icon (default: NULL).
`size`	Default icon size (default: 1).
`dpi`	Icon resolution (default: 50).
`legend_icons`	Show icons in legend (default: TRUE).
`stroke_width`	Numeric. Width of the icon outline/stroke.
`...`	additional parameters

Value

A ggplot layer.

Aesthetics

geom_icon_point uses standard ggplot2 scatter plot aesthetics:

- **x** - Numeric variable for x-axis
- **y** - Numeric variable for y-axis
- **icon** - Font Awesome icon name (optional, column or mapped)
- **color/colour** - Color grouping
- **alpha** - Transparency
- **size** - Icon size

Examples

```
library(ggplot2)
data <- data.frame(
  x = rnorm(20),
  y = rnorm(20),
  category = sample(c("A", "B", "C"), 20, replace = TRUE),
  icon = sample(c("heart", "star", "circle"), 20, replace = TRUE)
)

# Map icon to a column
ggplot(data, aes(x = x, y = y, icon = icon, color = category)) +
```

```
  geom_icon_point()

# Use a fixed icon
ggplot(data, aes(x = x, y = y, color = category)) +
  geom_icon_point(icon = "star")
```

---

geom_pop *Create a circular representative population chart*

Description

Draws a circular representative population chart based on group proportions, where each point (person) represents a fixed number of individuals. Each person is rendered as a Font Awesome icon.

Usage

```
geom_pop(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  icon = "ggmale",
  group_var = NULL,
  sample_size = NULL,
  arrange = FALSE,
  seed = NULL,
  sum_var = NULL,
  facet = NULL,
  size = 1,
  dpi = 50,
  legend_icons = TRUE,
  stroke_width = NULL,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
na.rm	logical, whether remove NA values
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
icon	Default icon to use when no icon column is mapped.
group_var	(Deprecated) Use aes(group = ...) instead.
sample_size	The total number of individuals (points) to draw.
arrange	Logical; if TRUE, output data is arranged by group.
seed	Optional numeric seed used only when arrange = FALSE.
sum_var	Optional variable to sum over instead of counting.

facet	Optional faceting variable. If provided, final plot must be faceted with ggplot2 (use validate_geom_pop_faceting(p)).
size	Icon size. If mapped in aes(size = ...) the parameter is ignored.
dpi	Height (in **pixels**) of the rendered PNG when using fontawesome::fa_png(). Higher values produce sharper icons.
legend_icons	Logical; if TRUE, legend displays the selected icons.
stroke_width	Numeric. Width of the icon outline in pixels (single value).
...	additional parameters

Value

A ggplot layer that renders a circular population chart with icons.

Aesthetics

geom_pop understands the following aesthetics:

- **icon**: Font Awesome icon name (mapped column)
- **group**: grouping variable for raw data mode
- **color/colour**: icon color
- **alpha**: transparency (must be mapped)
- **size**: icon size (mapped or fixed)

See Also

geom_icon_point, process_data, geom_image

Examples

```
library(ggplot2)

df <- data.frame(
  sex  = rep(c("F", "M"), each = 10),
  icon = rep(c("female", "male"), each = 10)
)

ggplot() +
  geom_pop(
    data = df,
    aes(icon = icon, group = sex, color = sex),
    size = 3,
    dpi = 80
  )
```

process_data *Process Population Data for Visualization*

Description

The process_data function processes a dataset to calculate group proportions and generates a sampled dataset based on specified parameters. This processed data is suitable for creating visual representations, such as population charts, where each sample represents a group with associated counts and proportions.

Usage

```
process_data(
  data,
  high_group_var = NULL,
  group_var,
  sum_var = NULL,
  sample_size = 100
)
```

Arguments

data	A data frame containing the population data to be processed.
high_group_var	Character vector, optional. The variables used to group individuals hierarchically. This should be a categorical variable. If provided, the function samples individuals within each group defined by these variables.
group_var	Quosure. The variable used to group individuals in the dataset. This should be a categorical variable.
sum_var	Quosure, optional. The variable to sum over within each group. If NULL, the function counts the number of individuals per group.
sample_size	Integer. The total number of individuals to sample based on group proportions. Must be a positive integer.

Value

A tibble (data frame) with the following columns:

type The sampled group type.

group The group identifier.

n The count of individuals in the group.

prop The proportion of the group relative to the total population.

`scale_legend_icon` *Legend helper for geom_pop/geom_icon_point legends*

Description

A convenience function to set appropriate legend key sizes for icon-based legends. This is equivalent to using theme(legend.key.size = ...) but provides sensible defaults for population icon plots.

Usage

```
scale_legend_icon(
  size = 10,
  unit = "mm",
  spacing = 0.2,
  size_multiplier = 2,
  ...
)
```

Arguments

`size`	Numeric. Legend key size in specified units (default 10).
`unit`	Character. Unit for legend key sizing (default "mm").
`spacing`	Numeric. Spacing between legend items as fraction of size (default 0.2).
`size_multiplier`	Numeric. Multiplier to apply to the size for spacing calculations (default 2).
`...`	Additional theme arguments.

Value

A ggplot2 theme object that can be added to a plot.

Examples

```
library(ggplot2)
df <- data.frame(
  type = rep(c("A", "B"), each = 10),
  icon = rep(c("circle", "square"), each = 10)
)
ggplot(df, aes(icon = icon, color = type)) +
  geom_pop() +
  scale_legend_icon(size = 20)
```

theme_pop *Population Plot Theme*

Description

A minimal theme optimized for icon-based population plots. Similar to `theme_void()` but with automatic legend key sizing, appropriate margins, and sensible defaults for population visualizations.

Usage

```
theme_pop(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  legend_icon_size = NULL,
  legend_spacing = NULL,
  plot_margin = NULL,
  legend_position = "right"
)
```

Arguments

`base_size`	Base font size in points (default: 11).
`base_family`	Base font family (default: "").
`base_line_size`	Base size for line elements (default: base_size/22).
`base_rect_size`	Base size for rect elements (default: base_size/22).
`legend_icon_size`	Size of legend icons in cm. If NULL (default), automatically calculated as base_size/20 for proportional sizing.
`legend_spacing`	Spacing between legend items in cm (default: 0.3 * legend_icon_size).
`plot_margin`	Plot margins. Default: margin(5.5, 5.5, 5.5, 5.5, "pt"). Can be a single numeric (applied to all sides) or margin() object.
`legend_position`	Position of legend: "none", "left", "right", "bottom", "top" (default: "right").

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
df <- data.frame(
  type = rep(c("A", "B"), each = 10),
  icon = rep(c("circle", "square"), each = 10)
```

```
)
ggplot(data = df, aes(icon = icon, color = type)) +
  geom_pop(size = 1) +
  theme_pop()
```

`theme_pop_dark` *Dark Population Plot Theme*

Description

A dark variant of theme_pop() with white text on black background. Perfect for presentations or dark-mode visualizations.

Usage

```
theme_pop_dark(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  legend_icon_size = NULL,
  legend_spacing = NULL,
  plot_margin = NULL,
  legend_position = "right",
  bg_color = "black",
  text_color = "white"
)
```

Arguments

`base_size`	Base font size in points (default: 11).
`base_family`	Base font family (default: "").
`base_line_size`	Base size for line elements (default: base_size/22).
`base_rect_size`	Base size for rect elements (default: base_size/22).
`legend_icon_size`	Size of legend icons in cm. If NULL (default), automatically calculated as base_size/20 for proportional sizing.
`legend_spacing`	Spacing between legend items in cm (default: 0.3 * legend_icon_size).
`plot_margin`	Plot margins. Default: margin(5.5, 5.5, 5.5, 5.5, "pt"). Can be a single numeric (applied to all sides) or margin() object.
`legend_position`	Position of legend: "none", "left", "right", "bottom", "top" (default: "right").
`bg_color`	Background color (default: "black").
`text_color`	Text color (default: "white").

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
df <- data.frame(
  type = rep(c("A", "B"), each = 10),
  icon = rep(c("circle", "square"), each = 10)
)
ggplot(data = df, aes(icon = icon, color = type)) +
  geom_pop(size = 1) +
  theme_pop_dark(base_size = 40)
```

---

theme_pop_minimal *Minimal Population Plot Theme*

Description

An ultra-minimal variant with no margins or legend, perfect for icon arrays without annotations.

Usage

```
theme_pop_minimal(base_size = 11, base_family = "")
```

Arguments

base_size	Base font size in points (default: 11).
base_family	Base font family (default: "").

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
df <- data.frame(
  type = rep(c("A", "B"), each = 10),
  icon = rep(c("circle", "square"), each = 10)
)
ggplot(data = df, aes(icon = icon, color = type)) +
  geom_pop(size = 1) +
  theme_pop_minimal()
```

Index