Package 'ggstratify'

September 2, 2026

Title Fast Stratified Descriptive Figures with a Point-and-Click GUI

Version 0.0.1

Description A point-and-click 'shiny' interface for the descriptive analysis that comes before any model is chosen. Pass a data frame, pick the variable to describe, and add the layers you want to see it within: a second variable becomes the panels of a 'ggplot2' facet_wrap(), and further variables become separate figures, one file each, taken either one variable at a time or crossed. Every stratum is reported with the number of observations behind it, on the figure and on each of its panels; strata that contain none are listed rather than dropped, and rows with a missing value in a layer variable are excluded and counted. A continuous variable can be categorized into quantile groups, equal-width bins or user-supplied cut points and then used as a layer; the figure types follow those offered by the 'ggplotgui' package and add the line plot for change over time, an optional LOWESS smoother, and the Kaplan-Meier curve estimated by 'survival', with an optional number-at-risk table. Columns are described as they are typed, so convert each to the type you mean first. Figures are written as PNG or SVG, and the application prints the 'ggplot2' code behind the figure on screen, so that a description can be repeated, shared or accounted for later. Everything runs locally, with no network access and no AI involved.

License GPL-3

URL https://github.com/AkiShiroshita/ggstratify, https://akishiroshita.github.io/ggstratify/

BugReports https://github.com/AkiShiroshita/ggstratify/issues

Encoding UTF-8

Depends R (>= 4.1)

Imports bslib (>= 0.5.0), checkmate (>= 2.1.0), data.table (>= 1.14.0), ggplot2 (>= 3.4.0), grDevices, patchwork (>= 1.1.0), ragg, shiny (>= 1.7.0), stats, survival (>= 3.2.0), utils

Suggests knitr, rmarkdown, svglite (>= 2.1.0), testthat (>= 3.0.0), tibble

VignetteBuilder knitr

LazyData true

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Akihiro Shiroshita [aut, cre, cph],
Yuki Kataoka [aut]

Maintainer Akihiro Shiroshita <akihirokun8@gmail.com>

Repository CRAN

Date/Publication 2026-09-02 11:40:02 UTC

Contents

`epi_cohort` *A synthetic clinical cohort*

Description

Six hundred simulated patients, shaped like the descriptive tables that motivate this package: a few continuous measurements crossed with several categorical variables worth stratifying on.

Usage

```
epi_cohort
```

Format

A data frame with 600 rows and 11 columns:

id Patient identifier, `"P0001"` to `"P0600"`.

age Age in years.

sex Factor: `"Male"`, `"Female"`.

site Factor: `"Site A"`, `"Site B"`, `"Site C"`, `"Site D"`. `"Site D"` has no observations.

treatment Factor: `"Control"`, `"Low dose"`, `"High dose"`.

severity Factor: `"Mild"`, `"Moderate"`, `"Severe"`.

bmi Body mass index, kg/m^2.

crp C-reactive protein, mg/L.

los_days Length of stay in days.

fu_days Days of follow-up, to death or to censoring. Censoring is by dropout or by the end of the study at 365 days.

death `1` if the patient died during follow-up, `0` if censored.

Details

Two features are deliberate. `site` declares a fourth level, `"Site D"`, that recruited nobody, so stratifying by `site` produces a stratum with `n = 0`; the app lists it and draws no figure. `severity` has a small `"Severe"` group – 20 patients against 381 and 199 – which is what the minimum-n control is there to be tried on: it is drawn at the default minimum of 10, and disappears from the figures, with a reason, as soon as the minimum is raised past 20.

The data are simulated. They describe no real patients and support no clinical conclusion.

`fu_days` and `death` make the data usable for a Kaplan-Meier curve, and `age` for the categorization controls.

Source

Simulated by `data-raw/epi_cohort.R`.

Examples

```
str(epi_cohort)

# The empty stratum that the app reports with n = 0.
table(epi_cohort$site)
```

`ggstratify` *Describe a data set, one layer at a time*

Description

Launches a point-and-click Shiny interface for descriptive analysis: pick the variable you want to describe, then add the layers you want to see it within. The figure, the number of observations behind it and the R code that reproduces it all appear together.

Usage

```
ggstratify(dataset, launch.browser = TRUE, ...)
```

Arguments

`dataset`	The data to describe, and the one thing the function needs: a data frame – including a `tibble` or a `data.table` – or a matrix, already read into your session and already of the types you mean. A file path is not accepted; see *Before you start*. Whatever it is given becomes a plain `data.table`, so a grouped `tibble` is described as its rows, not as its groups. The object is copied, never modified in place, and its name is what the generated code refers to it by – so pass a named object, rather than an expression: `ggstratify(head(cohort))` is described perfectly well, but the code it prints says `d <- mydata`, because `head(cohort)` is not a name.
`launch.browser`	Passed to `shiny::runApp()`. `TRUE` opens the system browser.
`...`	Further arguments passed to `shiny::runApp()`.

Value

Invisibly `NULL`, after the app is closed. Called for its side effect of running a Shiny application.

The layers

A description has a variable being described and, around it, the variables you want to see it within. `ggstratify` calls those the layers, and asks you to place each one:

One layer The variable on its own. A plain `ggplot2` figure.

Two layers The second variable becomes the panels of a `facet_wrap()`, so the whole comparison is one figure.

Three or more Two layers is what a single figure holds. Beyond that, you choose: one variable stays as the `facet_wrap()` panels, and the rest become separate figures – one file each. With columns A, B, C and D you might describe D, panel it by C, and get one figure per level of A and of B.

The variables that make separate figures can be treated separately, which is the default – ticking `sex` and `treatment` gives the figures `sex_M`, `sex_F`, `treatment_A`, ... – or crossed, giving one figure per observed combination. Separately is usually the right question for descriptive work: crossing two five-level variables gives twenty-five mostly empty figures.

What it tells you

Every level is reported with the number of observations it contains, and that number is written into the figure title and onto each panel strip inside it: a strip reads `site: Site A (N = 303)` rather than `Site A`, so that a level which does not explain itself – the bare range a categorized variable produces – still says which variable it is a level of. A level with no observations at all (an unused factor level) is listed with `N = 0` and produces no figure. Rows with a missing value in any layer variable are excluded – a row that does not say which panel it belongs to cannot be drawn in one – and the number excluded is reported on the Strata tab and by the generated code.

Three things descriptive work keeps needing

Under **Categorize**, a continuous variable becomes a categorical one – quantile groups, equal-width bins or your own cut points – and can then be used as a layer like any other categorical variable.

Under **Type of graph**, *Kaplan-Meier curve* draws survival curves from a time variable and an event indicator, fitted with `survival::survfit()` within each group, panel and figure, optionally with a confidence band, censoring marks and a number-at-risk table under the curve. *Line* draws change over time: put time on the X axis and the measurement on the Y axis, and name the subject identifier under **One line per** to get one line per subject.

A *Line* or *Scatter* figure can carry a LOWESS smoother – `stats::loess()` through `ggplot2::geom_smooth()` – with its span under your control and, when a grouping variable is set, one fit per group.

Before you start

Convert each column to the type you mean it to have – `numeric` for measurements, `factor` for groups, with the levels in the order you want them read – before passing the data. `ggstratify`

describes what it is given; it does not guess what you meant. A grouping variable stored as `1, 2, 3` will be described as a number.

This is why the data must be an object you already have in your session. There is no file to choose from inside the application, and no file path to hand it: a data set that arrives by being read from disk arrives with types that a reader guessed, and it is then described on those guessed types without anyone having looked at them. Read the file yourself, run `str()` or `summary()` over the result, fix the types that are wrong, and pass that object:

```
cohort <- read.csv("cohort.csv")
cohort$treatment <- factor(cohort$treatment,
                           levels = c("Control", "Low dose", "High dose"))
str(cohort)
ggstratify(cohort)
```

The figure types, themes and palettes otherwise match those offered by the **ggplotgui** package. All data handling uses **data.table**. Figures are written as PNG (through **ragg**) or as SVG, and the "R-code" tab shows the self-contained **ggplot2** code for the figure on screen – just the figure, since writing the files is what the export button is for. The preview, the export button and that code are all produced by the same code generator, so the code you are shown is the code that made the figure.

Examples

```
if (interactive()) {
  ggstratify(iris)
  ggstratify(epi_cohort)
}
```

Index