Package: chart (via r-universe)

July 3, 2024

Type Package

Version 1.5.2

Title Unified Interface (with Formula) for R Plots

Description Chart generalizes plot generation in R, being with base R plot function, lattice or ggplot2. A formula interface is available for both ggplot2 and lattice. The function 'chart()' automatically uses labels and units if they are defined in the data.

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 4.2.0), ggplot2 (>= 3.0), lattice (>= 0.21.0)

Imports cowplot (>= 1.1.1), data.io (>= 1.5.0), ggplotify (>= 0.1.0), ggpubr (>= 0.6.0), graphics (>= 4.2.0), grDevices (>= 4.2.0), latticeExtra (>= 0.6.30), pryr (>= 0.1.6), rlang (>= 1.1.1), scales (>= 1.2.1), stats (>= 4.2.0), svMisc (>= 1.4.0), utils (>= 4.2.0), viridis (>= 0.6.2)

Suggests MASS (>= 7.3.58.3), knitr (>= 1.42), rmarkdown (>= 2.21), spelling (>= 2.2.1), testthat (>= 3.0.0)

Remotes SciViews/data.io, SciViews/svMisc

License MIT + file LICENSE

URL https://github.com/SciViews/chart, https://www.sciviews.org/chart/

BugReports https://github.com/SciViews/chart/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

VignetteBuilder knitr

Encoding UTF-8

Language en-US

ByteCompile yes

Config/testthat/edition 3

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

RemoteUrl https://github.com/SciViews/chart

RemoteRef HEAD

RemoteSha 5ee9ecbecf979a4eabbdfaae0f1b1d58b9c19a29

Contents

`chart-package` *Unified Interface (with Formula) for R Plots*

Description

Unification of base plots, lattice and ggplot2, providing a single interface for all three plot engines.

Important functions

- `chart()` constructs a **Chart** object.
- `combine_charts()` combines multiple **Chart** objects into a single plot.
- `f_aes()` creates a formula for aesthetics mapping (use it instead of `ggplot2::aes()`.

`chart` *Create charts*

Description

`chart()` provides a unified interface for base plots, lattice and ggplot2.

Usage

```
chart(data, ..., type = NULL, env = parent.frame())

## Default S3 method:
chart(
  data,
  specif = NULL,
  formula = NULL,
  mapping = NULL,
```

```
  ...,
  type = NULL,
  auto.labs = TRUE,
  env = parent.frame()
)

## S3 method for class '`function`'
chart(data, ..., type = NULL, auto.labs = TRUE, env = parent.frame())

## S3 method for class 'subsettable_type'
x$name
```

Arguments

data	The dataset (a data.frame or tibble, usually).
...	Further arguments.
type	The type of plot to produce.
env	The environment where to evaluated the formula.
specif	Specification, being either aes(), or a formula.
formula	A formula.
mapping	An aes() object, as for ggplot().
auto.labs	Are labels (and units) automatically used for axes?
x	A subsettable_type function.
name	The value to use for the type= argument.

Details

....

See Also

f_aes(), ggplot()

Examples

```
urchin <- data.io::read("urchin_bio", package = "data.io", lang = "en")

# base R graphics
hist(urchin$height)
# ... translates to:
chart(function() hist(urchin$height))
# ... or if the expression is provided directly, specify it is a base chart
chart(hist(urchin$height), type = "base")
# ... or more concisely:
chart$base(hist(urchin$height))

# A lattice plot
histogram(~ height, data = urchin)
```

```
chart$histogram(urchin, ~ height)

# ggplot2 histogram
ggplot(urchin, aes(height)) + geom_histogram()
#... or with chart (notice similarities with lattice version)
chart(urchin,  ~ height) + geom_histogram()
#chart$geom_histogram(urchin,  ~ height)
```

`chart_theme` *Select chart theme*

Description

Lattice and ggplot2 provide themes for their plots, while with base R plots, one specifies appearance through `par()`. `chart_theme()` tries to get the plot appearance as uniform as possible between the three plot engines. So, setting themes this way change the appearance of all three kinds of plots.

Usage

```
chart_theme(theme)

theme_sciviews(font_size = 12, font_family = "", line_size = 0.5)

theme_sciviews_lattice(...)

theme_sciviews_graphics(...)

theme_svgray()

theme_svgray_lattice(...)

theme_svgray_graphics()

theme_svmap()

theme_svmap_lattice()

theme_svmap_graphics()
```

Arguments

`theme`	The theme to apply (character string).
`font_size`	The default size font for this theme.
`font_family`	The default font family in this theme.
`line_size`	The default line size in this theme.
`...`	Arguments passed to `ggplot2like()`, the most used being `n=` for the number of colors to generate in the palette.

See Also

chart(), theme(), trellis.par.set()

Examples

```
# TODO..
```

`combine_charts` *Combine charts*

Description

Assemble multiple charts on the same page. Wrapper around ggarrange() with different defaults.

Usage

```
combine_charts(chartlist, ncol = NULL, nrow = NULL, labels = "AUTO", ...)
```

Arguments

chartlist	List of charts to combine.
ncol	(optional) number of columns in the plot grid.
nrow	(optional) number of rows in the plot grid.
labels	(optional) labels to use for each individual plot. "AUTO"
...	further arguments passed to ggarrange(). (default value) auto-generates uppercase labels, and "auto" does the same for lowercase labels.

Value

An object of class ggarrange containing a list of ggplots.

See Also

chart(), ggarrange()

Examples

```
# TODO...
```

f_aes *Create aes()thetics from formula for ggplot2*

Description

This function allows to use a formula interface directly with **ggplot2**, or **chart**.

Usage

```
f_aes(formula, ..., with.facets = FALSE)
```

Arguments

formula	A formula.
...	Further aesthetics to set (like size, colour, etc.)
with.facets	Do we create special (non-ggplot2) aesthetics for facets (no by default)?

Value

An aesthetic object of class uneval, as those obtained with aes().

Examples

```
# TODO...
```

Sgg *Make a ggplot function pipeable*

Description

The set Sgg should be used like this: Sgg$geom_point(). It transforms on the fly an original {ggplot2} function supposed to be used with the + operator (like p + geom_point()) into a pipeable version (like p %>% Sgg$geom_point()).

Usage

```
Sgg(ggplot, ...)

## S3 method for class 'subsettable_Sgg'
x$name

## S3 method for class 'subsettable_Sgg'
.DollarNames(x, pattern = "")
```

Arguments

`ggplot`	An object of class "ggplot" (or "theme").
`...`	Further arguments passed to the the ggplot function (see Details).
`x`	The `Sgg()`function.
`name`	The name of the ggplot function to make pipeable.
`pattern`	A regular expression to list matching names.

Details

The function returned by `Sgg$fun` is a modified version of the function `fun` where a first argument `ggplot =` is added, and the instruction `ggplot + ...` is added in its body. A message is also added in the body to explicitly warn about these changes. All the other arguments of `fun` remain valid and should keep their original meaning.

The changes are done on the fly, and the original function `fun` is **not** altered anywhere else (and in particular, no alteration is done in a package or a namespace). When using this construct, make sure that: (1) you understand what is done, (2) you are aware that you use an altered version of the original function, (3) a bug or strange behavior may occur due to the patch and the original author of the function is not responsible in this case (the problem must be reported to the author of `Sgg` and the maintainer of the present package instead), and (4) the patched function exhibits an additional argument and behaves differently to what is described in the help page of the original, non-patched, function!

Value

The `Sgg()` function just returns an error message. When subsetted with the name of a {ggplot2} function (e.g., `Sgg$geom_point()`), it returns a modified version of that function in such a way that it can be used with a pipe operator.

Examples

```
library(ggplot2)
data(iris)
ggplot(aes(x = Petal.Length, y = Sepal.Length, col = Species) , data = iris) |>
  Sgg$geom_point() |>
  Sgg$labs(x = "Sepal length (mm)", y = "Petal length (mm)")
# Also try completion with Sgg$<tab>
```

Index