Package 'gginnards'

July 13, 2018

Type Package

Title Explore the Innards of 'ggplot2' Objects

Version 0.0.1

Date 2018-07-05

Maintainer Pedro J. Aphalo <pedro.aphalo@helsinki.fi>

Description Extensions to 'ggplot2' providing low-level debug tools: statistics and geometries reporting data passed to compute_group() and compute_panel() functions and to geometries. Layer manipulation: functions for deletion, insertion, extraction and reordering of layers of ``ggplot'' objects. Data manipulation: function for deletion of unused variables from the data object embedded in ``ggplot'' objects.

License GPL (>= 2)

LazyData TRUE

LazyLoad TRUE

ByteCompile TRUE

Depends R (>= 3.3.0), ggplot2 (>= 3.0.0)

Imports methods, grid, rlang (>= 0.2.0), stringr (>= 1.3.1), magrittr (>= 1.5), tibble (>= 1.4.2)

Suggests dplyr (>= 0.7.5), knitr (>= 1.20), rmarkdown (>= 1.9)

URL https://www.r4photobiology.info, https://bitbucket.org/aphalo/gginnards

BugReports https://bitbucket.org/aphalo/gginnards/issues

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Pedro J. Aphalo [aut, cre] (<https://orcid.org/0000-0003-3385-972X>)

Repository CRAN

Date/Publication 2018-07-13 16:40:07 UTC

R topics documented:

`gginnards-package` *gginnards: Explore the Innards of 'ggplot2' Objects*

Description

Extensions to 'ggplot2' providing low-level debug tools: statistics and geometries reporting data passed to compute_group() and compute_panel() functions and to geometries. Layer manipulation: functions for deletion, insertion, extraction and reordering of layers of "ggplot" objects. Data manipulation: function for deletion of unused variables from the data object embedded in "ggplot" objects.

Details

The new facilities for cleanly defining new stats and geoms added to package 'ggplot2' in version 2.0.0 gave origin to this package. I needed tools to help me learn how layers work and to debug the extenssions to 'ggplot2' that I was developing. I share them through this package in the hope that they will help other users of 'ggplot2' understand how this this vry popular graphics package works internally.

Extensions provided:

- "Debug" stats and a "debug" geom that print to the console a summary of their `data` input.
- Functions for inspecting and manipulating the list of layers of a ggplot object.
- Function that drops unused variables from the data embedded in ggplot objects.

Author(s)

Maintainer: Pedro J. Aphalo <pedro.aphalo@helsinki.fi> (0000-0003-3385-972X)

References

Package 'ggplot2' web site at http://ggplot2.org/
Package 'ggplot2' documentation at http://docs.ggplot2.org/
Package 'ggplot2' source code at https://github.com/hadley/ggplot2

See Also

Useful links:

- https://www.r4photobiology.info
- https://bitbucket.org/aphalo/gginnards
- Report bugs at https://bitbucket.org/aphalo/gginnards/issues

---

`delete_layers` *Layer manipulation*

Description

Delete, move or append one or more layers in a ggplot.

Usage

```
delete_layers(x, match_type = NULL, idx = NULL)

append_layers(x, object, position = "top")

move_layers(x, match_type = NULL, position = "top", idx = NULL)

shift_layers(x, match_type = NULL, idx = NULL, shift = 1L)

which_layers(x, match_type = NULL, idx = NULL)

extract_layers(x, match_type = NULL, idx = NULL)

top_layer(x)

bottom_layer(x)

num_layers(x)
```

Arguments

`x`	an object of class gg to be operated upon.
`match_type`	The name of the ggproto object class for the geom(s), position(s) or stat(s) matching that of the layers to be operated upon.
`idx`	integer vector Index into the list of layers used to select the layers to be operated upon.
`object`	a ggplot layer created by a `geom_` or `stat_` function or a list of such layers or an empty list.
`position`	character or interger, the position of the layer immediately above of which to move or append the moved or appended layers.
`shift`	integer.

Details

These functions must be used with care as they select all layers matching the provided geom, position or stat ggproto object class. Layers added with a stat do use a geom, and vice versa.

One and only one of `match_type` and `idx` must be passed a non-null argument.

In plots with several layers, it is possible that more than one layer matches the class name passed to `match_type`. It is also possible to pass a numeric vector with multiple indexes through parameter `idx`. In both cases multiple layers will be operated upon, but their relative positions will remain unchanged.

If a numeric vector with multiple position indexes is supplied as argument for `position`, the topmost position will be used. As indexing in R starts at 1, passing 0 or `"bottom"` as argument for `position` puts the moved or appended layer(s) behind all other layers (prepends the layer).

Value

An edited copy of `x` for `delete_layers`, `append_layers` and `move_layers`. An integer vector of indexes giving the positions of the matching layers in the list of layers contained in `x` in the case of `which_layers`.

Note

The functions described here are not expected to be useful in everyday plotting as one can more easily change the order in which layers are added to a ggplot. However, if one uses high level methods or functions that automatically produce a full plot using 'ggplot2' internally, one may need to add, move or delete layers so as to profit from such canned methods and retain enough flexibility.

References

https://stackoverflow.com/questions/13407236/remove-a-layer-from-a-ggplot2-chart

Examples

```
library(ggplot2)

df <- data.frame(
  gp = factor(rep(letters[1:3], each = 10)),
  y = rnorm(30)
)
p <- ggplot(df, aes(gp, y)) +
     geom_point() +
     stat_summary(fun.data = "mean_se", colour = "red")
p
delete_layers(p, "GeomPoint")
delete_layers(p, "StatSummary")
move_layers(p, "GeomPoint", position = "top")
move_layers(p, "GeomPointrange", position = "bottom")
move_layers(p, "StatSummary", position = "bottom")
move_layers(p, "GeomPointrange", position = 1L)
append_layers(p, geom_line(colour = "orange"), position = "bottom")
```

```
append_layers(p, geom_line(colour = "orange"), position = 1L)
extract_layers(p, "GeomPoint")
which_layers(p, "GeomPoint")
num_layers(p)
top_layer(p)
bottom_layer(p)
num_layers(ggplot())
top_layer(ggplot())
bottom_layer(ggplot())
```

`drop_vars` *Drop unused variables from data*

Description

Automatically remove unused variables from the data object embedded in a gg or `ggplot` object.

Usage

```
drop_vars(p, keep.vars = character(), guess.vars = TRUE)
```

Arguments

`p`	ggplot Plot object with embedded data.
`keep.vars`	character Names of unused variables to be kept.
`guess.vars`	logical Flag indicating whether to find used variables automatically.

Warning!

The current implementation drops variables only from the default data object. Data objects within layers are not modified.

Note

This function is under development and not yet thoroughly tested! It is a demonstration of how one can manipulate the internals of `ggplot` objects. This function may stop working after some future update to the 'ggplot2' package.

Rather than using this function after creating the `ggplot` object it may be more efficient to extract the variables of interest and pass a data frame containing only these to the `ggplot()` constructor.

Examples

```
library(ggplot2)

p <- ggplot(mpg, aes(factor(year), (cty + hwy) / 2)) +
  geom_boxplot() +
  facet_grid(. ~ class)
p

p.dp <- drop_vars(p)
p.dp

object.size(p)
object.size(p.dp)

names(p$data)
names(p.dp$data)
```

---

geom_debug *Geom which prints input data to console*

Description

The debug geom is used to print to the console a summary of the data being received by geoms as input `data` data frame.

Usage

```
geom_debug(mapping = NULL, data = NULL, stat = "identity",
  summary.fun = tibble::as_tibble, summary.fun.args = list(),
  print.fun = print, print.fun.args = list(), position = "identity",
  na.rm = FALSE, show.legend = FALSE, inherit.aes = TRUE, ...)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), is combined with the default mapping at the top level of the plot. You only need to supply `mapping` if there isn't a mapping defined for the plot.
`data`	A data frame. If specified, overrides the default data frame defined at the top level of the plot.
`stat`	The statistical transformation to use on the data for this layer, as a string.
`summary.fun`	A function used to print the `data` object received as input.
`summary.fun.args`	A list of additional arguments to be passed to `summary.fun`.
`print.fun`	A function used to print the value returned by `summary.fun`.
`print.fun.args`	A list of additional arguments to be passed to `print.fun`.

position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing 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.
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. borders.
...	other arguments passed on to layer. There are three types of arguments you can use here: • Aesthetics: to set an aesthetic to a fixed value, like color = "red" or size = 3. • Other arguments to the layer, for example you override the default stat associated with the layer. • Other arguments passed on to the stat.

Details

It can be useful when debugging the code of statistics or to learn how the stats and geoms work in 'ggplot2' (>= 2.0.0).

Note

This _geom_ is very unusual in that it does not produce visible graphic output. It only returns a grid::grid_null() grob (graphical object).

---

geom_null *A null geom or 'non-op' geom.*

Description

The null geom can be used to silence graphic output from a stat, such as stat_debug_group() and stat_debug_panel() defined in this same package. No visible graphical output is returned. An invisible grid::grid_null() grob is returned instead.

Usage

```
geom_null(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", na.rm = FALSE, show.legend = FALSE,
  inherit.aes = TRUE, ...)
```

Arguments

mapping	Set of aesthetic mappings created by aes or aes_. If specified and inherit.aes = TRUE (the default), is combined with the default mapping at the top level of the plot. You only need to supply mapping if there isn't a mapping defined for the plot.
data	A data frame. If specified, overrides the default data frame defined at the top level of the plot.
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing 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.
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. borders.
...	other arguments passed on to layer. There are three types of arguments you can use here: • Aesthetics: to set an aesthetic to a fixed value, like color = "red" or size = 3. • Other arguments to the layer, for example you override the default stat associated with the layer. • Other arguments passed on to the stat.

Note

This _geom_ is very unusual in that it does not produce visible graphic output. It only returns a grid::grid_null() grob (graphical object).

Although this geom accepts for consistency all the same parameters as normal geoms, these have no effect on the output, except for show.legend.

---

stat_debug_group *Print to console data received by the compute group function.*

Description

stat_debug reports all distinct values in group and PANEL, and nrow, ncol and the names of the columns or variables, and the class of x and y for each group in a ggplot as passed to the compute_group function in the ggproto object.

Usage

```
stat_debug_group(mapping = NULL, data = NULL, geom = "null",
  summary.fun = tibble::as_tibble, summary.fun.args = list(),
  position = "identity", na.rm = FALSE, show.legend = FALSE,
  inherit.aes = TRUE, ...)
```

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
summary.fun	A function used to print the data object received as input.
summary.fun.args	A list.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
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.
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. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Computed variables

x x at centre of range

y y at centre of range

nrow nrow() of data object

ncol ncol() of data object

colnames colnames() of data object

colclasses class() of x and y columns in data object

group all distinct values in group as passed in data object

PANEL all distinct values in PANEL as passed in data object

See Also

Other diagnosis functions: stat_debug_panel

Examples

```
library(ggplot2)
my.df <- data.frame(x = rep(1:10, 2),
                    y = rep(c(1,2), c(10,10)),
                    group = rep(c("A","B"), c(10,10)))
ggplot(my.df, aes(x,y)) + geom_point() + stat_debug_group()
ggplot(my.df, aes(x,y, colour = group)) + geom_point() + stat_debug_group()
ggplot(my.df, aes(x,y)) + geom_point() + facet_wrap(~group) + stat_debug_group()
```

`stat_debug_panel` *Print to console data received by the compute panel function.*

Description

`stat_debug` reports all distinct values in `group` and `PANEL`, and `nrow`, `ncol` and the names of the columns or variables, and the class of x and y for each panel in a ggplot as passed to the `compute_panel` function in the `ggproto` object.

Usage

```
stat_debug_panel(mapping = NULL, data = NULL, geom = "null",
  summary.fun = tibble::as_tibble, summary.fun.args = list(),
  position = "identity", na.rm = FALSE, show.legend = FALSE,
  inherit.aes = TRUE, ...)
```

Arguments

`mapping`	The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
`data`	A layer specific dataset - only needed if you want to override the plot defaults.
`geom`	The geometric object to use display the data
`summary.fun`	A function used to print the `data` object received as input.
`summary.fun.args`	A list.
`position`	The position adjustment to use for overlapping points on this layer
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.
`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.
`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. `borders`.
`...`	other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

Computed variables

x x at centre of range

y y at centre of range

nrow `nrow()` of `data` object

ncol `ncol()` of `data` object

colnames `colnames()` of `data` object

colclasses `class()` of `x` and `y` columns in `data` object

group all distinct values in group as passed in `data` object

PANEL all distinct values in PANEL as passed in `data` object

See Also

Other diagnosis functions: `stat_debug_group`

Examples

```
library(ggplot2)
my.df <- data.frame(x = rep(1:10, 2),
                    y = rep(c(1,2), c(10,10)),
                    group = rep(c("A","B"), c(10,10)))
ggplot(my.df, aes(x,y)) + geom_point() + stat_debug_panel()
ggplot(my.df, aes(x,y, colour = group)) + geom_point() + stat_debug_panel()
ggplot(my.df, aes(x,y)) + geom_point() + facet_wrap(~group) + stat_debug_panel()
```

Index