Package ‘gginnards’

Type Package

Title Explore the Innards of 'ggplot2' Objects

Version 0.1.2

Date 2023-05-24

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

Description Extensions to 'ggplot2' providing low-level debug tools: statistics and geometries echoing their data argument. Layer manipulation: deletion, insertion, extraction and reordering of layers. Deletion of unused variables from the data object embedded in "ggplot" objects.

License GPL (>= 2)

LazyLoad TRUE

ByteCompile TRUE

Depends R (>= 3.6.0), ggplot2 (>= 3.3.1)

Imports utils, methods, rlang (>= 0.4.0), stringr (>= 1.4.0), magrittr (>= 1.5), tibble (>= 2.1.0)

Suggests knitr (>= 1.24), rmarkdown (>= 1.14), sf (>= 0.9), pryr (>= 0.1.4)


BugReports https://github.com/aphalo/gginnards/issues

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2023-05-24 20:10:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>R Package</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>gginnards-package</td>
<td>2</td>
</tr>
<tr>
<td>delete_layers</td>
<td>3</td>
</tr>
<tr>
<td>drop_vars</td>
<td>6</td>
</tr>
<tr>
<td>geom_debug</td>
<td>8</td>
</tr>
<tr>
<td>geom_null</td>
<td>11</td>
</tr>
<tr>
<td>stat_debug_group</td>
<td>12</td>
</tr>
<tr>
<td>stat_debug_panel</td>
<td>15</td>
</tr>
<tr>
<td>str</td>
<td>18</td>
</tr>
</tbody>
</table>

### Description

Extensions to 'ggplot2' providing low-level debug tools: statistics and geometries echoing their data argument. Layer manipulation: deletion, insertion, extraction and reordering of layers. 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 extensions 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 very popular graphics package works internally. The vignettes provide examples of how to use these tools both for debugging and learning how ggplots are stored.

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.
- Functions for exploring and manipulating the data embedded in ggplot objects, including dropping unused variables.

### Author(s)

**Maintainer:** Pedro J. Aphalo <pedro.aphalo@helsinki.fi> (ORCID)

### References

Package 'tidyverse' web site at [https://www.tidyverse.org/](https://www.tidyverse.org/)
Package 'ggplot2' documentation at [https://ggplot2.tidyverse.org/](https://ggplot2.tidyverse.org/)
Package 'ggplot2' source code at [https://github.com/tidyverse/ggplot2](https://github.com/tidyverse/ggplot2)
delete_layers

See Also

Useful links:

- [https://www.r4photobiology.info](https://www.r4photobiology.info)
- [https://github.com/aphalo/gginnards](https://github.com/aphalo/gginnards)
- Report bugs at [https://github.com/aphalo/gginnards/issues](https://github.com/aphalo/gginnards/issues)

Examples

```r
# echo to the R console `data` as received by geoms
ggplot(mtcars, aes(cyl, mpg, color = factor(cyl))) +
  geom_point() +
  geom_debug()

# echo to the R console `data` as received by geoms
ggplot(mtcars, aes(cyl, mpg, colour = factor(cyl))) +
  stat_summary(fun.data = "mean_se") +
  stat_summary(fun.data = "mean_se", geom = "debug")

# echo to the R console `data` received by `compute_panel()`
ggplot(mtcars, aes(cyl, mpg, color = factor(cyl))) +
  geom_point() +
  stat_debug_panel()

# echo to the R console `data` received by `compute_group()`
ggplot(mtcars, aes(cyl, mpg, color = factor(cyl))) +
  geom_point() +
  stat_debug_group()
```

delete_layers  

Layer manipulation.

Description

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

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 integer, 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 top-most 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.
delete_layers

References

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

Examples

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")
ggplot(df, aes(gp, y)) + 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())

if (requireNamespace("sf", quietly = TRUE)) {
    nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
    nc_3857 <- sf::st_transform(nc, 3857)

    p.sf1 <- ggplot() +
      geom_sf(data = nc)
    p.sf1
    num_layers(p.sf1)
top_layer(p.sf1)

    append_layers(p.sf1,
      geom_sf(data = nc_3857, colour = "red", fill = NA),
      position = "top")

    p.sf2 <- ggplot() +
      geom_sf(data = nc) +
      geom_sf(data = nc_3857, colour = "red", fill = NA)
    p.sf2
    num_layers(p.sf2)
top_layer(p.sf2)
delete_layers(p.sf2, idx = 2L)
drop_vars

Explore and manipulate the embedded data.

Description

Automatically remove unused variables from the "default" data object embedded in a gg or ggplot object with drop_vars(). Explore data variables and their use with mapped_vars(), data_vars() and data_attributes().

Usage

drop_vars(p, keep.vars = character(), guess.vars = TRUE)
mapped_vars(p, invert = FALSE)
data_vars(p)
data_attributes(p)

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.
invert logical If TRUE return indices for elements of data that are not mapped to any aesthetic or facet.

Value

A "ggplot" object that is a copy of p but containing only a subset of the variables in its default data.
character vector with names of mapped variables in the default data object.
character vector with names of all variables in the default data object.
list containing all attributes of the default data object.
**Warning!**

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

**Note**

These functions are under development and not yet thoroughly tested! They are a demonstration of how one can manipulate the internals of ggpplot objects created with 'ggplot2' versions 3.1.0 and later. These functions may stop working after some future update to the 'ggplot2' package. Although I will maintain this package for use in some of my other packages, there is no guarantee that I will be able to achieve this transparently.

Obviously, rather than using function `drop_vars()` after creating the ggplot object it is usually more efficient to select the variables of interest and pass a data frame containing only these to the `ggplot()` constructor.

**Examples**

```r
library(ggplot2)

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

mapped_vars(p) # those in use
mapped_vars(p, invert = TRUE) # those not used

p.dp <- drop_vars(p) # we drop unused vars

# number of columns in the data member
ncol(p$data)
ncol(p.dp$data)

# which vars are in the data member
data_vars(p)
data_vars(p.dp)

# which variables in data are used in the plot
mapped_vars(p)
mapped_vars(p.dp)

# the plots identical
p
p.dp

# structure and size of p
str(p, max.level = 0)
str(p.dp, max.level = 0) # smaller in size

# structure and size of p['data']
str(p, components = "data")
str(p.dp, components = "data") # smaller in size
```
# shape data
if (requireNamespace("sf", quietly = TRUE)) {
  nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)

  p.sf <- ggplot(data = nc) +
  geom_sf()
  p.sf
  mapped_vars(p.sf)
  drop_vars(p.sf)
}

---

**geom_debug**

Geom that 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**

```r
geom_debug(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  summary.fun = "head",
  summary.fun.args = list(),
  parse = NULL,
  nudge_x = 0,
  nudge_y = 0,
  position = "identity",
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE,
  ...
)
```

```r
geom_debug_npc(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  summary.fun = "head",
  summary.fun.args = list(),
  parse = NULL,
  nudge_x = 0,
  nudge_y = 0,
```
position = "identity",
na.rm = FALSE,
show.legend = FALSE,
inherit.aes = TRUE,
...
)

Arguments

mapping  Set of aesthetic mappings created by \texttt{aes} or \texttt{aes...}. If specified and \texttt{inherit.aes} = \texttt{TRUE} (the default), is combined with the default mapping at the top level of the plot. You only need to supply \texttt{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 The name of a function as a character string (or a function) to be used to summarize the data object received as input before printing it.
summary.fun.args A list of additional arguments to be passed to \texttt{summary.fun}.
parse Ignored. Helps avoid warnings.
nudge_x, nudge_y Horizontal and vertical adjustments to nudge the starting position. The units for \texttt{nudge_x} and \texttt{nudge_y} are the same as for the data units on the x-axis and y-axis.
position Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm If \texttt{FALSE} (the default), removes missing values with a warning. If \texttt{TRUE} silently removes missing values.
show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes.
inherit.aes If \texttt{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. \texttt{borders}.
...
other arguments passed on to \texttt{layer}. There are three types of arguments you can use here:

• Aesthetics: to set an aesthetic to a fixed value, like \texttt{color = "red"} or \texttt{size = 3}.
• Other arguments to the layer, for example you override the default \texttt{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).
The intended use of this geometry is to help explore the data as they are added to a plot layer. It is very unusual in that it does not produce visible graphic output. It only returns a grid.null() grob (graphical object) when the plot is rendered. Also, differently to normal geometries, it passes its data input as argument to the first parameter of the function passed as argument to summary.fun. The value returned by this function is then printed to the R console. If summary.fun = NULL, the whole data object is printed.

Nudging with nudge_x and nudge_y behave as in geom_text. Arguments passed to position are obeyed. So the effects of positions are reflected in the data object printed or summarized to the R console. The argument passed to parse is currently ignored.

Many aesthetics are defined as optional so that they are accepted silently by geom_debug() and handled by 'ggplot2' as usual.

If the argument passed to summary.fun() is a character string, the name will appear in the header of the printout. However, the function must be available at the time the plot is rendered. If a function is passed as argument, its definition will be saved as part of the "gg" object.

Value

The panel function of this geometry always returns a nullGrob, the legend is is also set to nullGrob. This geometry used for its text printing side effect.

See Also

To access data, scales and grobs in a built ggplot, see ggplot_build.

Examples

```r
# echo to the R console |code(data) as received by geoms
ggplot(mtcars, aes(cyl, mpg, color = factor(cyl))) +
  geom_point() +
  geom_debug()

ggplot(mtcars, aes(cyl, mpg, color = factor(cyl))) +
  geom_point() +
  geom_debug(summary.fun = "head", summary.fun.args = list(n = 3))

ggplot(mtcars, aes(cyl, mpg, color = factor(cyl))) +
  geom_point() +
  geom_debug(summary.fun = "nrow")

ggplot(mtcars, aes(cyl, mpg, color = factor(cyl))) +
  geom_point() +
  geom_debug(summary.fun = "attributes")

# echo to the R console |code(data) as received by geoms
ggplot(mtcars, aes(cyl, mpg, colour = factor(cyl))) +
  stat_summary(fun.data = "mean_se") +
  stat_summary(fun.data = "mean_se", geom = "debug", summary.fun = NULL)

# shape data is not passed to geometries or statistics
if (requireNamespace("sf", quietly = TRUE)) {

```
geom_null

```r
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)

ggplot(data = nc) +
  geom_sf(color = "darkblue", fill = "white") +
  geom_debug()
```

data:image/png

---

A null geom or 'no-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

```r
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`. If specified and `inherit.aes = TRUE` (the default), are 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.

Value

A plot layer instance. Mainly used for the side-effect of printing to the console the data object.

Note

This geom is very unusual in that it does not produce visible graphic output. It only returns a grid.null grob (graphical object). However, it accepts for consistency all the same parameters as normal geoms, which have no effect on the graphical output, except for show.legend.

Examples

```r
ggplot(mtcars) +
  geom_null()

ggplot(mtcars, aes(cyl, mpg)) +
  geom_null()

# shape data
if (requireNamespace("sf", quietly = TRUE)) {
  nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)

  ggplot(data = nc) +
    geom_null()
}
```

---

**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.
**stat_debug_group**

**Usage**

```r
stat_debug_group(
  mapping = NULL,
  data = NULL,
  geom = "null",
  summary.fun = "head",
  summary.fun.args = list(),
  geom.summary.fun = NULL,
  geom.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, geom.summary.fun**: A function used to print the data object received as input.
- **summary.fun.args, geom.summary.fun.args**: A named 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.

**Details**

This stat is meant to be used for the side-effect of printing to the console the data object received as input by the `compute_grroup()` function, or a summary of it. This is the same as for any other statistics passed the same arguments (including defaults that may need to be overridden if they differ).

In principle any geom can be passed as argument to override "null". Keep in mind that this stat sets default mappings only for the x and y aesthetics: `geom_debug()` and `geom_text()` are useful.
Value

A tibble with a summary of the data received, which is not printed by default using `geom_null()`. Can be printed by passing `geom = "debug"`.

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

```r
my.df <- data.frame(x = rep(1:10, 2),
                     y = rep(c(1,2), c(10,10)) + rnorm(20),
                     group = rep(c("A","B"), c(10,10)))

# by default head() is used to show the top rows of data object
# and geom_null() to silence the data returned by the stat
ggplot(my.df, aes(x,y)) +
  geom_point() +
  stat_debug_group()

# geom_debug prints the data returned by the stat
ggplot(my.df, aes(x,y)) +
  geom_point() +
  stat_debug_group(geom = "debug")

# to print only the the data returned by the stat
# we pass as summary function a function that always returns NULL
ggplot(my.df, aes(x,y)) +
  geom_point() +
  stat_debug_group(geom = "debug",
                  summary.fun = function(x) {NULL})

ggplot(my.df, aes(x,y)) +
  geom_point() +
  stat_debug_group(aes(label = sprintf("nrow = %i, ncol = %i, colnames: %s",
                                 after_stat(nrow),
                                 after_stat(ncol),
                                 after_stat(colnames))))
```
stat_debug_panel

stat_debug_panel(  
  mapping = NULL,  
  data = NULL,  
  geom = "null",  
  summary.fun = "head",  
  summary.fun.args = list(),  
  geom.summary.fun = NULL,  
  geom.summary.fun.args = list(),  
  position = "identity",  
  na.rm = FALSE,  
  show.legend = FALSE,  
  geom = "text")

# here we show all the data object
ggplot(my.df, aes(x,y)) +
  geom_point() +
  stat_debug_group(summary.fun = NULL)

ggplot(my.df, aes(x,y)) +
  geom_point() +
  stat_debug_group(summary.fun = "nrow")

# with grouping
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()

# by default head() is used to show the top rows of data object
ggplot(my.df, aes(group,y)) +
  geom_point() +
  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 = "head",  
  summary.fun.args = list(),  
  geom.summary.fun = NULL,  
  geom.summary.fun.args = list(),  
  position = "identity",  
  na.rm = FALSE,  
  show.legend = FALSE,
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, geom.summary.fun
  A function used to print the data object received as input.

summary.fun.args, geom.summary.fun.args
  A named 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.

Details

This stat is meant to be used for the side-effect of printing to the console the data object received as input by the `compute_panel()` function, or a summary of it. This is the same as for any other statistics passed the same arguments (including defaults that may need to be overridden if they differ).

In principle any geom can be passed as argument to override "null". Keep in mind that this stat sets default mappings only for the x and y aesthetics: `geom_debug()` and `geom_text()` are useful.

Value

A tibble with a summary of the data received, which is not printed by default using `geom_null()`. Can be printed by passing `geom = "debug"`.

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

my.df <- data.frame(x = rep(1:10, 2),
                    y = rep(c(1,2), c(10,10)) + rnorm(20),
                    group = rep(c("A","B"), c(10,10)))

# by default head() is used to show the top rows of data object
# and geom_null() to silence the data returned by the stat
ggplot(my.df, aes(x,y)) +
      geom_point() +
      stat_debug_panel()

# geom_debug prints the data returned by the stat
ggplot(my.df, aes(x,y)) +
      geom_point() +
      stat_debug_panel(geom = "debug")

# to print only the the data returned by the stat
# we pass as summary function a function that always returns NULL
ggplot(my.df, aes(x,y)) +
      geom_point() +
      stat_debug_panel(geom = "debug",
                       summary.fun = function(x) {NULL})

ggplot(my.df, aes(x,y)) +
      geom_point() +
      stat_debug_panel(aes(label = sprintf("nrow = %i, ncol = %i, colnames: %s",
                                          after_stat(nrow),
                                          after_stat(ncol),
                                          after_stat(colnames)),
                      geom = "text")

# here we show all the data object
ggplot(my.df, aes(x,y)) +
      geom_point() +
      stat_debug_panel(summary.fun = NULL)

ggplot(my.df, aes(x,y)) +
      geom_point() +
      stat_debug_panel(summary.fun = "nrow")

# with grouping
str

Show the structure of a ggplot object.

Description

A `str()` method tailored to objects of class "ggplot". It adds to the output the size of the object, and the ability to subset individual components.

Usage

```r
## S3 method for class 'ggplot'
str(
  object,
  ..., 
  max.level = 1,
  components = TRUE,
  vec.len = 2,
  list.len = 99,
  give.attr = FALSE,
  comp.str = "$ \cdot ",
  nest.lev = 0,
  indent.str = paste(rep.int(" ", max(0, nest.lev + 1)), collapse = " ", collapse = ".")
)
```

Arguments

- **object**: ggplot Plot object with embedded data.
- **...**: accept additional parameter arguments
- **max.level**: integer Maximum depth of recursion (of lists within lists ...) to be printed.
- **components**: Vector of components to print, as indexes into object.
vec.len integer Approximate maximum length allowed when showing the first few values of a vector.

list.len integer Maximum number of components to show of any list that will be described.

give.attr logical Flag, determining whether a description of attributes will be shown.

comp.str character String to be used for separating list components.

nest.lev numeric current nesting level in the recursive calls to str().

indent.str character String used for each level of indentation.

size logical Flag, should the size of the object in bytes be printed?

Value

A NULL is returned invisibly. While a description of the structure of p or its components will be printed in outline form as a "side-effect", with indentation for each level of recursion, showing the internal storage mode, class(es) if any, attributes, and first few elements of each data vector. By default each level of list recursion is indicated and attributes enclosed in angle brackets.

Note

In the case of objects with a nested structure str() is called recursively and dispatched according to the class of each nested member.

See Also

A summary method for class ggplot is defined by package 'ggplot2'. Method summary() provides a more compact description of "ggplot" objects than method str(). Here we provide a wrapper on R's str() with different default arguments. A summary does not directly describe how the different components of an R object are stored, while the structure does.

Examples

```r
p <- ggplot(mpg, aes(factor(year), (cty + hwy) / 2)) +
  geom_boxplot() +
  geom_point(color = "red") +
  facet_grid(. ~ class) +
  ggtitle("Example plot")

p

# str(p) vs. summary(p)
str(p)
summary(p) # from package 'ggplot2'

# structure of p at 2 levels of nesting
str(p, max.level = 2, size = FALSE)

# top level structure and size of p
str(p, max.level = 0)
```
# names of ggplot members
names(p)

# structure and size of p["data"]
str(p, max.level = 2, components = "data")

# structure and size of p["layers"]
str(p, max.level = 1, components = "layers")
Index

* diagnosis functions
  - stat_debug_group, 12
  - stat_debug_panel, 15

aes, 9, 11, 13, 16
aes_, 9, 13, 16
append_layers (delete_layers), 3

borders, 9, 11, 13, 16
bottom_layer (delete_layers), 3

data_attributes (drop_vars), 6
data_vars (drop_vars), 6
delete_layers, 3
drop_vars, 6

extract_layers (delete_layers), 3

geom_debug, 8
geom_debug_npc (geom_debug), 8
geom_null, 11
geom_text, 10
gginnards (gginnards-package), 2
gginnards-package, 2
ggplot_build, 10
grid.null, 12

layer, 9, 12, 13, 16
mapped_vars (drop_vars), 6
move_layers (delete_layers), 3

nullGrob, 10
num_layers (delete_layers), 3

shift_layers (delete_layers), 3
stat_debug_group, 12, 17
stat_debug_panel, 14, 15
str, 18
summary, 19

top_layer (delete_layers), 3

which_layers (delete_layers), 3