Package 'ggpmisc'

April 2, 2019

Type Package

Title Miscellaneous Extensions to 'ggplot2'

Version 0.3.1

Date 2019-04-01

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

Description Extensions to 'ggplot2' respecting the grammar of graphics paradigm. Specialization of method ggplot(): accept and convert on the fly time series data. Geom: ``table", ``plot" and ``grob" add insets to plots using native data coordinates, while ``table_npc", ``plot_npc" and ``grob_npc" do the same using ``npc" coordinates through new aesthetics ``npcx" and ``npcy". Statistics: locate and tag peaks and valleys; count observations in different quadrants of a plot; select observations based on 2D density; label with the equation of a polynomial fitted with lm() or other types of models; labels with P-value, R^2 or adjusted R^2 or information criteria for fitted models; label with ANOVA table for fitted models; label with summary for fitted models. Model fit classes for which suitable methods are provided by package 'broom' are supported.

License GPL (>= 2)

LazyData TRUE

LazyLoad TRUE

ByteCompile TRUE

Depends R (>= 3.4.0), ggplot2 (>= 3.1.0)

Imports grid, rlang (>= 0.3.1), magrittr (>= 1.5), gridExtra (>= 2.3), scales (>= 1.0.0), MASS (>= 7.3-51.1), polynom (>= 1.3-9), splus2R (>= 1.2-2), tibble (>= 2.0.1), plyr (>= 1.8.4), dplyr (>= 0.8.0.1), xts (>= 0.11-2), zoo (>= 1.8-4), broom (>= 0.5.1), lubridate (>= 1.7.4), stringr (>= 1.4.0)

Suggests knitr (>= 1.22), rmarkdown (>= 1.12), nlme (>= 3.1-137), ggrepel (>= 0.8.0), magick (>= 2.0)

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

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

Encoding UTF-8

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2019-04-02 08:30:12 UTC

R topics documented:

ggpmisc-package *ggpmisc: Miscellaneous Extensions to 'ggplot2'*

Description

Extensions to 'ggplot2' respecting the grammar of graphics paradigm. Specialization of method ggplot(): accept and convert on the fly time series data. Geom: "table", "plot" and "grob" add insets to plots using native data coordinates, while "table_npc", "plot_npc" and "grob_npc" do the same using "npc" coordinates through new aesthetics "npcx" and "npcy". Statistics: locate and tag peaks and valleys; count observations in different quadrants of a plot; select observations based on 2D density; label with the equation of a polynomial fitted with lm() or other types of models; labels with P-value, R^2 or adjusted R^2 or information criteria for fitted models; label with ANOVA table for fitted models; label with summary for fitted models. Model fit classes for which suitable methods are provided by package 'broom' are supported.

Details

The new facilities for cleanly defining new stats and geoms added to 'ggplot2' in version 2.0.0 and the support for nested tibbles and new syntax for mapping computed values to aesthetics added to 'ggplot2' in version 3.0.0 are used in this package's code. This means that 'ggpmisc' (>= 0.3.0) requires version 3.0.0 or later of ggplot2 while 'ggpmisc' (< 0.3.0) requires version 2.0.0 or later of ggplot2.

Extensions provided:

- Function for conversion of time series data into tibbles that can be plotted with ggplot.
- `ggplot()` method for time series data.
- Stats for locating and tagging "peaks" and "valleys" (local or global maxima and minima).
- Stat for generating labels from a `lm()` model fit, including formatted equation. By default labels are expressions but tikz device is supported optionally with LaTeX formatted labels.
- Stats for extracting information from a any model fit supported by package 'broom'.
- Stats for filtering-out/filtering-in observations in regions of a panel or group where the density of observations is high.
- Geom for annotating plots with tables.

The stats for peaks and valleys are coded so as to work correctly both with numeric and POSIXct variables mapped to the x aesthetic. Special handling was needed as text labels are generated from the data.

Warning!

`geom_null()`, `stat_debug_group()`, `stat_debug_panel()`, `geom_debug()`, `append_layers()`, `bottom_layer()`, `delete_layers()`, `extract_layers()`, `move_layers()`, `num_layesr()`, `shift_layers()`, `top_layer()` and `which_layers()` have been moved from package 'ggpmisc' into their own seperate package '`gginnards-package`.

Acknowledgements

We thank Kamil Slowikowski not only for contributing ideas and code examples to this package but also for adding new features to his package 'ggrepel' that allow new use cases for `stat_dens2d_labels` from this package.

Note

The signatures of `stat_peaks()` and `stat_valleys()` are identical to those of `stat_peaks` and `stat_valleys` from package `photobiology` but the variables returned are a subset as values related to light spectra are missing. Furthermore the stats from package `ggpmisc` work correctly when the x aesthetic uses a date or datetime scale, while those from package `photobiology` do not generate correct labels in this case.

Author(s)

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

Other contributors:

- Kamil Slowikowski [contributor]

References

Package suite 'r4photobiology' web site at https://www.r4photobiology.info/
Package 'ggplot2' documentation at https://ggplot2.tidyverse.org/
Package 'ggplot2' source code at https://github.com/hadley/ggplot2

See Also

Useful links:

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

Examples

```
library(tibble)

ggplot(lynx, as.numeric = FALSE) + geom_line() +
stat_peaks(colour = "red") +
  stat_peaks(geom = "text", colour = "red", angle = 66,
             hjust = -0.1, x.label.fmt = "%Y") +
  ylim(NA, 8000)

formula <- y ~ poly(x, 2, raw = TRUE)
ggplot(cars, aes(speed, dist)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..eq.label..), formula = formula,
               parse = TRUE)
```

```
formula <- y ~ x
ggplot(PlantGrowth, aes(group, weight)) +
  stat_summary(fun.data = "mean_se") +
  stat_fit_tb(method = "lm",
              method.args = list(formula = formula),
              tb.type = "fit.anova") +
  theme_classic()
```

---

geom_grob *Inset graphical objects*

Description

geom_grob adds a Grob as inset to the ggplot using syntax similar to that of geom_label.

Usage

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

geom_grob_npc(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., na.rm = FALSE, show.legend = FALSE,
  inherit.aes = 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.
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.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
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.

Details

Note the "width" and "height" like of a text element are 0, so stacking and dodging Grobs will not work by default, and axis limits are not automatically expanded to include all inset Grobs. Obviously, Grobs do have height and width, but they are in physical units, not data units. The amount of space they occupy on the main plot is constant in data units.

Alignment

You can modify table alignment with the `vjust` and `hjust` aesthetics. These can either be a number between 0 (right/bottom) and 1 (top/left) or a character ("left", "middle", "right", "bottom", "center", "top").

Inset size

You can modify inset plot size with the `vp.width` and `vp.height` aesthetics. These can be a number between 0 (smallest posisble inset) and 1 (whole plotting area width or height). The default value for for both of these aesthetics is 1/3.

Note

These geoms work only with tibbles as `data`, as they expects a list of graphics objects ("grob") to be mapped to the `label` aesthetic. Aesthetics mappings in the inset plot are independent of those in the base plot.

In the case of `geom_grob()`, `x` and `y` aesthetics determine the position of the whole inset grob, similarly to that of a text label, justification is interpreted as indicating the position of the grob with respect to the x and y coordinates in the data, and `angle` is used to rotate the plot as a whole.

In the case of `geom_grob_npc()`, `npcx` and `npcy` aesthetics determine the position of the whole inset plot, similarly to that of a text label, justification is interpreted as indicating the position of the grob with respect to the x and y coordinates in "npc" units, and `angle` is used to rotate the plot as a whole.

`annotate()` **cannot be used with** `geom = "grob"`. Use `annotation_custom` directly when adding inset plots as annotations.

References

The idea of implementing a `geom_custom()` for grobs has been discussed as an issue at https://github.com/tidyverse/ggplot2/issues/1399.

Examples

```
library(tibble)
df <- tibble(x = 2, y = 15, grob = list(grid::circleGrob(r = 0.2)))
ggplot(data = mtcars, aes(wt, mpg)) +
  geom_point(aes(colour = factor(cyl))) +
  geom_grob(data = df, aes(x, y, label = grob))
```

geom_label_npc *Text with Normalised Parent Coordinates*

Description

'geom_text_npc()' adds text directly to the plot. 'geom_label_npc()' draws a rectangle behind the text, making it easier to read. The difference is that x and y mappings are expected to be given in 'npc' graphic units. They are intended to be used for positioning text relative to the physical dimensions of a plot. This can be achieved with 'annotate()' except when facetting is used.

Usage

```
geom_label_npc(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., parse = FALSE, nudge_x = 0,
  nudge_y = 0, label.padding = unit(0.25, "lines"),
  label.r = unit(0.15, "lines"), label.size = 0.25, na.rm = FALSE,
  show.legend = FALSE, inherit.aes = FALSE)

geom_text_npc(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., parse = FALSE, nudge_x = 0,
  nudge_y = 0, check_overlap = FALSE, na.rm = FALSE,
  show.legend = FALSE, inherit.aes = 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.
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.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.
nudge_x, nudge_y	Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales.
label.padding	Amount of padding around label. Defaults to 0.25 lines.
label.r	Radius of rounded corners. Defaults to 0.15 lines.
label.size	Size of label border, in mm.
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.
check_overlap	If 'TRUE', text that overlaps previous text in the same layer will not be plotted.

Details

Note that the "width" and "height" of a text element are 0, so stacking and dodging text will not work by default, and axis limits are not automatically expanded to include all text. Obviously, labels do have height and width, but they are physical units, not data units. The amount of space they occupy on the plot is not constant in data units: when you resize a plot, labels stay the same size, but the size of the axes changes.

'geom_text_npc()' and 'geom_label_npc()' add labels for each row in the data, even if coordinates x, y are set to single values in the call to 'geom_label_npc()' or 'geom_text_npc()'. To add labels at specified points use [annotate()] with 'annotate(geom = "text_npc", ...)' or 'annotate(geom = "label_npc", ...)'.

'geom_label_npc()'

Currently 'geom_label_npc()' does not support the 'angle' aesthetic and is considerably slower than 'geom_text_npc()'. The 'fill' aesthetic controls the background colour of the label.

Alignment

You can modify text alignment with the 'vjust' and 'hjust' aesthetics. These can either be a number between 0 (right/bottom) and 1 (top/left) or a character ('"left"', '"middle"', '"right"', '"bottom"', '"center"', '"top"'). There are two special alignments: '"inward"' and '"outward"'. Inward always aligns text towards the center, and outward aligns it away from the center.

Note

This geom is identical to 'ggplot2' geom_text() except that it interprets x and y positions in npc units. It translates x and y coordinates from npc units to native data units and calls functions from 'ggplot2"s GeomText().

See Also

geom_text

Examples

```
df <- data.frame(
  x = c(0, 0, 1, 1, 0.5),
  x.chr = c("left", "left", "right", "right", "center"),
  y = c(0, 1, 0, 1, 0.5),
  y.chr = c("bottom", "top", "bottom", "top", "middle"),
```

```
  text = c("bottom-left", "top-left", "bottom-right", "top-right", "center-middle")
)
ggplot(df) +
  geom_text_npc(aes(npcx = x, npcy = y, label = text))

ggplot(df) +
  geom_text_npc(aes(npcx = x.chr, npcy = y.chr, label = text))

ggplot(data = mtcars, mapping = aes(wt, mpg)) +
  geom_point() +
  geom_text_npc(data = df, aes(npcx = x, npcy = y, label = text))

ggplot(data = mtcars, mapping = aes(wt, mpg)) +
  geom_point() +
  geom_text_npc(data = df, aes(npcx = x, npcy = y, label = text)) +
  expand_limits(y = 40, x = 6)

ggplot(data = mtcars) +
  geom_point(mapping = aes(wt, mpg)) +
  geom_label_npc(data = df, aes(npcx = x, npcy = y, label = text))
```

`geom_plot` *Inset plots*

Description

`geom_plot` adds ggplot objects as insets to the base ggplot, using syntax similar to that of `geom_label`.

Usage

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

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

...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
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.

Details

Note the "width" and "height" like of a text element are 0, so stacking and dodging inset plots will not work by default, and axis limits are not automatically expanded to include all inset plots. Obviously, plots do have height and width, but they are physical units, not data units. The amount of space they occupy on the main plot is not constant in data units of the base plot: when you modify scale limits, inset plots stay the same size relative to the physiscal size of the base plot.

Inset alignment

You can modify inset plot alignment with the vjust and hjust aesthetics. These can either be a number between 0 (right/bottom) and 1 (top/left) or a character ("left", "middle", "right", "bottom", "center", "top"). The angle aesthetics can be used to rotate the inset plots.

Inset size

You can modify inset plot size with the vp.width and vp.height aesthetics. These can be a number between 0 (smallest posisble inset) and 1 (whole plotting area width or height). The default value for for both of these aesthetics is 1/3.

Known problem!

In some cases when explicit coordinates are added to the inner plot, it may be also necessary to add explicitly coordinsates to the outer plots.

Note

These geoms work only with tibbles as data, as they expects a list of ggplots ("gg" objects) to be mapped to the label aesthetic. Aesthetics mappings in the inset plot are independent of those in the base plot.

In the case of geom_plot(), x and y aesthetics determine the position of the whole inset plot, similarly to that of a text label, justification is interpreted as indicating the position of the plot with respect to the x and y coordinates in the data, and angle is used to rotate the plot as a whole.

In the case of geom_plot_npc(), npcx and npcy aesthetics determine the position of the whole inset plot, similarly to that of a text label, justification is interpreted as indicating the position of the plot with respect to the x and y coordinates in "npc" units, and angle is used to rotate the plot as a whole.

annotate() **cannot be used with** geom = "plot". Use annotation_custom directly when adding inset plots as annotations.

References

The idea of implementing a geom_custom() for grobs has been discussed as an issue at https://github.com/tidyverse/ggplot2/issues/1399.

Examples

```
# inset plot with enlarged detail from a region of the main plot
library(tibble)
p <-
  ggplot(data = mtcars, mapping = aes(wt, mpg)) +
  geom_point()

df <- tibble(x = 0.01, y = 0.01,
             plot = list(p +
                         coord_cartesian(xlim = c(3, 4),
                                         ylim = c(13, 16)) +
                         labs(x = NULL, y = NULL) +
                         theme_bw(10)))
p +
  expand_limits(x = 0, y = 0) +
  geom_plot_npc(data = df, aes(npcx = x, npcy = y, label = plot))
```

---

geom_table *Inset tables*

Description

geom_table adds a textual table directly to the ggplot using syntax similar to that of geom_label while geom_table_npc is similar to geom_label_npc in that x and y coordinates are given in npc units.

Usage

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

geom_table_npc(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., parse = FALSE, na.rm = FALSE,
  show.legend = FALSE, inherit.aes = 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.
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.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.
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.

Details

Note the "width" and "height" like of a text element are 0, so stacking and dodging tables will not work by default, and axis limits are not automatically expanded to include all tables. Obviously, tables do have height and width, but they are physical units, not data units. The amount of space they occupy on that plot is not constant in data units: when you resize a plot, tables stay the same size, but the size of the axes changes.

Alignment

You can modify table alignment with the vjust and hjust aesthetics. These can either be a number between 0 (right/bottom) and 1 (top/left) or a character ("left", "middle", "right", "bottom", "center", "top").

Inset size

You can modify inset table size with the size aesthetics, which determines the size of text within the table.

Note

These geoms work only with tibbles as data, as they expects a list of data frames or tibbles ("tb" objects) to be mapped to the label aesthetic. Aesthetics mappings in the inset plot are independent of those in the base plot.

In the case of geom_table(), x and y aesthetics determine the position of the whole inset table, similarly to that of a text label, justification is interpreted as indicating the position of the table with respect to the x and y coordinates in the data, and angle is used to rotate the plot as a whole.

In the case of geom_table_npc(), npcx and npcy aesthetics determine the position of the whole inset table, similarly to that of a text label, justification is interpreted as indicating the position of the table with respect to the x and y coordinates in "npc" units, and angle is used to rotate the plot as a whole.

annotate() **cannot be used with** geom = "table". Use annotation_custom directly when adding inset tables as annotations.

References

This geometry is inspired on answers to two questions in Stackoverflow. In contrast to these earlier examples, the current geom obeys the grammar of graphics, and attempts to be consistent with the behaviour of 'ggplot2' geometries. https://stackoverflow.com/questions/12318120/adding-table-within-the-plotting-region-of-a-ggplot-in-r https://stackoverflow.com/questions/25554548/adding-sub-tables-on-each-panel-of-a-facet-ggplot-in-r?

See Also

function tableGrob as it is used to construct the table.

Examples

```
library(dplyr)
library(tibble)
mtcars %>%
  group_by(cyl) %>%
  summarize(wt = mean(wt), mpg = mean(mpg)) %>%
  ungroup() %>%
  mutate(wt = sprintf("%.2f", wt),
         mpg = sprintf("%.1f", mpg)) -> tb
df <- tibble(x = 0.95, y = 0.95, tb = list(tb))
ggplot(mtcars, aes(wt, mpg, colour = factor(cyl))) +
  geom_point() +
  geom_table_npc(data = df, aes(npcx = x, npcy = y, label = tb),
                 hjust = 1, vjust = 1)
```

---

geom_x_margin_arrow *Reference arrows on the margins*

Description

Small arrows on plot margins can supplement a 2d display with annotations. Arrows can be used to highligth specific values along a margin. The geometries geom_x_margin_arrow() and geom_y_margin_arrow() behave similarly geom_vline() and geom_hline() and share their "double personality" as both annotations and geometries.

Usage

```
geom_x_margin_arrow(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., xintercept, sides = "b",
  arrow.length = 0.03, na.rm = FALSE, show.legend = FALSE,
  inherit.aes = FALSE)

geom_y_margin_arrow(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., yintercept, sides = "l",
  arrow.length = 0.03, na.rm = FALSE, show.legend = FALSE,
  inherit.aes = 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.
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.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
xintercept, yintercept	numeric Parameters that control the position of the marginal points. If these are set, data, mapping and show.legend are overridden.
sides	A string that controls which sides of the plot the rugs appear on. It can be set to a string containing any of ‘"trbl"‘, for top, right, bottom, and left.
arrow.length	numeric value expressed in npc units for the length of the arows inwards from the edge of the plotting area.
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.

Examples

```
p <- ggplot(mtcars, aes(wt, mpg)) +
  geom_point()
p
p + geom_x_margin_arrow(xintercept = 3.5)
p + geom_y_margin_arrow(yintercept = c(18, 28, 15))
p + geom_x_margin_arrow(data = data.frame(x = c(2.5, 4.5)),
                        mapping = aes(xintercept = x))
p + geom_x_margin_arrow(data = data.frame(x = c(2.5, 4.5)),
```

```
                        mapping = aes(xintercept = x),
                        sides="tb")
```

---

geom_x_margin_grob *Add Grobs on the margins*

Description

Marging points can supplement a 2d display with annotations. Marging points can highligth individual cases or values along a margin. The geometries geom_x_margin_grob() and geom_y_margin_grob() behave similarly geom_vline() and geom_hline() and share their "double personality" as both annotations and geometries.

Usage

```
geom_x_margin_grob(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., xintercept, sides = "b",
  grob.shift = 0, na.rm = FALSE, show.legend = FALSE,
  inherit.aes = FALSE)

geom_y_margin_grob(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., yintercept, sides = "l",
  grob.shift = 0, na.rm = FALSE, show.legend = FALSE,
  inherit.aes = 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.
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.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
xintercept, yintercept	numeric Parameters that control the position of the marginal points. If these are set, data, mapping and show.legend are overridden.
sides	A character string of length one that controls on which side of the plot the grob annotations appear on. It can be set to a string containing one of "t", "r", "b" or "l", for top, right, bottom, and left.
grob.shift	numeric value expressed in npc units for the shift of the marginal grob inwards from the edge of the plotting area.

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.

Examples

```
# We can add icons to the margin of a plot to signal events
```

---

geom_x_margin_point *Reference points on the margins*

Description

Marging points can supplement a 2d display with annotations. Marging points can highligth individual cases or values along a margin. The geometries geom_x_margin_point() and geom_y_margin_point() behave similarly geom_vline() and geom_hline() and share their "double personality" as both annotations and geometries.

Usage

```
geom_x_margin_point(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., xintercept, sides = "b",
  point.shift = 0.017, na.rm = FALSE, show.legend = FALSE,
  inherit.aes = FALSE)

geom_y_margin_point(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., yintercept, sides = "l",
  point.shift = 0.017, na.rm = FALSE, show.legend = FALSE,
  inherit.aes = 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.
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.

...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
xintercept, yintercept	numeric Parameters that control the position of the marginal points. If these are set, data, mapping and show.legend are overridden.
sides	A string that controls which sides of the plot the rugs appear on. It can be set to a string containing any of ‘"trbl"‘, for top, right, bottom, and left.
point.shift	numeric value expressed in npc units for the shift of the rug points inwards from the edge of the plotting area.
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.

Examples

```
p <- ggplot(mtcars, aes(wt, mpg)) +
  geom_point()
p
p + geom_x_margin_point(xintercept = 3.5)
p + geom_y_margin_point(yintercept = c(18, 28, 15))
p + geom_x_margin_point(data = data.frame(x = c(2.5, 4.5)),
                        mapping = aes(xintercept = x))
p + geom_x_margin_point(data = data.frame(x = c(2.5, 4.5)),
                        mapping = aes(xintercept = x),
                        sides="tb")
```

---

ggplot *Create a new ggplot plot from time series data*

Description

ggplot() initializes a ggplot object. It can be used to declare the input spectral object for a graphic and to optionally specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage

```
## S3 method for class 'ts'
ggplot(data, mapping = NULL, ..., time.resolution = "day",
  as.numeric = TRUE, environment = parent.frame())
```

```
## S3 method for class 'xts'
ggplot(data, mapping = NULL, ...,
  time.resolution = "day", as.numeric = TRUE,
  environment = parent.frame())
```

Arguments

data	Default spectrum dataset to use for plot. If not a spectrum, the methods used will be those defined in package ggplot2. See ggplot. If not specified, must be suppled in each layer added to the plot.
mapping	Default list of aesthetic mappings to use for plot. If not specified, in the case of spectral objects, a default mapping will be used.
...	Other arguments passed on to methods. Not currently used.
time.resolution	character The time unit to which the returned time values will be rounded.
as.numeric	logical If TRUE convert time to numeric, expressed as fractional calendar years.
environment	If an variable defined in the aesthetic mapping is not found in the data, ggplot will look for it in this environment. It defaults to using the environment in which ggplot() is called.

Details

ggplot() is typically used to construct a plot incrementally, using the + operator to add layers to the existing ggplot object. This is advantageous in that the code is explicit about which layers are added and the order in which they are added. For complex graphics with multiple layers, initialization with ggplot is recommended.

There are three common ways to invoke ggplot:

- ggplot(ts, aes(x, y, <other aesthetics>))
- ggplot(ts)

The first method is recommended if all layers use the same data and the same set of aesthetics, although this method can also be used to add a layer using data from another data frame. See the first example below. The second method specifies the default spectrum object to use for the plot, and the units to be used for y in the plot, but no aesthetics are defined up front. This is useful when one data frame is used predominantly as layers are added, but the aesthetics may vary from one layer to another. The third method specifies the default spectrum object to use for the plot, but no aesthetics are defined up front. This is useful when one spectrum is used predominantly as layers are added, but the aesthetics may vary from one layer to another.

Note

Current implementation does not merge default mapping with user supplied mapping. If user supplies a mapping, it is used as is. To add to the default mapping, aes() can be used by itself to compose the ggplot.

Examples

```
library(ggplot2)
ggplot(lynx) + geom_line()
```

Moved *Moved to package 'gginnards'*

Description

Some stats, geoms and the plot layer manipulation functions have been moved from package 'ggpmisc' to a separate new package called 'gginnards'.

Details

To continue using any of these functions and methods, simply run at the R prompt or add to your script `library(gginnards)`, after installing package 'gginnards'.

See Also

`gginnards-package`, `geom_null`, `stat_debug_group`, `stat_debug_panel`, `geom_debug` and `delete_layers`.

---

scale_continuous_npc *Position scales for continuous data (npcx & npcy)*

Description

'scale_npcx_continuous()' and 'scale_npcy_continuous()' are scales for continuous npcx and npcy aesthetics expressed in "npc" units. There are no variants. Obviously limits are always the full range of "npc" units and transformations meaningless. These scales are used by the newly defined aesthetics `npcx` and `npcy`.

Usage

```
scale_npcx_continuous(...)

scale_npcy_continuous(...)
```

Arguments

...	Other arguments passed on to 'continuous_scale()'

stat_apply_group *Apply a function to x or y values*

Description

stat_apply_group and stat_apply_panel apply functions to data. In most cases one should simply use transformations through scales or summary functions through stat_summary(). There are some computations that are not scale transformations but are not usual summaries either, the number of data values does not decrease. It is always possible to precompute quantities like cummulative sums or running medians, and normalizations it can be convenient to apply such functions on-the-fly to ensure that grouping is consistent between computations and aesthetics. One particularity of these statistics is that they can apply simulateneously different functions to x values and to y values when needed. In contrast geom_smooth applies a function that takes both x and y values as arguments.

Usage

```
stat_apply_group(mapping = NULL, data = NULL, geom = "line",
  .fun.x = NULL, .fun.x.args = list(), .fun.y = NULL,
  .fun.y.args = list(), position = "identity", na.rm = FALSE,
  show.legend = FALSE, inherit.aes = TRUE, ...)

stat_apply_panel(mapping = NULL, data = NULL, geom = "line",
  .fun.x = NULL, .fun.x.args = list(), .fun.y = NULL,
  .fun.y.args = list(), position = "identity", na.rm = FALSE,
  show.legend = FALSE, inherit.aes = TRUE, ...)
```

Arguments

mapping	The aesthetic mapping, usually constructed with 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
.fun.x, .fun.y	function to be applied or the name of the function to be applied as a character string. One and only one of these parameters should be passed a non-null argument.
.fun.x.args, .fun.y.args	additional arguments to be passed to the function as 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

The function to be applied is expected to be vectorized and to return a vector of the same length. The vector mapped to the x or y aesthetic is passed as the first positional argument to the call. The function must accept as first argument a vector or list that matches the data.

Computed variables

One x or y of is replaced by the vector returned by the applied function.

x x-value as returned by `.fun.x`

y y-value as returned by `.fun.y`

Note

This stat is at early stages of development and its interface may change at any time, or it may be even removed from the package without previous notice.

References

Answers question "R ggplot on-the-fly calculation by grouping variable" at https://stackoverflow.com/questions/51412522.

Examples

```
set.seed(123456)
df <- data.frame(X = rep(1:20,2),
                 Y = runif(40),
                 category = rep(c("A","B"), each = 20))
ggplot(df, aes(x = X, y = Y, colour = category)) +
  stat_apply_group(.fun.y = cumsum)
ggplot(df, aes(x = X, y = Y, colour = category)) +
  geom_point() +
  stat_apply_group(.fun.y = runmed, .fun.y.args = list(k = 5))
ggplot(df, aes(x = X, y = Y, colour = category)) +
  stat_apply_panel(.fun.y = function(x) {(x - min(x)) / (max(x) - min(x))})
```

stat_dens2d_filter *Filter observations by local density*

Description

stat_dens2d_filter Filters-out/filters-in observations in regions of a plot panel with high density of observations. stat_dens2d_filter_g does the filtering by group instead of by panel. This second stat is useful for highlighting observations, while the first one tends to be most useful when the aim is to prevent clashes among text labels.

Usage

```
stat_dens2d_filter(mapping = NULL, data = NULL, geom = "point",
  position = "identity", keep.fraction = 0.1, keep.number = Inf,
  keep.sparse = TRUE, na.rm = TRUE, show.legend = FALSE,
  inherit.aes = TRUE, h = NULL, n = NULL, ...)

stat_dens2d_filter_g(mapping = NULL, data = NULL, geom = "point",
  position = "identity", keep.fraction = 0.1, keep.number = Inf,
  keep.sparse = TRUE, na.rm = TRUE, show.legend = FALSE,
  inherit.aes = TRUE, h = NULL, n = NULL, ...)
```

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.
position	The position adjustment to use for overlapping points on this layer
keep.fraction	numeric [0..1].
keep.number	integer number of labels to keep.
keep.sparse	logical If false the observations from the densest regions are kept.
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.
h	vector of bandwidths for x and y directions. Defaults to normal reference bandwidth (see bandwidth.nrd). A scalar value will be taken to apply to both directions.
n	Number of grid points in each direction. Can be scalar or a length-2 integer vector

... 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

labels x at centre of range

See Also

kde2d used internally.

Examples

```
library(ggrepel)

random_string <- function(len = 6) {
paste(sample(letters, len, replace = TRUE), collapse = "")
}

# Make random data.
set.seed(1001)
d <- tibble::tibble(
  x = rnorm(100),
  y = rnorm(100),
  group = rep(c("A", "B"), c(50, 50)),
  lab = replicate(100, { random_string() })
)

ggplot(data = d, aes(x, y)) +
  geom_point() +
  stat_dens2d_filter(color = "red")

ggplot(data = d, aes(x, y)) +
  geom_point() +
  stat_dens2d_filter(color = "red", keep.fraction = 0.5)

ggplot(data = d, aes(x, y)) +
  geom_point() +
  stat_dens2d_filter(color = "red",
                     keep.fraction = 0.5,
                     keep.number = 12)

ggplot(data = d, aes(x, y, color = group)) +
  geom_point() +
  stat_dens2d_filter(shape = 1, size = 3, keep.fraction = 1/4)

ggplot(data = d, aes(x, y, color = group)) +
  geom_point() +
  stat_dens2d_filter_g(shape = 1, size = 3, keep.fraction = 1/4)

ggplot(data = d, aes(x, y, label = lab, color = group)) +
```

```
  geom_point() +
  stat_dens2d_filter(geom = "text")

ggplot(data = d, aes(x, y, label = lab, color = group)) +
  geom_point() +
  stat_dens2d_filter(geom = "text_repel")
```

`stat_dens2d_labels` *Reset labels of observations in high density regions*

Description

`stat_low_dens` Sets `labels` to NA in regions of a plot panel with high density of observations.

Usage

```
stat_dens2d_labels(mapping = NULL, data = NULL, geom = "text",
  position = "identity", keep.fraction = 0.1, keep.number = Inf,
  h = NULL, n = NULL, label.fill = "", na.rm = TRUE,
  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.
`position`	The position adjustment to use for overlapping points on this layer
`keep.fraction`	numeric [0..1].
`keep.number`	integer number of labels to keep.
`h`	vector of bandwidths for x and y directions. Defaults to normal reference bandwidth (see bandwidth.nrd). A scalar value will be taken to apply to both directions.
`n`	Number of grid points in each direction. Can be scalar or a length-2 integer vector
`label.fill`	character.
`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

labels x at centre of range

See Also

kde2d used internally.

Examples

```
library(ggrepel)

random_string <- function(len = 6) {
paste(sample(letters, len, replace = TRUE), collapse = "")
}

# Make random data.
set.seed(1001)
d <- tibble::tibble(
  x = rnorm(100),
  y = rnorm(100),
  group = rep(c("A", "B"), c(50, 50)),
  lab = replicate(100, { random_string() })
)

ggplot(data = d, aes(x, y, label = lab)) +
  geom_point() +
  stat_dens2d_labels()

ggplot(data = d, aes(x, y, label = lab, color = group)) +
  geom_point() +
  stat_dens2d_labels()

ggplot(data = d, aes(x, y, label = lab, color = group)) +
  geom_point() +
  stat_dens2d_labels(geom = "text_repel")

ggplot(data = d, aes(x, y, label = lab, color = group)) +
  geom_point() +
  stat_dens2d_labels(geom = "text_repel", label.fill = NA)
```

---

stat_fit_augment *Augment data with fitted values and statistics*

Description

stat_fit_augment fits a model and returns the data augmented with information from the fitted model, using package 'broom'.

Usage

```
stat_fit_augment(mapping = NULL, data = NULL, geom = "smooth",
  method = "lm", method.args = list(formula = y ~ x),
  augment.args = list(), level = 0.95, y.out = ".fitted",
  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
method	character.
method.args	list of arguments to pass to method.
augment.args	list of arguments to pass to broom:augment.
level	numeric Level of confidence interval to use (0.95 by default)
y.out	character (or numeric) index to column to return as y.
position	The position adjustment to use for overlapping points on this layer
na.rm	logical 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

The output of augment is returned as is, except for y which is set based on y.out and y.observed which preserves the y returned by the broom::augment methods. This renaming is needed so that the geom works as expected.

Note

The statistics stat_fit_augment accepts only methods that accept formulas under any formal parameter name and a data argument. Use ggplot2::stat_smooth() instead of stat_fit_augment in production code if the additional features are not needed. At the moment stat_fit_augment is under development and may change.

`stat_fit_deviations` *Residuals from model fit as segments*

Description

`stat_fit_deviations` fits a linear model and returns fitted values and residuals ready to be plotted as segments.

Usage

```
stat_fit_deviations(mapping = NULL, data = NULL, geom = "segment",
  method = "lm", formula = NULL, 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
`method`	character Currently only "lm" is implemented.
`formula`	a "formula" object.
`position`	The position adjustment to use for overlapping points on this layer
`na.rm`	a logical 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 should not 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 can be used to automatically show residuals as segments in a plot of a fitted model equation. At the moment it supports only linear models fitted with function `lm()`. This stat only generates the residuals, the predicted values need to be separately added to the plot, so to make sure that the same model formula is used in all steps it is best to save the formula as an object and supply this object as argument to the different statistics.

Computed variables

Data frame with same nrow as data as subset for each group containing five numeric variables.

x1 x coordinates of observations

x2 x coordinates of fitted values

y1 y coordinates of observations

y2 y coordinates of fitted values

Note

For linear models x1 is equal to x2.

Examples

```
library(ggplot2)
# generate artificial data
set.seed(4321)
x <- 1:100
y <- (x + x^2 + x^3) + rnorm(length(x), mean = 0, sd = mean(x^3) / 4)
my.data <- data.frame(x, y, group = c("A", "B"), y2 = y * c(0.5,2))
# give a name to a formula
my.formula <- y ~ poly(x, 3, raw = TRUE)
# plot
ggplot(my.data, aes(x, y)) +
  geom_smooth(method = "lm", formula = my.formula) +
  stat_fit_deviations(formula = my.formula, color = "red") +
  geom_point()
```

---

stat_fit_glance *One row summary data frame for a fitted model*

Description

stat_fit_glance fits a model and returns a summary "glance" of the model's statistics, using package 'broom'.

Usage

```
stat_fit_glance(mapping = NULL, data = NULL, geom = "text_npc",
  method = "lm", method.args = list(formula = y ~ x),
  label.x = "left", label.y = "top", hstep = 0, vstep = NULL,
  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
method	character.
method.args	list of arguments to pass to method.
label.x, label.y	numeric with range 0..1 "normalized parent coordinates" (npc units) or character if using geom_text_npc() or geom_label_npc(). If using geom_text() or geom_label() numeric in native data units. If too short they will be recycled.
hstep, vstep	numeric in npc units, the horizontal and vertical step used between labels for different groups.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical 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

The output of glance is returned as is in the data object. If you do not know what names to expect for the variables returned, use broom::glance() and names() or print() to find out.

Warning!

The current implementation works only with methods that accept a formula as argument and which have a data paremeter through which a data frame can be passed. For example, lm() should be used with the formula interface, as the evaluation of x and y needs to be delayed until the internal object of the ggplot is available.

stat_fit_glance applies the function given by method separately to each group of observations, and factors mapped to aesthetics generate a separate group for each factor level. Because of this, it is not useful for annotating plots with results from t.test() or ANOVA or ANCOVA. In such cases use the stat_fit_tb() statistic which does the model fitting per panel.

Note

The names of the columns in the returned data are consitent with those returned by method glance() from package 'broom', that will frequently differ from the name of values returned by the print methods corresponding to fit or test function used. With some methods like cor.test() the data embedded in the "ggplot" object cannot be automatically passed as argument for the data parameter of the test or model fit function.

Examples

```
# Regression example
my.df <-
  data.frame(X = c(44.4, 45.9, 41.9, 53.3, 44.7, 44.1, 50.7, 45.2, 60.1),
             Y = c( 2.6,  3.1,  2.5,  5.0,  3.6,  4.0,  5.2,  2.8,  3.8))
# We need to check the names of the returned values!
broom::glance(lm(formula = Y ~ X, data = my.df ))
ggplot(my.df, aes(X, Y)) +
  geom_point() +
  stat_fit_glance(method = "lm",
                  method.args = list(formula = y ~ x),
                  aes(label = sprintf('r^2~"="~%.3f~~italic(P)~"="~%.2f',
                      stat(r.squared), stat(p.value))),
                  parse = TRUE)
```

---

stat_fit_residuals *Residuals from a model fit*

Description

`stat_fit_residuals` fits a linear model and returns residuals ready to be plotted as points.

Usage

```
stat_fit_residuals(mapping = NULL, data = NULL, geom = "point",
  method = "lm", formula = NULL, resid.type = NULL,
  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
`method`	character Currently only "lm" is implemented.
`formula`	a "formula" object.
`resid.type`	character passed to `residuals()` as argument for `type`.
`position`	The position adjustment to use for overlapping points on this layer
`na.rm`	a logical 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 should not 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 can be used to automatically plot residuals as points in a plot. At the moment it supports only linear models fitted with function lm(). This stat only generates the residuals.

Computed variables

Data frame with same nrow as data as subset for each group containing five numeric variables.

x1 x coordinates of observations

x2 x coordinates of fitted values

y1 y coordinates of observations

y2 y coordinates of fitted values

residuals residuals from the fit

Examples

```
library(ggplot2)
# generate artificial data
set.seed(4321)
x <- 1:100
y <- (x + x^2 + x^3) + rnorm(length(x), mean = 0, sd = mean(x^3) / 4)
my.data <- data.frame(x, y, group = c("A", "B"), y2 = y * c(0.5,2))
# give a name to a formula
my.formula <- y ~ poly(x, 3, raw = TRUE)
# plot
ggplot(my.data, aes(x, y)) +
  stat_fit_residuals(formula = my.formula, resid.type = "working")
```

---

stat_fit_tb *Model-fit summary or ANOVA*

Description

stat_fit_tb fits a model and returns a "tidy" version of the model's summary or ANOVA table, using package 'broom'. The annotation is added to the plots in tabular form.

Usage

```
stat_fit_tb(mapping = NULL, data = NULL, geom = "table_npc",
  method = "lm", method.args = list(formula = y ~ x),
  tb.type = "fit.summary", tb.vars = NULL, digits = 3,
  label.x = "center", label.y = "top", label.x.npc = NULL,
  label.y.npc = NULL, 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
method	character.
method.args	list of arguments to pass to method.
tb.type	character One of "fit.summary", "fit.anova" or "fit.coefs".
tb.vars	character vector, optionally named, used to select and or rename the columns of the table returned.
digits	integer indicating the number of significant digits to be used.
label.x, label.y	numeric Coordinates (in data units) to be used for absolute positioning of the output. If too short they will be recycled.
label.x.npc, label.y.npc	numeric with range 0..1 or character. Coordinates to be used for positioning the output, expressed in "normalized parent coordinates" or character string. If too short they will be recycled.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical 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

The output of tidy is returned as a single "cell" in a tibble (i.e. a tibble nested within a tibble). The returned data object contains a single, containing the result from a single model fit to all data in a panel. If grouping is present, it is ignored.

See Also

tidy for details on how the tidying of the resulst of model fits is done. See geom_table for details on how the formating and location of the table can be adjusted.

Examples

```
library(ggplot2)
# t-test example
x <- c(44.4, 45.9, 41.9, 53.3, 44.7, 44.1, 50.7, 45.2, 60.1)
group <- factor(c(rep("A", 4), rep("B", 5)))
my.df <- data.frame(x, group)

ggplot(my.df, aes(group, x)) +
  geom_point() +
  stat_fit_tb(method = "t.test",
              tb.vars = c("italic(t)" = "estimate", "italic(P)" = "p.value"),
              parse = TRUE)
```

---

stat_fit_tidy *One row data frame with fitted parameter estimates*

Description

stat_fit_tidy fits a model and returns a "tidy" version of the model's summary, using package 'broom'. To add the summary in tabular form use stat_fit_tb. When using stat_fit_tidy() you will most likely want to change the default mapping for label.

Usage

```
stat_fit_tidy(mapping = NULL, data = NULL, geom = "text_npc",
  method = "lm", method.args = list(formula = y ~ x),
  label.x = "left", label.y = "top", hstep = 0, vstep = NULL,
  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
method	character.
method.args	list of arguments to pass to method.

label.x, label.y
: numeric with range 0..1 or character. Coordinates to be used for positioning the output, expressed in "normalized parent coordinates" or character string. If too short they will be recycled.

hstep, vstep
: numeric in npc units, the horizontal and vertical step used between labels for different groups.

position
: The position adjustment to use for overlapping points on this layer

na.rm
: a logical 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

The output of tidy is returned after reshaping it into a single row. Grouping is respected, and the model fit separatately to each group of data. The returned data object has one row for each group within a panel.

Examples

```
# Regression example
my.df <-
  data.frame(X = c(44.4, 45.9, 41.9, 53.3, 44.7, 44.1, 50.7, 45.2, 60.1),
             Y = c( 2.6,  3.1,  2.5,  5.0,  3.6,  4.0,  5.2,  2.8,  3.8))

ggplot(my.df, aes(X, Y)) +
  geom_point() +
  stat_fit_tidy(method = "lm",
                method.args = list(formula = y ~ x),
                mapping = aes(label = sprintf("Slope = %.3g\np-value = %.3g",
                                              stat(x_estimate), stat(x_p.value))))
```

---

stat_fmt_tb *Select and slice a tibble nested in* data

Description

stat_partial_tb selects columns and/or remanes them and/or slices rows from a tible nested in data. This stat is designed to be used to pre-process tibble objects mapped to the label aesthetic before adding them to a plot with geom_table.

Usage

```
stat_fmt_tb(mapping = NULL, data = NULL, geom = "table",
  tb.vars = NULL, tb.rows = NULL, digits = 3,
  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
tb.vars	character vector, optionally named, used to select and or rename the columns of the table returned.
tb.rows	integer vector of row indexes of rows to be retained.
digits	integer indicating the number of significant digits to be retained in data.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical 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

The output of sequentially applying slice with tb.rows as argument and select with tb.vars to a list variable list mapped to label and containing a single tibble per row in data.

Examples

```
library(ggplot2)
my.df <- tibble::tibble(x = c(1, 2),
                        y = c(0, 4),
                        group = c("A", "B"),
                        tbs = list(a = tibble::tibble(X = 1:6, Y = rep(c("x", "y"), 3)),
                                   b = tibble::tibble(X = 1:3, Y = "x")))

ggplot(my.df, aes(x, y, label = tbs)) +
  stat_fmt_tb() +
  expand_limits(x = c(0,3), y = c(-2, 6))

ggplot(my.df, aes(x, y, label = tbs)) +
  stat_fmt_tb(tb.vars = c(value = "X", group = "Y"),
```

```
              tb.rows = 1:3) +
  expand_limits(x = c(0,3), y = c(-2, 6))
```

---

stat_peaks *Local maxima (peaks) or minima (valleys)*

Description

stat_peaks finds at which x positions local y maxima are located and stat_valleys finds at which x positions local y minima are located. Both stats return x and y numeric values for peaks or valleys and formatted character labels. The formatting is determined by a format string suitable for sprintf().

Usage

```
stat_peaks(mapping = NULL, data = NULL, geom = "point", span = 5,
  ignore_threshold = 0, strict = FALSE, label.fmt = "%.4g",
  x.label.fmt = NULL, y.label.fmt = label.fmt, position = "identity",
  na.rm = FALSE, show.legend = FALSE, inherit.aes = TRUE, ...)

stat_valleys(mapping = NULL, data = NULL, geom = "point", span = 5,
  ignore_threshold = 0, strict = FALSE, label.fmt = "%.4g",
  x.label.fmt = NULL, y.label.fmt = label.fmt, 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
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element. The default value is 5, meaning that a peak is bigger than two consecutive neighbors on each side. A NULL value for span is taken as a span covering the whole of the data range.
ignore_threshold	numeric value between 0.0 and 1.0 indicating the size threshold below which peaks will be ignored.
strict	logical flag: if TRUE, an element must be strictly greater than all other values in its window to be considered a peak. Default: FALSE.
label.fmt	character string giving a format definition for converting values into character strings by means of function sprintf.
x.label.fmt	character string giving a format definition for converting x-values into character strings by means of function sprintf or strftime.

y.label.fmt	character string giving a format definition for converting y-values into character strings by means of function sprintf.
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

These stats use geom_point by default as it is the geom most likely to work well in almost any situation without need of tweaking. The default aesthetics set by these stats allow their direct use with geom_text, geom_label, geom_line, geom_rug, geom_hline and geom_vline. The formatting of the labels returned can be controlled by the user.

Computed variables

x x-value at the peak (or valley) as numeric

y y-value at the peak (or valley) as numeric

x.label x-value at the peak (or valley) as character

y.label y-value at the peak (or valley) as character

Note

These stats check the scale of the x aesthetic and if is Datetime they correctly generate the labels by transforming the numeric x values to POSIXct objects, in which case the x.label.fmt must be suitable for strftime() rather than for sprintf(). These stats work nicely together with geoms geom_text_repel and geom_label_repel from package ggrepel to solve the problem of overlapping labels by displacing them. Alternatively, to discard overlapping labels use check_overlap = TRUE as argument to geom_text. By default the labels are character values suitable to be plotted as is, but with a suitable label.fmt labels suitable for parsing by the geoms (e.g. into expressions containing Greek letters, super- or subscripts, maths symbols or maths constructs) can be also easily obtained.

See Also

Other peaks and valleys functions: find_peaks

Examples

```
library(ggplot2)
lynx.df <- data.frame(year = as.numeric(time(lynx)), lynx = as.matrix(lynx))
ggplot(lynx.df, aes(year, lynx)) + geom_line() +
  stat_peaks(colour = "red") +
  stat_valleys(colour = "blue")
ggplot(lynx.df, aes(year, lynx)) + geom_line() +
  stat_peaks(colour = "red") +
  stat_peaks(colour = "red", geom = "rug")
```

`stat_poly_eq` *Equation, p-value, R^2, AIC or BIC of fitted polynomial*

Description

`stat_poly_eq` fits a polynomial and generates several labels including the equation and/or p-value, coefficient of determination (R^2), 'AIC' or 'BIC'.

Usage

```
stat_poly_eq(mapping = NULL, data = NULL, geom = "text_npc",
  position = "identity", ..., formula = NULL,
  eq.with.lhs = "italic(y)~`=`~", eq.x.rhs = NULL, coef.digits = 3,
  rr.digits = 2, label.x = "left", label.y = "top",
  label.x.npc = NULL, label.y.npc = NULL, hstep = 0, vstep = NULL,
  output.type = "expression", 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
`position`	The position adjustment to use for overlapping points on this layer
`...`	other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.
`formula`	a formula object.
`eq.with.lhs`	If `character` the string is pasted to the front of the equation label before parsing or a `logical` (see note).
`eq.x.rhs`	`character` this string will be used as replacement for `"x"` in the model equation when generating the label before parsing it.
`coef.digits, rr.digits`	integer Number of significant digits to use in for the vector of fitted coefficients and for R^2 labels.

`label.x, label.y`
: `numeric` with range 0..1 "normalized parent coordinates" (npc units) or character if using `geom_text_npc()` or `geom_label_npc()`. If using `geom_text()` or `geom_label()` numeric in native data units. If too short they will be recycled.

`label.x.npc, label.y.npc`
: `numeric` with range 0..1 (npc units) DEPRECATED, use label.x and label.y instead; together with a geom using npcx and npcy aesthetics.

`hstep, vstep`
: numeric in npc units, the horizontal and vertical step used between labels for different groups.

`output.type`
: character One of "expression", "LaTeX" or "text".

`na.rm`
: a logical 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`.

Details

This stat can be used to automatically annotate a plot with R^2, adjusted R^2 or the fitted model equation. It supports only linear models fitted with function `lm()`. The R^2 and adjusted R^2 annotations can be used with any linear model formula. The fitted equation label is correctly generated for polynomials or quasi-polynomials through the origin. Model formulas can use `poly()` or be defined algebraically with terms of powers of increasing magnitude with no missing intermediate terms, except possibly for the intercept indicated by "- 1" or "-1" in the formula. The validity of the `formula` is not checked in the current implementation, and for this reason the default aesthetics sets R^2 as label for the annotation. This stat only generates the label, the predicted values need to be separately added to the plot, so to make sure that the same model formula is used in all steps it is best to save the formula as an object and supply this object as argument to the different statistics.

Aesthetics

`stat_poly_eq` understands `x` and `y`, to be referenced in the `formula` and `weight` passed as argument to parameter `weights` of `lm()`. All three must be mappeed to `numeric` variables. In addition the aesthetics undertood by the geom used (`"text"` by default) are understood and grouping respected.

Computed variables

x x position for left edge

y y position near upper edge

eq.label equation for the fitted polynomial as a character string to be parsed

rr.label R^2 of the fitted model as a character string to be parsed

adj.rr.label Adjusted R^2 of the fitted model as a character string to be parsed

AIC.label AIC for the fitted model.

BIC.label BIC for the fitted model.

hjust Set to zero to override the default of the "text" geom.

Warning!

if using `output.type = "expression"`, then `parse = TRUE` is needed, while if using `output.type = "LaTeX"` `parse = FALSE`, the default of `geom_text` and `geom_label`, should be used.

Note

For backward compatibility a logical is accepted as argument for `eq.with.lhs`, giving the same output than the current default character value. By default "x" is retained as independent variable as this is the name of the aesthetic. However, it can be substituted by providing a suitable replacement character string through `eq.x.rhs`.

References

Written as an answer to a question at Stackoverflow. https://stackoverflow.com/questions/7549694/adding-regression-line-equation-and-r2-on-graph

Examples

```
library(ggplot2)
# generate artificial data
set.seed(4321)
x <- 1:100
y <- (x + x^2 + x^3) + rnorm(length(x), mean = 0, sd = mean(x^3) / 4)
my.data <- data.frame(x = x, y = y,
                      group = c("A", "B"),
                      y2 = y * c(0.5,2),
                      w = sqrt(x))
# give a name to a formula
formula <- y ~ poly(x, 3, raw = TRUE)
# no weights
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(formula = formula, parse = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(formula = formula, parse = TRUE,
               label.y = "bottom", label.x = "right")
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(formula = formula, parse = TRUE,
               label.y = 0.1, label.x = 0.9)
# using weights
ggplot(my.data, aes(x, y, weight = w)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(formula = formula, parse = TRUE)
# no weights, digits for R square
ggplot(my.data, aes(x, y)) +
```

```
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(formula = formula, rr.digits = 4, parse = TRUE)
# user specified label
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label =  paste(stat(eq.label), stat(adj.rr.label), sep = "~~~~")),
               formula = formula, parse = TRUE)
# user specified label and digits
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label =  paste(stat(eq.label), stat(adj.rr.label), sep = "~~~~")),
               formula = formula, rr.digits = 3, coef.digits = 2, parse = TRUE)
# geom = "text"
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(geom = "text", label.x = 100, label.y = 0, hjust = 1,
               formula = formula, parse = TRUE)
```

`stat_quadrant_counts` *Number of observations in quadrants*

Description

`stat_quadrant_counts()` counts the number of observations in each quadrant of a plot panel. By default it adds a text label to the far corner of each quadrant. It can also be used to obtain the total number of observations in each of two pairs of quadrants or in the whole panel. Grouping is ignored, so en every case a single count is computed for each quadrant in a plot panel.

Usage

```
stat_quadrant_counts(mapping = NULL, data = NULL, geom = "text_npc",
  position = "identity", quadrants = NULL, pool.along = "none",
  xintercept = 0, yintercept = 0, label.x = NULL, label.y = NULL,
  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
`position`	The position adjustment to use for overlapping points on this layer

`quadrants` integer vector indicating which quadrants are of interest, with a `0L` indicating the whole plot.

`pool.along` character, one of "none", "x" or "y", indicating which quadrants to pool to calculate counts by pair of quadrants.

`xintercept, yintercept` numeric the coordinates of the origin of the quadrants.

`label.x, label.y` `numeric` Coordinates (in npc units) to be used for absolute positioning of the labels.

`na.rm` a logical 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 should not 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 can be used to automatically count observations in each of the four quadrants of a plot, and by default add these counts as text labels.

Computed variables

Data frame with one to four rows, one for each quadrant for which observations are present in `data`.

quadrant integer, one of 0:4

x extreme x value in the quadrant

y extreme y value in the quadrant

count number of observations

Note

Values exactly equal to zero are counted as belonging to the positve quadrant. An argument value of zero, passed to formal parameter `quadrants` is interpreted as a request for the count of all observations in each plot panel. By default, which quadrants to compute counts for is decided based on which quadrants are expected to be visible in the plot. In the current implementation, the default positions of the labels is based on the range of the coordinates in a given panel. Consequently, when using facets even with free limits for x and y axes, the location of the labels will be consistent accross panels. This is achieved by use of `geom = "text_npc"` or `geom = "label_npc"`. To pass the positions in native data units, pass `geom = "text"` explicitly as argument, or labels will be positioned relative to the data range in each panel.

Examples

```
# generate artificial data
set.seed(4321)
x <- 1:100
y <- rnorm(length(x), mean = 10)
my.data <- data.frame(x, y)

ggplot(my.data, aes(x, y)) +
  geom_point() +
  stat_quadrant_counts()

ggplot(my.data, aes(x, y)) +
  geom_point() +
  stat_quadrant_counts(aes(label = sprintf("%i observations", stat(count)))) +
  expand_limits(y = 12.7)

ggplot(my.data, aes(x, y)) +
  geom_hline(yintercept = 10, colour = "blue") +
  geom_vline(xintercept = 50, colour = "blue") +
  geom_point() +
  stat_quadrant_counts(colour = "blue", xintercept = 50, yintercept = 10) +
  scale_y_continuous(expand = expand_scale(mult = 0.15, add = 0))

ggplot(my.data, aes(x, y)) +
  geom_hline(yintercept = 10, colour = "blue") +
  geom_point() +
  stat_quadrant_counts(colour = "blue", label.x = "right",
                       pool.along = "x", yintercept = 10) +
  expand_limits(y = c(7, 13))

ggplot(my.data, aes(x, y)) +
  geom_point() +
  stat_quadrant_counts(quadrants = 0, label.x = "left", label.y = "bottom")

ggplot(my.data, aes(x, y)) +
  geom_point() +
  stat_quadrant_counts(geom = "label_npc", quadrants = 0,
                       label.x = "left", label.y = "bottom")

ggplot(my.data, aes(x, y)) +
  geom_point() +
  stat_quadrant_counts(geom = "text") # use "text_npc" instead
```

---

stat_quadrat_counts *Renamed to stat_quadrant_counts*

Description

Function `stat_quadrant_counts()` was initially named `stat_quadrat_counts()` by mistake, and later its name corrected to `stat_quadrant_counts()`.

`try_data_frame` *Convert an R object into a tibble*

Description

This functions tries to convert any R object into a data.frame object. If `x` is already a data.frame, it is returned as is. If it is a list or a vector it is converted by means of `as.data.frame()`. If of any other type, a conversion into an object of class `xts` is attempted by means of `try.xts()` and if successful the `xts` object is converted into a data frame with a variable `time` containing times as `POSIXct` and the remaining data columns with the time series data. In this conversion row names are stripped.

Usage

```
try_data_frame(x, time.resolution = "month", as.numeric = FALSE,
  col.names = NULL)

try_tibble(x, time.resolution = "month", as.numeric = FALSE,
  col.names = NULL)
```

Arguments

`x`	An R object
`time.resolution`	character The time unit to which the returned time values will be rounded.
`as.numeric`	logical If TRUE convert time to numeric, expressed as fractional calendar years.
`col.names`	character vector

Value

A `tibble::tibble` object, derived from `data.frame`.

Warning!

The time zone was set to "UTC" by try.xts() in the test cases I used. Setting TZ to "UTC" can cause some trouble as several frequently used functions have as default the local or system TZ and will apply a conversion before printing or plotting time data, which in addition is affected by summer/winter time transitions. This should be taken into account as even for yearly data when conversion is to POSIXct a day (1st of January) will be set, but then shifted some hours if printed on a TZ different from "UTC". I recommend reading the documentation of package `lubridate-package` where the irregularities of time data and the difficulties they cause are very well described. In many cases when working with time series with yearly observations it is best to work with numeric values for years.

Note

This function can be used to easily convert time series data into a format that can be easily plotted with package `ggplot2`. `try_tibble` is another name for `try_data_frame` which tracks the separation and re-naming of `data_frame` into `tibble::tibble` in the imported packages.

Examples

```
library(xts)
class(lynx)
try_data_frame(lynx)
try_data_frame(lynx, "year")
class(austres)
try_data_frame(austres)
try_data_frame(austres, "quarter")
class(cars)
try_data_frame(cars)
```

Index