Package 'ggpmisc'

July 16, 2018

Type Package

Title Miscellaneous Extensions to 'ggplot2'

Version 0.3.0

Date 2018-07-16

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", adds tables to plots. Statistics: locate and tag peaks and valleys in 2D plots; 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 (>= 3.3.0), ggplot2 (>= 3.0.0)

Imports grid, rlang (>= 0.2.1), magrittr (>= 1.5), gridExtra (>= 2.3), tibble (>= 1.4.2), MASS (>= 7.3-50), polynom (>= 1.3-9), splus2R (>= 1.2-2), plyr (>= 1.8.4), dplyr (>= 0.7.6), xts (>= 0.10-2), zoo (>= 1.8-1), broom (>= 0.4.4), lubridate (>= 1.7.4)

Suggests knitr (>= 1.20), rmarkdown (>= 1.10), nlme (>= 3.1-137), ggrepel (>= 0.8.0)

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

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

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2018-07-16 13:40:03 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", adds tables to plots. Statistics: locate and tag peaks and valleys in 2D plots; 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' web site at http://ggplot2.org/
Package 'ggplot2' documentation at http://docs.ggplot2.org/
Package 'ggplot2' source code at https://github.com/hadley/ggplot2

See Also

Useful links:

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

Examples

```
library(ggplot2)

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_table` *Tables*

Description

`geom_table` adds a textual table directly to the ggplot using syntax similar to that of `geom_label`.

Usage

```
geom_table(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., parse = FALSE, check_overlap = FALSE,
  na.rm = FALSE, show.legend = NA, 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.
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.
check_overlap	If TRUE, text that overlaps previous text in the same layer will not be plotted.
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").

Note

This geom works only with tibbles as data, as it expects a list of data frames or a list of tibbles to be mapped to the label aesthetic. In the current version the following aesthetics affect the text within the table size, colour, and alpha. The argument to parameter parse is simply passed forward to gridExtra::ttheme_default(). As x and y determine the position of the whole 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 table as a whole. Other aesthetics, including `fill` are not yet implemented, neither are themes for table formatting. `annotate()` **cannot be used with** `geom = "table"`. Use `geom_table` directly also for adding 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-

See Also

function `tableGrob` as it is used to construct the table.

---

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.

---

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 = "null",
  method = "lm", method.args = list(formula = y ~ x),
  label.x.npc = "left", label.y.npc = "top", label.x = NULL,
  label.y = 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.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.
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.
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!

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 fit or test function used.

Examples

```
library(ggplot2)
# Correlation example
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)
my.df <- data.frame(x, y)
# estimated correlation is in column "estimate"!
broom::glance(cor.test(x, y))
ggplot(my.df, aes(x, y)) +
  geom_point() +
  stat_fit_glance(geom = "text",
                  method = "cor.test",
                  method.args = list(x = x, y = y, method = "spearman"),
                  aes(label = sprintf('r[s]~"="~%.2f~~italic(P)~"="~%.2f',
                      stat(estimate), 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'.

Usage

```
stat_fit_tb(mapping = NULL, data = NULL, geom = "table", method = "lm",
  method.args = list(formula = y ~ x), tb.type = "fit.summary",
  tb.vars = NULL, digits = 3, label.x.npc = "center",
  label.y.npc = "top", label.x = NULL, label.y = 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.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.
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.
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.

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'.

Usage

```
stat_fit_tidy(mapping = NULL, data = NULL, geom = "null", method = "lm",
  method.args = list(formula = y ~ x), label.x.npc = "left",
  label.y.npc = "top", label.x = NULL, label.y = 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.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.
`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.
`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.

---

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", formula = NULL,
  eq.with.lhs = "italic(y)~`=`~", eq.x.rhs = NULL, coef.digits = 3,
  rr.digits = 2, label.x.npc = "left", label.y.npc = "top",
  label.x = NULL, label.y = NULL, output.type = "expression",
  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
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.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.
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.
output.type	character One of "expression", "LaTeX" or "text".
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.

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)
# 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)
```

`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",
  position = "identity", quadrants = NULL, pool.along = "none",
  origin.x = 0, origin.y = 0, labels.range.x = NULL,
  labels.range.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.
origin.x, origin.y	numeric the coordinates of the origin of the quadrants.
labels.range.x, labels.range.y	numeric Coordinates (in data 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 data ploted in a given panel. Consequently, when using facets unless using free limits for x and y axes, the location of the labels will need supplied by the user when consistent placement accross panels is desired.

Examples

```
library(ggplot2)
# 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 - 50, y - 10)) +
  geom_hline(yintercept = 0, colour = "blue") +
  geom_vline(xintercept = 0, colour = "blue") +
  geom_point() +
  stat_quadrant_counts(colour = "blue")

ggplot(my.data, aes(x - 50, y - 10)) +
  geom_hline(yintercept = 0, colour = "blue") +
  geom_point() +
  stat_quadrant_counts(colour = "blue", pool.along = "x")

ggplot(my.data, aes(x - 50, y - 10)) +
  geom_vline(xintercept = 0, colour = "blue") +
  geom_point() +
  stat_quadrant_counts(colour = "blue", pool.along = "y")

ggplot(my.data, aes(x - 50, y - 10)) +
  geom_point() +
  stat_quadrant_counts(quadrants = 0)
```

`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