

# Package ‘ggpmisc’

May 14, 2017

**Type** Package

**Title** Miscellaneous Extensions to 'ggplot2'

**Version** 0.2.15

**Date** 2017-05-14

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

**Description** Extensions to 'ggplot2' respecting the grammar of graphics paradigm. Provides new statistics to locate and tag peaks and valleys in 2D plots, a statistics to add a label with the equation of a polynomial fitted with `lm()`, or  $R^2$  or adjusted  $R^2$  or information criteria for any model fitted with function `lm()`. Additional statistics give access to functions in package 'broom'. Provides a function for flexibly converting time series to data frames suitable for plotting with `ggplot()`. In addition provides statistics and `ggplot` geometries useful for diagnosing what data are passed to `compute_group()` and `compute_panel()` functions and to geometries.

**License** GPL (>= 2)

**LazyData** TRUE

**LazyLoad** TRUE

**ByteCompile** TRUE

**Depends** R (>= 3.2.0), ggplot2 (>= 2.2.1)

**Imports** tibble (>= 1.2), MASS (>= 7.3-45), polynom (>= 1.3-8), spls2R (>= 1.2-2), plyr (>= 1.8.4), dplyr (>= 0.5.0), xts (>= 0.9-7), zoo (>= 1.7-13), broom (>= 0.4.1), lubridate (>= 1.6.0)

**Suggests** knitr (>= 1.14), rmarkdown (>= 1.1), nlme (>= 3.1-128), ggrepel (>= 0.6.3)

**URL** <http://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],  
Kamil Slowikowski [ctb]  
**Repository** CRAN  
**Date/Publication** 2017-05-14 17:29:59 UTC

**R topics documented:**

ggpmisc-package . . . . .	2
geom_debug . . . . .	4
geom_null . . . . .	5
ggplot . . . . .	6
stat_debug_group . . . . .	7
stat_debug_panel . . . . .	9
stat_dens2d_filter . . . . .	10
stat_dens2d_labels . . . . .	12
stat_fit_augment . . . . .	14
stat_fit_deviations . . . . .	15
stat_fit_glance . . . . .	17
stat_fit_residuals . . . . .	18
stat_fit_tidy . . . . .	19
stat_peaks . . . . .	20
stat_poly_eq . . . . .	22
try_data_frame . . . . .	25
<b>Index</b>	<b>27</b>

---

ggpmisc-package	<i>ggpmisc: Miscellaneous Extensions to 'ggplot2'</i>
-----------------	---

---

**Description**

Extensions to 'ggplot2' respecting the grammar of graphics paradigm. Provides new statistics to locate and tag peaks and valleys in 2D plots, a statistics to add a label with the equation of a polynomial fitted with lm(), or R^2 or adjusted R^2 or information criteria for any model fitted with function lm(). Additional statistics give access to functions in package 'broom'. Provides a function for flexibly converting time series to data frames suitable for plotting with ggplot(). In addition provides statistics and ggplot geometries useful for diagnosing what data are passed to compute\_group() and compute\_panel() functions and to geometries.

**Details**

The new facilities for cleanly defining new stats and geoms added to 'ggplot2' in version 2.0.0 have made this package easy to code. However, this means that this package requires version 2.0.0 or later of ggplot2.

Extensions provided:

- Function for conversion of time series data into data frames that can be plotted with ggplot.
- 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.
- "Debug" stats and a "debug" geom that print to the console a summary of their data input.

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.

### 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)

Pedro J. Aphalo

### References

Package suite 'r4photobiology' web site at <http://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:

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

geom\_debug

*Geom which prints input data to console***Description**

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

**Usage**

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

**Arguments**

mapping	Set of aesthetic mappings created by <a href="#">aes</a> or <a href="#">aes_</a> . If specified and <code>inherit.aes = TRUE</code> (the default), is combined with the default mapping at the top level of the plot. You only need to supply mapping if there isn't a mapping defined for the plot.
data	A data frame. If specified, overrides the default data frame defined at the top level of the plot.
stat	The statistical transformation to use on the data for this layer, as a string.
summary.fun	A function used to print the data object received as input.
summary.fun.args	A list of additional arguments to be passed to <code>summary.fun</code> .
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . There are three types of arguments you can use here: <ul style="list-style-type: none"> <li>• Aesthetics: to set an aesthetic to a fixed value, like <code>color = "red"</code> or <code>size = 3</code>.</li> <li>• Other arguments to the layer, for example you override the default stat associated with the layer.</li> <li>• Other arguments passed on to the stat.</li> </ul>

## Details

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

## Note

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

---

geom_null	<i>A null geom or 'non-op' geom.</i>
-----------	--------------------------------------

---

## Description

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

## Usage

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

## Arguments

mapping	Set of aesthetic mappings created by <a href="#">aes</a> or <a href="#">aes_</a> . If specified and <code>inherit.aes = TRUE</code> (the default), is combined with the default mapping at the top level of the plot. You only need to supply mapping if there isn't a mapping defined for the plot.
data	A data frame. If specified, overrides the default data frame defined at the top level of the plot.
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . There are three types of arguments you can use here: <ul style="list-style-type: none"> <li>• Aesthetics: to set an aesthetic to a fixed value, like <code>color = "red"</code> or <code>size = 3</code>.</li> </ul>

- Other arguments to the layer, for example you override the default `stat` associated with the layer.
- Other arguments passed on to the `stat`.

### Note

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

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

---

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

<code>data</code>	Default spectrum dataset to use for plot. If not a spectrum, the methods used will be those defined in package <code>ggplot2</code> . See <a href="#">ggplot</a> . If not specified, must be supplied in each layer added to the plot.
<code>mapping</code>	Default list of aesthetic mappings to use for plot. If not specified, in the case of spectral objects, a default mapping will be used.
<code>...</code>	Other arguments passed on to methods. Not currently used.
<code>time.resolution</code>	character The time unit to which the returned time values will be rounded.
<code>as.numeric</code>	logical If TRUE convert time to numeric, expressed as fractional calendar years.
<code>environment</code>	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 <code>ggplot()</code> 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()
```

---

stat\_debug\_group

---

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


---

## Description

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

## Usage

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

## Arguments

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

## Computed variables

**x** x at centre of range  
**y** y at centre of range  
**nrow** `nrow()` of data object  
**ncol** `ncol()` of data object  
**colnames** `colnames()` of data object  
**colclasses** `class()` of x and y columns in data object  
**group** all distinct values in group as passed in data object  
**PANEL** all distinct values in PANEL as passed in data object

## See Also

Other diagnosis functions: [stat\\_debug\\_panel](#)

## Examples

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



---

stat_debug_panel	<i>Print to console data received by the compute panel function.</i>
------------------	--

---

## Description

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

## Usage

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

## Arguments

mapping	The aesthetic mapping, usually constructed with <a href="#">aes</a> or <a href="#">aes_string</a> . Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
summary.fun	A function used to print the data object received as input.
summary.fun.args	A list.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . This can include aesthetics whose values you want to set, not map. See <a href="#">layer</a> for more details.

## Computed variables

**x** x at centre of range  
**y** y at centre of range  
**nrow** nrow() of data object  
**ncol** ncol() of data object  
**colnames** colnames() of data object

**colclasses** class() of x and y columns in data object  
**group** all distinct values in group as passed in data object  
**PANEL** all distinct values in PANEL as passed in data object

### See Also

Other diagnosis functions: [stat\\_debug\\_group](#)

### Examples

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

---

stat_dens2d_filter	<i>Filter observations in high density regions.</i>
--------------------	---

---

### 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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. <a href="#">borders</a> .
h	vector of bandwidths for x and y directions. Defaults to normal reference bandwidth (see <a href="#">bandwidth.nrd</a> ). 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 <a href="#">layer</a> . This can include aesthetics whose values you want to set, not map. See <a href="#">layer</a> 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	<i>Reset labels of observations in high density regions.</i>
--------------------	--

---

## 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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 <code>bandwidth.nrd</code> ). 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. <code>borders</code> .
...	other arguments passed on to <code>layer</code> . This can include aesthetics whose values you want to set, not map. See <code>layer</code> 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	<i>Return the data augmented with fitted values and statistics.</i>
------------------	---

---

## 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . This can include aesthetics whose values you want to set, not map. See <a href="#">layer</a> 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.

### Note

The statistics `stat_fit_augment` and `stat_fit_augment_panel` at the moment 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.

---

<code>stat_fit_deviations</code>	<i>Plot residuals from fit as segments.</i>
----------------------------------	---

---

### 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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	<i>Return one row summary data frame for a fitted model.</i>
-----------------	--

---

## 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . This can include aesthetics whose values you want to set, not map. See <a href="#">layer</a> for more details.

## Computed variables

The output of [glance](#) is returned as is.

---

stat_fit_residuals	<i>Plot residuals from a model fit.</i>
--------------------	---

---

## 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . This can include aesthetics whose values you want to set, not map. See <a href="#">layer</a> 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_residuais(formula = my.formula, resid.type = "working")
```

---

stat\_fit\_tidy

---

Return 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . This can include aesthetics whose values you want to set, not map. See <a href="#">layer</a> for more details.

### Computed variables

The output of [tidy](#) is returned after reshaping it into a single row.

---

stat_peaks	<i>Find and label local maxima (peaks) or minima (valleys).</i>
------------	---

---

### 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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 <a href="#">sprintf</a> .
x.label.fmt	character string giving a format definition for converting \$x\$-values into character strings by means of function <a href="#">sprintf</a> or <a href="#">strftime</a> .
y.label.fmt	character string giving a format definition for converting \$y\$-values into character strings by means of function <a href="#">sprintf</a> .
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. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . This can include aesthetics whose values you want to set, not map. See <a href="#">layer</a> 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

---

Add a label for a fitted linear model to a plot.

---

**Description**

`stat_poly_eq` fits a polynomial and generates several labels with an equation and/or 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 <a href="#">aes</a> or <a href="#">aes_string</a> . 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. <a href="#">borders</a> .
...	other arguments passed on to <a href="#">layer</a> . This can include aesthetics whose values you want to set, not map. See <a href="#">layer</a> 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.

### 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.

### 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`.

### 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
formula <- y ~ poly(x, 3, raw = TRUE)
# plot
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(formula = formula, parse = TRUE)
# plot
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(formula = formula, rr.digits = 4, parse = TRUE)
# plot
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = paste(..eq.label.., ..adj.rr.label.., sep = "~~~~")),
               formula = formula, parse = TRUE)
# plot
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = paste(..eq.label.., ..adj.rr.label.., sep = "~~~~")),
               formula = formula, rr.digits = 3, coef.digits = 2, parse = TRUE)
```



---

try_data_frame	<i>Convert an R object containing observations into a tibble.</i>
----------------	---

---

## 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

<code>x</code>	An R object
<code>time.resolution</code>	character The time unit to which the returned time values will be rounded.
<code>as.numeric</code>	logical If TRUE convert time to numeric, expressed as fractional calendar years.
<code>col.names</code>	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](#) 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

`aes`, [4](#), [5](#), [8–10](#), [12](#), [14](#), [15](#), [17–19](#), [21](#), [23](#)  
`aes_`, [4](#), [5](#)  
`aes_string`, [8–10](#), [12](#), [14](#), [15](#), [17–19](#), [21](#), [23](#)  
`augment`, [15](#)

`borders`, [4](#), [5](#), [8](#), [9](#), [11](#), [13](#), [15–18](#), [20](#), [21](#), [23](#)

`find_peaks`, [22](#)

`geom_debug`, [4](#)  
`geom_label_repel`, [22](#)  
`geom_null`, [5](#)  
`geom_text_repel`, [22](#)  
`ggplot`, [6](#), [6](#)  
`ggpmisc` (ggpmisc-package), [2](#)  
`ggpmisc-package`, [2](#)  
`ggrepel`, [22](#)  
`glance`, [17](#)

`kde2d`, [11](#), [13](#)

`layer`, [4](#), [5](#), [8](#), [9](#), [11](#), [13](#), [15–18](#), [20](#), [21](#), [23](#)  
`lubridate`, [25](#)

`sprintf`, [21](#)  
`stat_debug_group`, [7](#), [10](#)  
`stat_debug_panel`, [8](#), [9](#)  
`stat_dens2d_filter`, [10](#)  
`stat_dens2d_filter_g`  
    (`stat_dens2d_filter`), [10](#)  
`stat_dens2d_labels`, [12](#)  
`stat_fit_augment`, [14](#)  
`stat_fit_deviations`, [15](#)  
`stat_fit_glance`, [17](#)  
`stat_fit_residuals`, [18](#)  
`stat_fit_tidy`, [19](#)  
`stat_peaks`, [20](#)  
`stat_poly_eq`, [22](#)  
`stat_valleys` (`stat_peaks`), [20](#)  
`strftime`, [21](#)

`tidy`, [20](#)  
`try_data_frame`, [25](#)  
`try_tibble` (`try_data_frame`), [25](#)