R topics documented:

as_ggsideCoord ...................................................... 2
geom_xsidebar ....................................................... 3
geom_xsideboxplot .................................................. 6
geom_xsidedensity ................................................... 9
geom_xsidefreqpoly ................................................. 11
geom_xsidefunction ............................................... 13
geom_xsidehistogram .............................................. 15
geom_xsidelabel ..................................................... 17
geom_xsideline ....................................................... 19
geom_xsidepoint ..................................................... 21
gem.xsidesection .................................................... 23
geom_xsidetext ....................................................... 25
geom_xsidetile ....................................................... 27
geom_xsideviolin ................................................... 29
ggside ................................................................. 32
ggside-ggproto-facets .............................................. 33
ggside-scales-continuous ........................................ 34
ggside-scales-discrete ............................................ 36
is.ggside ............................................................. 37
position_rescale ..................................................... 37
scale_xcolour ......................................................... 39
scale_xfill ........................................................... 39
scale_ycolour_hue .................................................. 40
scale_yfill_hue ....................................................... 40
stat_summarise ....................................................... 41
theme_ggside_grey .................................................. 42
use_xside_aes ....................................................... 45
xside ................................................................. 46
yside ................................................................. 47

Index ................................................................. 48

------
as_ggsideCoord  Coord Compatible with ggside

Description

S3 class that converts old Coord into one that is compatible with ggside. Can also update ggside on
the object. Typically, the new ggproto will inherit from the object being replaced.
**Usage**

```r
as_ggsideCoord(coord)
```

## Default S3 method:
```
as_ggsideCoord(coord)
```

## S3 method for class `CoordCartesian`
```
as_ggsideCoord(coord)
```

## S3 method for class `CoordSide`
```
as_ggsideCoord(coord)
```

## S3 method for class `CoordTrans`
```
as_ggsideCoord(coord)
```

## S3 method for class `CoordFixed`
```
as_ggsideCoord(coord)
```

**Arguments**

- `coord` : coord ggproto Object to replace

---

**Description**

The `xside` and `yside` variants of `geom_bar` is `geom_xsidebar` and `geom_ysidebar`. These variants both inherit from `geom_bar` and only differ on where they plot data relative to main panels.

The `xside` and `yside` variants of `geom_col` is `geom_xsidecol` and `geom_ysidecol`. These variants both inherit from `geom_col` and only differ on where they plot data relative to main panels.

**Usage**

```r
geom_xsidebar(
  mapping = NULL,
  data = NULL,
  stat = "count",
  position = "stack",
  ...,
  width = NULL,
  na.rm = FALSE,
  orientation = "x",
  show.legend = NA,
  inherit.aes = TRUE
)
```
geom_ysidebar(
  mapping = NULL,
  data = NULL,
  stat = "count",
  position = "stack",
  ...,
  width = NULL,
  na.rm = FALSE,
  orientation = "y",
  show.legend = NA,
  inherit.aes = TRUE
)

geom_xsidecol(
  mapping = NULL,
  data = NULL,
  position = "stack",
  ...,
  width = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_ysidecol(
  mapping = NULL,
  data = NULL,
  position = "stack",
  ...,
  width = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  orientation = "y"
)

Arguments

  mapping  Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

  data     The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(x, 10)`).

**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()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

**width**  
Bar width. By default, set to 90% of the resolution of the data.

**na.rm**  
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

**orientation**  
The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the Orientation section for more detail.

**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. It can also be a named logical vector to finely select the aesthetics to display.

**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()`.

**Value**  
XLayer or YLayer object to be added to a ggplot object

**Aesthetics**  
Required aesthetics are in bold.

- **x**
- **y**
- **fill or xfill** Fill color of the xsidebar
- **fill or yfill** Fill color of the ysidebar
- **width** specifies the width of each bar
- **height** specifies the height of each bar
- **alpha** Transparency level of xfill or yfill
- **size** size of the border line.

**See Also**

`geom_xsidehistogram`, `geom_ysidehistogram`
Examples

```r
p <- ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species, fill = Species)) +
  geom_point()

# sidebar - uses StatCount
p +
  geom_xsidebar() +
  geom_ysidebar()

# sidecol - uses Global mapping
p +
  geom_xsidecol() +
  geom_ysidecol()
```

---

**geom_xsideboxplot**  
*Side boxplots*

**Description**

The `xside` and `yside` variants of `geom_boxplot` is `geom_xsideboxplot` and `geom_ysideboxplot`.

**Usage**

```r
geom_xsideboxplot(
  mapping = NULL,
  data = NULL,
  stat = "boxplot",
  position = "dodge2",
  ...,
  outlier.colour = NULL,
  outlier.color = NULL,
  outlier.fill = NULL,
  outlier.shape = 19,
  outlier.size = 1.5,
  outlier.stroke = 0.5,
  outlier.alpha = NULL,
  notch = FALSE,
  notchwidth = 0.5,
  varwidth = FALSE,
  na.rm = FALSE,
  orientation = "x",
  show.legend = NA,
  inherit.aes = TRUE
)
```

```r
geom_ysideboxplot(
```
Arguments

mapping Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
   If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
   A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
   A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

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()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

outlier.colour, outlier.color, outlier.fill, outlier.shape, outlier.size, outlier.stroke, outlier.alpha Default aesthetics for outliers. Set to `NULL` to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting `outlier.shape = NA`. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.

**notch**
If FALSE (default) make a standard box plot. If TRUE, make a notched box plot. Notches are used to compare groups; if the notches of two boxes do not overlap, this suggests that the medians are significantly different.

**notchwidth**
For a notched box plot, width of the notch relative to the body (defaults to `notchwidth = 0.5`).

**varwidth**
If FALSE (default) make a standard box plot. If TRUE, boxes are drawn with widths proportional to the square-roots of the number of observations in the groups (possibly weighted, using the `weight` aesthetic).

**na.rm**
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

**orientation**
The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting `orientation` to either "x" or "y". See the Orientation section for more detail.

**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. It can also be a named logical vector to finely select the aesthetics to display.

**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()`.

**Value**

XLayer or YLayer object to be added to a ggplot object

**See Also**

`geom_*sideviolin`

**Examples**

```r
df <- expand.grid(UpperCase = LETTERS, LowerCase = letters)
df$Combo_Index <- as.integer(df$UpperCase)*as.integer(df$LowerCase)

p1 <- ggplot(df, aes(UpperCase, LowerCase)) +
geom_tile(aes(fill = Combo_Index))
```

#sideboxplots
#Note - Mixing discrete and continuous axis scales
#using xsideboxplots when the y aesthetic was previously
#mapped with a continuous variable will prevent
#any labels from being plotted. This is a feature that
# will hopefully be added to ggside in the future.

p1 + geom_xsideboxplot(aes(y = Combo_Index)) +  
geom_ysideboxplot(aes(x = Combo_Index))

# sideboxplots with swapped orientation
# Note - Discrete before Continuous
# If you are to mix Discrete and Continuous variables on
# one axis, ggplot2 prefers the discrete variable to be mapped
# BEFORE the continuous.

ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) +  
  geom_xsideboxplot(aes(y = Species), orientation = "y") +  
  geom_point()

# Alternatively, you can recast discrete as a factor and then
# a numeric

ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) +  
  geom_point() +  
  geom_xsideboxplot(aes(y = as.numeric(Species)), orientation = "y") +  
  geom_ysideboxplot(aes(x = as.numeric(Species)), orientation = "x")

---

**geom_xsideboxplot**

**Side density distributions**

### Description

The `xside` and `yside` variants of `geom_density` is `geom_xsideboxplot` and `geom_ysideboxplot`.

### Usage

```r
geom_xsideboxplot(
  mapping = NULL,
  data = NULL,
  stat = "density",
  position = "identity",
  ...,  
  na.rm = FALSE,
  orientation = "x",
  show.legend = NA,
  inherit.aes = TRUE,
  outline.type = "upper"
)
```

```r
geom_ysideboxplot(
  mapping = NULL,
  data = NULL,
  stat = "density",
  position = "identity",
  ...,  
  na.rm = FALSE,
  orientation = "y",
  show.legend = NA,
  inherit.aes = TRUE,
  outline.type = "upper"
)
```
..., 
na.rm = FALSE, 
orientation = "y", 
show.legend = NA, 
inherit.aes = TRUE, 
outline.type = "upper"
)

Arguments

mapping Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes(...)}. If specified and \texttt{inherit.aes = TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data The data to be displayed in this layer. There are three options:

If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}. A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x, 10)}).

stat Use to override the default connection between \texttt{geom_density()} and \texttt{stat_density()}.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red"} or \texttt{size = 3}. They may also be parameters to the paired geom/stat.

na.rm If \texttt{FALSE}, the default, missing values are removed with a warning. If \texttt{TRUE}, missing values are silently removed.

orientation The orientation of the layer. The default (\texttt{NA}) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting \texttt{orientation} to either \texttt{"x"} or \texttt{"y"}. See the \texttt{Orientation} section for more detail.

show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If \texttt{FALSE}, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders()}.

outline.type Type of the outline of the area; "both" draws both the upper and lower lines, "upper"/"lower" draws the respective lines only. "full" draws a closed polygon around the area.

Value

XLayer or YLayer object to be added to a ggplot object
Examples

```r
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point(size = 2) +
  geom_xsidefreqpoly() +
  geom_ysidefreqpoly() +
  theme(axis.text.x = element_text(angle = 90, vjust = .5))
```

```r
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point(size = 2) +
  geom_xsidefreqpoly(aes(y = after_stat(count)), position = "identity") +
  geom_ysidefreqpoly(aes(x = after_stat(scaled))) +
  theme(axis.text.x = element_text(angle = 90, vjust = .5))
```

---

**geom_xsidefreqpoly**  
*Side Frequency Polygons*

**Description**

The `xside` and `yside` variants of `geom_freqpoly` is `geom_xsidefreqpoly` and `geom_ysidefreqpoly`.

**Usage**

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

```r
geom_ysidefreqpoly(
  mapping = NULL,
  data = NULL,
  stat = "bin",
  position = "identity",
  ...
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```
Arguments

**mapping**
Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes` = `TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

**data**
The data to be displayed in this layer. There are three options:
If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a `formula` (e.g. `~ head(.x, 10)`).

**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()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

**na.rm**
If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

**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. It can also be a named logical vector to finely select the aesthetics to display.

**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()`.

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```r
ggplot(diamonds, aes(price, carat, colour = cut)) + geom_point() + geom_xsidefreqpoly(aes(y=after_stat(count)), binwidth = 500) + geom_ysidefreqpoly(aes(x=after_stat(count)), binwidth = .2)
```
Description

The `xside` and `yside` variants of `geom_function`.

Usage

gem_xsidefunction(
mapping = NULL,
data = NULL,
stat = "function",
position = "identity",
..., 
na.rm = FALSE,
show.legend = NA, 
inherit.aes = TRUE

)

stat_xsidefunction(
mapping = NULL,
data = NULL,
geom = "xsidefunction",
position = "identity",
..., 
fun, 
xlim = NULL,
n = 101, 
args = list(), 
na.rm = FALSE, 
show.legend = NA, 
inherit.aes = TRUE

)

gem_ysidefunction(
mapping = NULL,
data = NULL,
stat = "ysidefunction",
position = "identity",
..., 
na.rm = FALSE,
show.legend = NA, 
inherit.aes = TRUE

)

stat_ysidefunction(}
mapping = NULL,
data = NULL,
geom = "ysidefunction",
position = "identity",
..., 
fun, 
ylim = NULL, 
n = 101, 
args = list(), 
na.rm = FALSE, 
show.legend = NA, 
inherit.aes = TRUE
}

Arguments

mapping Set of aesthetic mappings created by aes() or aes_. If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data Ignored by stat_function(), do not use.
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(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
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. It can also be a named logical vector to finely select the aesthetics to display.
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().
geom The geometric object to use display the data
fun Function to use. Either 1) an anonymous function in the base or rlang formula syntax (see rlang::as_function()) or 2) a quoted or character name referencing a function; see examples. Must be vectorised.
xlim Optionally, restrict the range of the function to this range.
n Number of points to interpolate along the x axis.
args List of additional arguments passed on to the function defined by fun.
ylim Optionally, restrict the range of the function to this range (y-axis)

Value

XLayer or YLayer object to be added to a ggplot object
Examples

```r
x <- rweibull(100, 2.6, 3)
y <- rweibull(100, 1.8, 3)
xy.df <- data.frame(cbind(x, y))
p <- ggplot(xy.df, aes(x, y)) +
    geom_point(colour = "blue", size = 0.25) +
    geom_density2d() +
    geom_xside density(fill = "blue", alpha = .3) +
    geom_ysidedensity(fill = "blue", alpha = .3) +
    stat_xsidefunction(fun = dweibull, args = list(shape = 1.8, scale = 3), colour = "red") +
    stat_ysidefunction(fun = dweibull, args = list(shape = 2.6, scale = 3), colour = "red") +
    theme_classic()
p
```

---

**geom_xsidehistogram**  
Side Histograms

**Description**

The `xside` and `yside` variants of `geom_histogram` is `geom_xsidehistogram` and `geom_ysidehistogram`. These variants both inherit from `geom_histogram` and only differ on where they plot data relative to main panels.

**Usage**

```r
geom_xsidehistogram(
    mapping = NULL,
    data = NULL,
    stat = "bin",
    position = "stack",
    ...,
    binwidth = NULL,
    bins = NULL,
    na.rm = FALSE,
    orientation = "x",
    show.legend = NA,
    inherit.aes = TRUE
)
```

```r
geom_ysidehistogram(
    mapping = NULL,
    data = NULL,
    stat = "bin",
    position = "stack",
    ...,
    binwidth = NULL,
    bins = NULL,
```
Arguments

mapping  Set of aesthetic mappings created by aes() or aes_( ). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

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(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

binwidth The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data.
The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.

bins Number of bins. Overridden by binwidth. Defaults to 30.

na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

orientation The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the Orientation section for more detail.

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. It can also be a named logical vector to finely select the aesthetics to display.
**geom_xsidelabel**

- **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()`.

**Value**

XLayer or YLayer object to be added to a ggplot object

**Aesthetics**

`geom_*sidehistogram` uses the same aesthetics as `geom_*sidebar()`

**Examples**

```r
p <- ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species, fill = Species)) + geom_point()

# sidehistogram
p + geom_xsidehistogram(binwidth = 0.1) + geom_ysidehistogram(binwidth = 0.1)
p + geom_xsidehistogram(aes(y = after_stat(density)), binwidth = 0.1) + geom_ysidehistogram(aes(x = after_stat(density)), binwidth = 0.1)
```

---

**geom_xsidelabel** *Side label*

**Description**

The `xside` and `yside` variants of `geom_label`.

**Usage**

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

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

Arguments

mapping
Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()}. If specified and \texttt{inherit.aes} = \texttt{TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data
The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}.
A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x, 10)}).

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. Cannot be jointly specified with \texttt{nudge_x} or \texttt{nudge_y}.

parse
If \texttt{TRUE}, the labels will be parsed into expressions and displayed as described in \texttt{?plotmath}.

nudge_x, nudge_y
Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.
geom_xsideline

label.padding  Amount of padding around label. Defaults to 0.25 lines.
label.r          Radius of rounded corners. Defaults to 0.15 lines.
label.size       Size of label border, in mm.
na.rm            If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
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. It can also be a named logical vector to finely select the aesthetics to display.
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().

Value

XLayer or YLayer object to be added to a ggplot object

Description

The xsid and ysid of geom_line. The xsid and ysid variants of geom_path

Usage

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

geom_ysideline(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
Arguments

mapping Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()}. If specified and \texttt{inherit.aes = TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}.
A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x, 10)}).

stat The statistical transformation to use on the data for this layer, as a string.
geom_xsidepoint

position  Position adjustment, either as a string, or the result of a call to a position adjustment function.

na.rm  If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

orientation  The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the Orientation section for more detail.

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. It can also be a named logical vector to finely select the aesthetics to display.

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(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

lineend  Line end style (round, butt, square).

linejoin  Line join style (round, mitre, bevel).

linemitre  Line mitre limit (number greater than 1).

arrow  Arrow specification, as created by grid::arrow().

Value

XLayer or YLayer object to be added to a ggplot object

Examples

#sideline
ggplot(economics, aes(date, pop)) +
  geom_xsideline(aes(y = unemploy)) +
  geom_col()
Usage

```r
geom_xsidepoint(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...
)
```

```r
geom_ysidepoint(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...
)
```

Arguments

- **mapping**: Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

- **data**: The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a `data.frame`. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

- **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, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
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. It can also be a named logical vector to finely select the aesthetics to display.

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().

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```r
ggplot(diamonds, aes(depth, table, alpha = .2)) +
  geom_point() +
  geom_ysidepoint(aes(x = price)) +
  geom_xsidepoint(aes(y = price)) +
  theme(
    ggside.panel.scale = .3
  )
```

Description

The xside and yside of geom_segment.

Usage

```r
geom_xsidepoint(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

```
geom_ysidepoint(
  mapping = NULL,
  data = NULL,
```
stat = "identity",
position = "identity",
..., 
arrow = NULL,
arrow.fill = NULL,
lineend = "butt",
linejoin = "round",
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE
)

Arguments

mapping Set of aesthetic mappings created by aes() or aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
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(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
arrow specification for arrow heads, as created by arrow().
asarrow.fill fill colour to use for the arrow head (if closed). NULL means use colour aesthetic.
lineend Line end style (round, butt, square).
linejoin Line join style (round, mitre, bevel).
na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
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. It can also be a named logical vector to finely select the aesthetics to display.
inheritaes 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().
Value

XLayer or YLayer object to be added to a ggplot object

Examples

```r
library(dplyr)
library(tidyr)
library(ggdendro)
#dendrogram with geom_xsidesegment
df0 <- mutate(diamonds,
   colclar = interaction(color, clarity,
                        sep = "_", drop = TRUE))
df1 <- df0 %>%
   group_by(color, clarity, colclar, cut) %>%
   summarise(m_price = mean(price))
df <- df1 %>%
   pivot_wider(id_cols = colclar,
                names_from = cut,
                values_from = m_price,
                values_fill = 0L)
mat <- as.matrix(df[,2:6])
rownames(mat) <- df["colclar"]
dst <- dist(mat)
hc_x <- hclust(dst)
lvl <- rownames(mat)[hc_x$order]
df1["colclar"] <- factor(df1["colclar"], levels = lvl)
dendro <- dendro_data(hc_x)
p <- ggplot(df1, aes(x = colclar, cut)) +
   geom_tile(aes(fill = m_price)) +
   viridis::scale_fill_viridis(option = "magma") +
   theme(axis.text.x = element_text(angle = 90, vjust = .5))
p +
   geom_xsidesegment(data = dendro$segments,aes(x = x, y = y, xend = xend, yend = yend))
```

Description

The `xside` and `yside` variants of `geom_text`.

Usage

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

position = "identity",
...,
parse = FALSE,
nudge_x = 0,
nudge_y = 0,
check_overlap = FALSE,
nal.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE
}

gem_ysidetext(
  mapping = NULL,
data = NULL,
stat = "identity",
position = "identity",
...,parse = FALSE,
nudge_x = 0,
nudge_y = 0,
check_overlap = FALSE,
nal.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE
)

Arguments

mapping Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()}. If specified and \texttt{inherit.aes = TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}.
A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x, 10)}).

stat The statistical transformation to use on the data for this layer, as a string.
positional Position adjustment, either as a string, or the result of a call to a position adjustment function. Cannot be jointly specified with \texttt{nudge_x} or \texttt{nudge_y}.

... Other arguments passed on to \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red"} or \texttt{size = 3}. They may also be parameters to the paired geom/stat.
parse | If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

nudge_x, nudge_y | Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.

check_overlap | If TRUE, text that overlaps previous text in the same layer will not be plotted. check_overlap happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling geom_text(). Note that this argument is not supported by geom_label().

na.rm | If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

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. It can also be a named logical vector to finely select the aesthetics to display.

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().

**Value**

XLayer or YLayer object to be added to a ggplot object

---

**geom_xsidetile**

*Side tile plot*

**Description**

The xside and yside variants of geom_tile

**Usage**

```r
geom_xsidetile(
    mapping = NULL,
    data = NULL,
    stat = "identity",
    position = "identity",
    ...,
    linejoin = "mitre",
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
)
```

```r
geom_ysidetile(
    mapping = NULL,
```
data = NULL,
stat = "identity",
position = "identity",
...,  
linejoin = "mitre",
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE  
)

Arguments

mapping Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()}. If specified and \texttt{inherit.aes} = \texttt{TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.
data The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}.
A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x, 10)}).
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 \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red" or size = 3}. They may also be parameters to the paired geom/stat.
linejoin Line join style (round, mitre, bevel).
na.rm If \texttt{FALSE}, the default, missing values are removed with a warning. If \texttt{TRUE}, missing values are silently removed.
show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes If \texttt{FALSE}, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders()}.

Value

XLayer or YLayer object to be added to a ggplot object
Examples

```r
code
library(dplyr)
library(tidyr)
df <- mutate(diamonds,
  colclar = interaction(color, clarity, sep = " ", drop = TRUE)) %>%
  group_by(color, clarity, colclar, cut) %>%
  summarise(m_price = mean(price))

xside_data <- df %>%
  ungroup() %>%
  select(colclar, clarity, color) %>%
  mutate_all(~factor(as.character(.x), levels = levels(.x))) %>%
  pivot_longer(cols = c(clarity, color)) %>%
  distinct()

p <- ggplot(df, aes(x = colclar, cut)) +
  geom_tile(aes(fill = m_price)) +
  viridis::scale_fill_viridis(option = "magma") +
  theme(axis.text.x = element_blank())

p + geom_xsidetile(data = xside_data, aes(y = name, xfill = value)) +
  guides(xfill = guide_legend(nrow = 8))
```

Description

The `xside` and `yside` variants of `geom_violin`.

Usage

```r
geom_xsideviolin(
  mapping = NULL,
  data = NULL,
  stat = "ydensity",
  position = "dodge",
  ...,
  draw_quantiles = NULL,
  trim = TRUE,
  scale = "area",
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

```r
geom_ysideviolin()
```
```
mapping = NULL,  
data = NULL,    
stat = "ydensity", 
position = "dodge", 
...,           
draw_quantiles = NULL, 
trim = TRUE,    
scale = "area",  
na.rm = FALSE,  
orientation = "y", 
show.legend = NA, 
inherit.aes = TRUE 
```
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. It can also be a named logical vector to finely select the aesthetics to display.

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()`.

Value

XLayer or YLayer object to be added to a ggplot object

See Also  

`geom_*sideboxplot`

Examples

```r
df <- expand.grid(UpperCase = LETTERS, LowerCase = letters)
df$Combo_Index <- as.integer(df$UpperCase)*as.integer(df$LowerCase)

p1 <- ggplot(df, aes(UpperCase, LowerCase)) + geom_tile(aes(fill = Combo_Index))

#sideviolins
#Note - Mixing discrete and continuous axis scales
#using xsideviolins when the y aesthetic was previously
#mapped with a continuous variable will prevent
#any labels from being plotted. This is a feature that
#will hopefully be added to ggside in the future.

p1 + geom_xsideviolin(aes(y = Combo_Index)) + geom_ysideviolin(aes(x = Combo_Index))

#sideviolins with swapped orientation
#Note - Discrete before Continuous
#If you are to mix Discrete and Continuous variables on
#one axis, ggplot2 prefers the discrete variable to be mapped
#BEFORE the continuous.

ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) + geom_xsideviolin(aes(y = Species), orientation = "y") + geom_point()

#Alternatively, you can recast the value as a factor and then
# a numeric

ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) + geom_point() + geom_xsideviolin(aes(y = as.numeric(Species)), orientation = "y") + geom_ysideviolin(aes(x = as.numeric(Species)), orientation = "x")
```

ggside options

Description

Set characteristics of side panels

Usage

```r
ggside(
  x.pos = "top",
  y.pos = "right",
  scales = "fixed",
  collapse = NULL,
  draw_x_on = c("default", "main", "side"),
  draw_y_on = c("default", "main", "side"),
  strip = c("default", "main")
)
```

Arguments

- `x.pos`  
  x side panel can either take "top" or "bottom"

- `y.pos`  
  y side panel can either take "right" or "left"

- `scales`  
  Determines side panel’s unaligned axis scale. Inputs are similar to facet_* scales function. Default is set to "fixed", but "free_x", "free_y" and "free" are acceptable inputs. For example, xside panels are aligned to the x axis of the main panel. Setting "free" or "free_y" will cause all y scales of the x side Panels to be independent.

- `collapse`  
  Determines if side panels should be collapsed into a single panel. Set "x" to collapse all x side panels, set "y" to collapse all y side panels, set "all" to collapse both x and y side panels.

- `draw_x_on`, `draw_y_on`  
  Determines where the axis is rendered. For example: By default, the bottom x-axis is rendered on the bottom most panel per column. If set to "main", then the axis is rendered on the bottom of the bottom most main panel. If set to "side", then the x-axis is rendered on the bottom of the bottom most side panel(s). You may apply this logic to all axis positions.

- `strip`  
  Determines if the strip should be rendered on the main plot or on their default locations. Only has an effect on facet_grid.

Value

A object of class 'ggside_options' or to be added to a ggpplot

See Also

For more information regarding the ggside api: see xside or yside
Description

S3 class that converts old Facet into one that is compatible with ggside. Can also update ggside on the object. Typically, the new ggproto will inherit from the object being replaced.

check_scales_collapse is a helper function that is meant to be called after the inherited Facet’s compute_layout method

sidePanelLayout is a helper function that is meant to be called after the inherited Facet’s compute_layout method and after check_scales_collapse

prep_map_data is a utility function to help modify the data and layout variables of the Facet’s $map_data method. This will be sure to include the column PANEL_TYPE that will assist where data should map to. Please be sure to join against this column as well.

Usage

as_ggsideFacet(facet, ggside)

check_scales_collapse(data, params)

sidePanelLayout(layout, ggside)

prep_map_data(layout, data)

Arguments

facet Facet ggproto Object to replace
ggside ggside object to update
data data passed through ggproto object
params parameters passed through ggproto object
layout layout computed by inherited ggproto Facet compute_layout method

Value

ggproto object that can be added to a ggplot object

Extended Facets

The following is a list ggplot2 facets that are available to use by ggside base.

- FacetNull -> FacetSideNull
- FacetGrid -> FacetSideGrid
- FacetWrap -> FacetSideWrap
Description

The `xside` and `yside` variants of `scale_x_continuous/scale_y_continuous`, `scale_xsidey_continuous` enables better control on how the y-axis is rendered on the xside panel and `scale_yxsidex_continuous` enables better control on how the x-axis is rendered on the yside panel.

Arguments

- **name**: The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
- **breaks**: One of:
  - `NULL` for no breaks
  - `waiver()` for the default breaks computed by the transformation object
  - A numeric vector of positions
  - A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.
- **minor_breaks**: One of:
  - `NULL` for no minor breaks
  - `waiver()` for the default breaks (one minor break between each major break)
  - A numeric vector of positions
  - A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation.
- **n.breaks**: An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.
- **labels**: One of:
  - `NULL` for no labels
  - `waiver()` for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as breaks)
  - A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.
- **limits**: One of:
  - `NULL` to use the default scale range
  - A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
• A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang `lambda` function notation. Note that setting limits on positional scales will remove data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`expand` For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

`oob` One of:
  • Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang `lambda` function notation.
  • The default (`scales::censor()`) replaces out of bounds values with `NA`.
  • `scales::squish()` for squishing out of bounds values into range.
  • `scales::squish_infinite()` for squishing infinite values into range.

`na.value` Missing values will be replaced with this value.

`trans` For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called `<name>_trans` (e.g., `scales::boxcox_trans()`). You can create your own transformation with `scales::trans_new()`.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`position` For position scales, The position of the axis. left or right for y axes, top or bottom for x axes.

Value

`ggside_scale` object inheriting from `ggplot2::ScaleContinuousPosition`

Examples

```r
library(ggside)
library(ggplot2)
# adding continuous y-scale to the x-side panel, when main panel mapped to discrete data
ggplot(mpg, aes(hwy, class, colour = class)) +
  geom_boxplot() +
  geom_xsidey_continuous(position = "stack") +
  theme(ggside.panel.scale = .3) +
  scale_xsidey_continuous(minor_breaks = NULL, limits = c(NA,1))

#If you need to specify the main scale, but need to prevent this from
```
# affecting the side scale. Simply add the appropriate `scale_{x}|y_side*_{*}()` function.

ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  geom_xsidehistogram() +
  geom_ysidehistogram() +
  scale_x_continuous(
    breaks = seq(1, 6, 1),
    # would otherwise remove the histogram
    # as they have a lower value of 0.
    limits = (c(1, 6))
  ) +
  scale_ysidex_continuous() # ensures the x-axis of the y-side panel has its own scale.

---

**Description**

The `xside` and `yside` variants of `scale_x_discrete/scale_y_discrete`. `scale_xsidey_discrete` enables better control on how the y-axis is rendered on the xside panel and `scale_ysidex_discrete` enables better control on how the x-axis is rendered on the yside panel.

**Arguments**

- **expand**: For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

- **guide**: A function used to create a guide or its name. See `guides()` for more information.

- **position**: For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

**Value**

ggside_scale object inheriting from `ggplot2::ScaleDiscretePosition`

**Examples**

```r
library(ggside)
library(ggplot2)
# adding discrete y-scale to the x-side panel, when main panel mapped to continuous data
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point() +
  geom_xsideboxplot(aes(y=class), orientation = "y") +
```

theme(ggside.panel.scale = .3) +
scale_xsidey_discrete(guide = guide_axis(angle = 45))

# If you need to specify the main scale, but need to prevent this from
#affecting the side scale. Simply add the appropriate `scale_*side_*()` function.
ggplot(mpg, aes(class, displ)) +
  geom_boxplot() +
  geom_ysideboxplot(aes(x = "all"), orientation = "x") +
  scale_x_discrete(guide = guide_axis(angle = 90)) + # rotate the main panel text
  scale_ysidex_discrete() # leave side panel as default

---

is.ggside

### Description

Check ggside objects

### Usage

- `is.ggside(x)`
- `is.ggside_layer(x)`
- `is.ggside_options(x)`
- `is.ggside_scale(x)`

### Arguments

- `x` Object to test

### Value

A logical value

---

position_rescale

#### Rescale x or y onto new range in margin

### Description

Take the range of the specified axis and rescale it to a new range about a midpoint. By default the range will be calculated from the associated main plot axis mapping. The range will either be the resolution or 5% of the axis range, depending if original data is discrete or continuous respectively. Each layer called with position_rescale will possess an instance value that indexes with axis rescale. By default, each position_rescale will dodge the previous call unless instance is specified to a previous layer.
Usage

```r
position_rescale(
  rescale = "y",
  midpoint = NULL,
  range = NULL,
  location = NULL,
  instance = NULL
)
```

```r
position_yrescale(
  rescale = "y",
  midpoint = NULL,
  range = NULL,
  location = NULL,
  instance = NULL
)
```

```r
position_xrescale(
  rescale = "x",
  midpoint = NULL,
  range = NULL,
  location = NULL,
  instance = NULL
)
```

**Arguments**

- `rescale` character value of "x" or "y". specifies which mapping data will be rescaled
- `midpoint` default set to NULL. Center point about which the rescaled x/y values will reside.
- `range` default set to NULL and auto generates from main mapping range. Specifies the size of the rescaled range.
- `location` specifies where `position_rescale` should try to place midpoint. If midpoint is specified, location is ignored and placed at the specified location.
- `instance` integer that indexes rescaled axis calls. instance may be specified and if a previous layer with the same instance exists, then the same midpoint and range are used for rescaling. x and y are indexed independently.

**Format**

An object of class `PositionRescale` (inherits from `Position`, `ggproto`, `gg`) of length 10.

**Value**

a `ggproto` object inheriting from 'Position' and can be added to a `ggplot`
scale_xcolour

Scales for the *colour aesthetics

Description

These are the various scales that can be applied to the xsidebar or ysidebar colour aesthetics, such as xcolour and ycolour. They have the same usage as existing standard ggplot2 scales.

Value

returns a ggproto object to be added to a ggplot

Related Functions

• scale_xcolour_hue
• scale_ycolour_hue
• scale_xcolour_discrete
• scale_ycolour_discrete
• scale_xcolour_continuous
• scale_ycolour_continuous
• scale_xcolour_manual
• scale_ycolour_manual
• scale_xcolour_gradient
• scale_ycolour_gradient
• scale_xcolour_gradientn
• scale_ycolour_gradientn

scale_xfill

Scales for the *fill aesthetics

Description

These are the various scales that can be applied to the xsidebar or ysidebar fill aesthetics, such as xfill and yfill. They have the same usage as existing standard ggplot2 scales.

Value

returns a ggproto object to be added to a ggplot
Related Functions

- scale_xfill_hue
- scale_yfill_hue
- scale_xfill_discrete
- scale_yfill_discrete
- scale_xfill_continuous
- scale_yfill_continuous
- scale_xfill_manual
- scale_yfill_manual
- scale_xfill_gradient
- scale_yfill_gradient
- scale_xfill_gradientn
- scale_yfill_gradientn

---

scale_ycolour_hue  scale_ycolour_hue

---

Description

scale_ycolour_hue
scale_ycolour_manual
scale_ycolour_gradient
scale_ycolour_discrete
scale_ycolour_discrete
scale_ycolour_continuous
scale_ycolour_continuous

---

scale_yfill_hue  scale_yfill_hue

---

Description

scale_yfill_hue
scale_yfill_manual
scale_yfill_gradient
scale_yfill_discrete
scale_yfill_continuous
scale_yfill_continuous
stat_summarise

Summarise by grouping variable

Description

Applies a function to a specified grouping variable

Usage

stat_summarise(
  mapping = NULL,
  data = NULL,
  geom = "bar",
  position = "identity",
  ...,
  fun = NULL,
  args = list(),
  show.legend = NA,
  inherit.aes = TRUE
)

stat_summarize(
  mapping = NULL,
  data = NULL,
  geom = "bar",
  position = "identity",
  ...,
  fun = NULL,
  args = list(),
  show.legend = NA,
  inherit.aes = TRUE
)

Arguments

mapping  Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

data  The data to be displayed in this layer. There are three options:

  If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

  A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

  A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).
geom  The geometric object to use display the data
position  Position adjustment, either as a string, or the result of a call to a position adjustment function.
...  additional arguments to pass to layer.
fun  Summarising function to use. If no function provided it will default to length.
args  List of additional arguments passed to the function.
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. It can also be a named logical vector to finely select the aesthetics to display.
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().

Format

An object of class StatSummarise (inherits from Stat, ggproto, gg) of length 5.
An object of class StatSummarize (inherits from Stat, ggproto, gg) of length 5.

Value

A Layer object to be added to a ggplot

Aesthetics

Using stat_summarise requires that you use domain as an aesthetic mapping. This allows you to summarise other data instead of assuming that x is the function’s domain.

Examples

library(tidyrr)
i <- gather(iris, "key", "value", -Species)
ggplot(i, aes(Species, fill = key, domain = value)) +
  geom_bar(aes(y = after_stat(summarise)), stat = "summarise", fun = mean) +
  stat_summarise(aes(y = after_stat(summarise),
                    label = after_stat(summarise)),
                  position = position_stack(vjust = .5), geom = "text", fun = mean)

theme_ggside_grey  ggside custom themes

Description

Theme elements to help customize the look and feel of ggside’s side panels.
theme_ggside_grey

Usage

theme_ggside_grey(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_ggside_gray(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_ggside_bw(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_ggside_linedraw(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_ggside_light(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_ggside_dark(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_ggside_minimal(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)
theme_ggside_grey

)

theme_ggside_classic(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_ggside_void(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

Arguments

base_size  base font size, given in pts.
base_family base font family
base_line_size  base size for line elements
base_rect_size  base size for rect elements

Details

Incomplete themes:
Unlike the complete themes like theme_grey, ggside's variants are not considered "complete". This is because the user may want to specify the side panels separately from the theme of the main panel. This means that theme_ggside_*() functions should be called after any of ggplot2's complete themes.

ggside theme elements

ggside panel.scale, ggside panel.scale.x, ggside panel.scale.y

ggside panel.spacing, ggside panel.spacing.x, ggside panel.spacing.y

ggside panel.background

ggside panel.grid, ggside panel.grid.major, ggside panel.grid.minor, ggside panel.grid.major.x, ggside panel.grid.minor.x

ggside axis.text, ggside axis.text.x, ggside axis.text.y, ggside axis.text.x.top, ggside axis.text.x.bottom

ggside axis.line, ggside axis.line.x, ggside axis.line.y, ggside axis.line.x.top, ggside axis.line.x.bottom

ggside axis.ticks, ggside axis.ticks.x, ggside axis.ticks.y, ggside axis.ticks.x.top, ggside axis.ticks.x.bottom
**Examples**

```r
library(ggplot2)
library(ggside)

p <- ggplot(iris, aes(Sepal.Width, Petal.Length, color = Species)) +
  geom_point() +
  geom_xsidedensity() +
  geom_ysidedensity() +
  theme_dark()

p

p + theme_ggside_classic()
p + theme_ggside_void()
p + theme_ggside_linedraw() +
  theme(ggside.panel.border = element_rect(colour = "red"))
```

---

**Description**

These ggproto classes are slightly modified from their respective inherited ggproto class. The biggest difference is exposing `x/yfill`, `x/ycolour`, and `x/ycolor` as viable aesthetic mappings.

**Usage**

```r
use_xside_aes(data)

use_yside_aes(data)

parse_side_aes(data, params)
```

**Arguments**

- `data` data passed internally
- `params` params available to ggproto object

**Value**

`ggproto` object that is usually passed to `layer`
The xsid geometries

Description
xsid refers to the api of ggside. Any geom_ with xsid will plot its respective geometry along the x-axis per facet panel. By default the xsid panel will plot above the main panel. This xsid panel will always share the same scale as it’s main panel, but is expected to have a separate y-axis scaling.

Value
geom_xsid* return a XLayer object to be added to a ggplot

New Aesthetics
All xsid Geometries have xfill, xcolour/xcolor available for aesthetic mappings. These mappings behave exactly like the default counterparts except that they are considered separate scales. All xsid geometries will use xfill over fill, but will default to fill if xfill is not provided. The same goes for xcolour in respects to colour. This comes in handy if you wish to map both fill to one geometry as continuous, you can still map xfill for a separate xsid geometry without conflicts. See more information in vignette("ggside").

Exported Geometries
The following are the xsid variants of the ggplot2 Geometries

• geom_xsidebar
• geom_xsideboxplot
• geom_xsidecol
• geom_xsidedensity
• geom_xsidefreqpoly
• geom_xsidehistogram
• geom_xsideline
• geom_xsidepath
• geom_xsidepoint
• geom_xsidetext
• geom_xsidetile
• geom_xsideviolin

See Also
ysid
The yside geometries

Description

yside refers to the api of ggsdien. Any geom_ with yside will plot its respective geometry along the y-axis per facet panel. The yside panel will plot to the right of the main panel by default. This yside panel will always share the same scale as it’s main panel, but is expected to have a separate x-axis scaling.

Value

geom_yside* return a YLayer object to be added to a ggplot

New Aesthetics

All yside Geometries have yfill, ycolour/ycolor available for aesthetic mappings. These mappings behave exactly like the default counterparts except that they are considered separate scales. All yside geometries will use yfill over fill, but will default to fill if yfill is not provided. The same goes for ycolour in respects to colour. This comes in handy if you wish to map both fill to one geometry as continuous, you can still map yfill for a separate yside geometry without conflicts. See more information in vignette("ggsdien").

#' @section Exported Geometries:

The following are the yside variants of the ggplot2 Geometries

- geom_ysidebar
- geom_ysideboxplot
- geom_ysidecol
- geom_ysidedensity
- geom_ysidefreqpoly
- geom_ysidehistogram
- geom_ysideline
- geom_ysidepath
- geom_ysidepoint
- geom_ysidetext
- geom_ysidetile
- geom_ysideviolin

See Also

xside
Index

* datasets
  ggside-ggproto-facets, 33
  ggside-ggproto-facets, 33
  position_rescale, 37
  stat_summarise, 41
  use_xside_aes, 45
  aes(), 4, 7, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 41
  aes_(), 4, 7, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 41
  as_ggsidCoord, 2
  as_ggsidFacet (ggsid-ggproto-facets), 33
  borders(), 5, 8, 10, 12, 14, 17, 19, 21, 23, 24, 27, 28, 31, 42
  check_scalesCollapse
    (ggsid-ggproto-facets), 33
  coord_cartesian(), 35
  expansion(), 35, 36
  FacetGrid, 33
  FacetNull, 33
  FacetSideGrid (ggsid-ggproto-facets), 33
  FacetSideNull (ggsid-ggproto-facets), 33
  FacetSideWrap (ggsid-ggproto-facets), 33
  FacetWrap, 33
  fortify(), 4, 7, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 41
  geom_*freqpoly (geom_xsidefreqpoly), 11
  geom_*sidebar (geom_xsidebar), 3
  geom_*sidebar(), 17
  geom_*sideboxplot, 31
  geom_*sideboxplot (geom_xsideboxplot), 6
  geom_*sidedensity (geom_xsidedensity), 9
  geom_*sidefunction
    (geom_xsidefunction), 13
  geom_*sidehistrogram
    (geom_xsidehistrogram), 15
  geom_*sidelabel (geom_xsidelabel), 17
  geom_*sideline (geom_xsideline), 19
  geom_*sidepoint (geom_xsidepoint), 21
  geom_*sidesegment (geom_xsidesegment), 23
  geom_*sidetext (geom_xsidetext), 25
  geom_*sidetile (geom_xsidetile), 27
  geom_*sideviolin, 8
  geom_*sideviolin (geom_xsideviolin), 29
  geom_bar, 3
  geom_boxplot, 6
  geom_col, 3
  geom_density, 9
  geom_freqpoly, 11
  geom_function, 13
  geom_histogram, 15
  geom_label, 17
  geom_line, 19
  geom_path, 19
  geom_point, 21
  geom_segment, 23
  geom_text, 25
  geom_tile, 27
  geom_violin, 29
  geom_xsidebar, 3, 3, 46
  geom_xsideboxplot, 6, 6, 46
  geom_xsidecol, 3, 46
  geom_xsidecol (geom_xsidebar), 3
  geom_xsidedensity, 9, 9, 46
  geom_xsidefreqpoly, 11, 11, 46
  geom_xsidefunction, 13
  geom_xsidehistrogram, 5, 15, 15, 46
  geom_xsidelabel, 17
  geom_xsideline, 19, 46
  geom_xsidepath, 46
geom_xsidepath (geom_xsideline), 19
geom_xsidepoint, 21, 46
geom_xsidepoint(), 21
geom_xsidesegment, 23
gbm_xsidetext, 25, 46
gbm_xsidetile, 27, 46
gbm_xsideviolin, 29, 46
gbm_ysidebar, 3, 47
gbm_ysidebar (geom_xsidebar), 3
gbm_ysideboxplot, 6, 47
gbm_ysideboxplot (geom_xsideboxplot), 6
gbm_ysidecol, 3, 47
gbm_ysidecol (geom_xsidebar), 3
gbm_y sidelength, 9, 47
gbm_y sidewidgetdensity (geom_xsidelengthdensity), 9
gbm_y sidefreqpoly, 11, 47
gbm_y sidefreqpoly (geom_xsidefreqpoly), 11
gbm_y sidefunction (geom_xsidefunction), 13
gbm_y sidestatistic (geom_xsidestatistic), 15
gbm_y sideviolin (geom_xsideviolin), 17
gbm_y sideviolin (geom_xsideline), 19
gbm_y sidemargin, 47
gbm_y sidepath (geom_xsidepath), 19
gbm_y sidepath (geom_xsidepoint), 21
gbm_y sidetext, 21
gbm_y sidetext (geom_xsidetext), 21
gbm_y sidetext (geom_xsidetext), 25
gbm_y sidetile, 27
gbm_y sidetile (geom_xsidetile), 27
gbm_y sidetile (geom_xsidetile), 29
gbm_y sideviolin (geom_xsideviolin), 29
gbm_y sideviolin (use_xside_aes), 45
gbm_y sideviolin (geom_xsideline), 46
gbm_y sideviolin (use_xside_aes), 45
gbm_y sideviolin (geom_xsidepath), 46
gbm_y sidemargin (geom_xsidemargin), 46
gbm_y sidemargin (use_xside_aes), 46
gbm_y sidemargin (geom_xsidetext), 46
gbm_y sidemargin (use_xside_aes), 46
gbm_y sidemargin (geom_xsidetile), 46
gbm_y sidemargin (use_xside_aes), 46
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scale_x_continuous</code></td>
<td>34</td>
</tr>
<tr>
<td><code>scale_x_discrete</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_xcolor</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xcolor_continuous</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xcolor_discrete</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xcolor_gradient</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xcolor_gradientn</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xcolor_manual</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xfill</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xfill_continuous</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xfill_discrete</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xfill_gradient</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xfill_gradientn</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xfill_manual</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_xside_continuous</code></td>
<td>34</td>
</tr>
<tr>
<td><code>scale_xside_discrete</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_xside</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_xside_continuous</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_xside_discrete</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_xside_function</code></td>
<td>13</td>
</tr>
<tr>
<td><code>scale_y_continuous</code></td>
<td>34</td>
</tr>
<tr>
<td><code>scale_y_discrete</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_ycolor</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_ycolor_continuous</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_ycolor_discrete</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_ycolor_gradient</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_ycolor_gradientn</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_ycolor_hue</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_yfill</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_yfill_continuous</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_yfill_discrete</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_yfill_gradient</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_yfill_gradientn</code></td>
<td>40</td>
</tr>
<tr>
<td><code>scale_yfill_manual</code></td>
<td>39</td>
</tr>
<tr>
<td><code>scale_yside_continuous</code></td>
<td>34</td>
</tr>
<tr>
<td><code>scale_yside_discrete</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_yside</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_yside_continuous</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_yside_discrete</code></td>
<td>36</td>
</tr>
<tr>
<td><code>scale_yside_function</code></td>
<td>13</td>
</tr>
<tr>
<td><code>sidePanelLayout</code></td>
<td>33</td>
</tr>
<tr>
<td><code>stat_summarise</code></td>
<td>41</td>
</tr>
<tr>
<td><code>stat_summarize</code></td>
<td>41</td>
</tr>
<tr>
<td><code>stat_xsidefunction</code></td>
<td>13</td>
</tr>
<tr>
<td><code>stat_ysidefunction</code></td>
<td>13</td>
</tr>
<tr>
<td><code>StatSummarise</code></td>
<td>41</td>
</tr>
<tr>
<td><code>StatSummarize</code></td>
<td>41</td>
</tr>
<tr>
<td><code>theme_ggside_bw</code></td>
<td>42</td>
</tr>
<tr>
<td><code>theme_ggside_classic</code></td>
<td>42</td>
</tr>
<tr>
<td><code>theme_ggside_dark</code></td>
<td>42</td>
</tr>
<tr>
<td><code>theme_ggside_gray</code></td>
<td>42</td>
</tr>
</tbody>
</table>

**INDEX**
theme_ggside_grey, 42
theme_ggside_light (theme_ggside_grey), 42
theme_ggside_linedraw (theme_ggside_grey), 42
theme_ggside_minimal (theme_ggside_grey), 42
theme_ggside_void (theme_ggside_grey), 42
theme_grey, 44
transformation object, 34
use_xside_aes, 45
use_yside_aes (use_xside_aes), 45
xside, 3, 6, 9, 11, 13, 15, 17, 19, 23, 25, 27, 29, 32, 34, 36, 46, 47
yside, 3, 6, 9, 11, 13, 15, 17, 19, 23, 25, 27, 29, 32, 34, 36, 46, 47