

Package ‘ggside’

September 13, 2025

Type Package

Title Side Grammar Graphics

Version 0.4.0

Maintainer Justin Landis <jtlandis314@gmail.com>

Description The grammar of graphics as shown in 'ggplot2' has provided an expressive API for users to build plots. 'ggside' extends 'ggplot2' by allowing users to add graphical information about one of the main panel's axis using a familiar 'ggplot2' style API with tidy data. This package is particularly useful for visualizing metadata on a discrete axis, or summary graphics on a continuous axis such as a boxplot or a density distribution.

License MIT + file LICENSE

URL <https://github.com/jtlandis/ggside>

BugReports <https://github.com/jtlandis/ggside/issues>

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

Depends R (>= 4.1), ggplot2 (>= 4.0.0)

Imports grid, gtable, rlang, scales (>= 1.3.0), cli, glue, stats, tibble, vctrs, S7, lifecycle

Suggests tidyr, dplyr, testthat (>= 3.0.3), knitr, rmarkdown, vdiffr (>= 1.0.0), ggdendro, viridis, waldo

Config/testthat/edition 3

Collate 'utils-ggproto.R' 'utils-calls.R' 'utils-ggplot2-reimpl-.R' 'utils-constructors.R' 'side-layer.R' 'constructor-.R' 'utils-.R' 'ggside.R' 'utils-side-facet.R' 'side-facet_.R' 'utils-side-coord.R' 'side-coord-cartesian.R' 'add_gg.R' 'ggplot_add.R' 'side-layout-.r' 'all_classes.r' 'geom-sideabline.r' 'geom-sidebar.r' 'geom-sideboxplot.r' 'geom-sidecol.r' 'geom-sidedensity.r' 'geom-sidefreqpoly.r' 'geom-sidefunction.r' 'geom-sidehistogram.r' 'geom-sidehline.r' 'geom-sidelabel.r' 'geom-sideline.r' 'geom-sidepath.r'

'geom-sidepoint.r' 'geom-sidesegment.r' 'geom-sidetext.r'
 'geom-sidetile.r' 'geom-sideviolin.r' 'geom-sidevline.r'
 'ggside-ggproto.r' 'ggside-package.r' 'ggside-themes.R'
 'plot-construction.R' 'position_rescale.r' 'scales-sides-.R'
 'scales-xycolour.R' 'scales-xyfill.R'
 'utils-ggplot2-reimpl-facet.R' 'side-facet-wrap.R'
 'side-facet-grid.R' 'side-facet-null.R' 'stats.r'
 'update_ggplot.R' 'z-deprecated.R' 'zzz.R'

NeedsCompilation no

Author Justin Landis [aut, cre] (ORCID:

<<https://orcid.org/0000-0001-5501-4934>>)

Repository CRAN

Date/Publication 2025-09-13 05:10:41 UTC

Contents

as_ggside	3
check_scales_collapse	4
class_definitions	5
geom_xsideabline	5
geom_xsidebar	8
geom_xsideboxplot	12
geom_xsidedensity	17
geom_xsidefreqpoly	19
geom_xsidefunction	22
geom_xsidehistogram	25
geom_xsidelabel	29
geom_xsideline	32
geom_xsidepoint	35
geom_xsidesegment	38
geom_xsidetext	41
geom_xsidetile	43
geom_xsideviolin	46
ggside	50
ggside-deprecated	51
ggside-scales	51
ggside-scales-binned	51
ggside-scales-continuous	54
ggside-scales-discrete	57
ggside_coord	59
ggside_geom	60
ggside_layer	61
ggside_layout	63
is_ggside	63
parse_side_aes	64
position_rescale	64

<i>as_ggside</i>	3
scale_xcolour	65
scale_xfill	66
scale_ycolour_hue	67
scale_yfill_hue	67
stat_summarise	67
theme_ggside_grey	70
xside	72
yside	73
Index	75

<i>as_ggside</i>	<i>Explicit conversion to ggside object</i>
------------------	---

Description

Function is only exported for possible extensions to ggside. ggplot2 objects are implicitly converted to ggside objects by 'adding' a ggside object such as a ggside_layer object.

Usage

```
as_ggside(x, ...)

## Default S3 method:
as_ggside(x, ...)

## S3 method for class 'ggplot'
as_ggside(x, ggside = NULL, ...)

## S3 method for class '`ggside::ggside`'
as_ggside(x, ggside = NULL, ...)

## S3 method for class 'ggside'
as_ggside(x, ggside = NULL, ...)
```

Arguments

<i>x</i>	an object to convert
<i>...</i>	unused argument
<i>ggside</i>	new ggside object to add

check_scales_collapse *Extending base ggproto classes for ggside*

Description

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

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.

Usage

```
check_scales_collapse(data, params)
```

```
sidePanelLayout(layout, ggside)
```

```
ggside_facet(facet, ggside)
```

Arguments

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

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

class_definitions	<i>Class Definitions</i>
-------------------	--------------------------

Description

This documentation provides an overview of the *S7* and *ggproto* classes used in the *ggside* package.

ggproto classes

- `class_ggside_opt` is a subclass of `class_ggproto` and is more described in the [ggside-options](#) documentation.
- `class_ggside_layer` is a subclass of `class_ggproto` and is more described in the [ggside-layers](#) documentation.
- `class_ggside_scale` is a subclass of `class_ggproto` and is more described in the [ggside-scales](#) documentation.

S7 classes

- `class_ggside` is a subclass of *ggplot2*'s `class_ggplot` and is used to represent a *ggplot* object with *ggside* options.

geom_xsideabline	<i>Side Reference lines</i>
------------------	-----------------------------

Description

The `xside` and `yside` variants of `geom_abline`, `geom_hline` and `geom_vline` are `geom_*abline`, `geom_*hline`, and `geom_*vline`.

Usage

```
geom_xsideabline(  
  mapping = NULL,  
  data = NULL,  
  ...,  
  slope,  
  intercept,  
  na.rm = FALSE,  
  show.legend = NA  
)
```

```
geom_ysideabline(  
  mapping = NULL,  
  data = NULL,
```

```
    ...,
    slope,
    intercept,
    na.rm = FALSE,
    show.legend = NA
  )

geom_xsidehline(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  yintercept,
  na.rm = FALSE,
  show.legend = NA
)

geom_ysidehline(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  yintercept,
  na.rm = FALSE,
  show.legend = NA
)

geom_xsidevline(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  xintercept,
  na.rm = FALSE,
  show.legend = NA
)

geom_ysidevline(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  xintercept,
  na.rm = FALSE,
  show.legend = NA
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() .
data	<p>The data to be displayed in this layer. There are three options:</p> <p>If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>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.</p> <p>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. <code>~ head(.x, 10)</code>).</p>
...	<p>Other arguments passed on to layer()'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can <i>not</i> be passed through Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of layer() may also be passed on through This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	<p>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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.</p>
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position.

- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as "jitter".
- For more information and other ways to specify the position, see the [layer position](#) documentation.

`xintercept`, `yintercept`, `slope`, `intercept`

Parameters that control the position of the line specifically for the [xside](#) or [yside](#) variants. If these are set, `data`, `mapping` and `show.legend` are overridden.

geom_xsidebar

Side bar Charts

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

```
geom_xsidebar(
  mapping = NULL,
  data = NULL,
  stat = "count",
  position = "stack",
  ...,
  just = 0.5,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  orientation = "x"
)
```

```
geom_ysidebar(
  mapping = NULL,
  data = NULL,
  stat = "count",
  position = "stack",
  ...,
  just = 0.5,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
```



```

    show.legend = NA,
    inherit.aes = TRUE,
    orientation = "y"
  )

```

```

geom_xsidecol(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "stack",
  ...,
  just = 0.5,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

```

```

geom_ysidecol(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "stack",
  ...,
  just = 0.5,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  orientation = "y"
)

```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>

stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
just	<p>Adjustment for column placement. Set to 0.5 by default, meaning that columns will be centered about axis breaks. Set to 0 or 1 to place columns to the left/right of axis breaks. Note that this argument may have unintended behaviour when used with alternative positions, e.g. <code>position_dodge()</code>.</p>

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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. annotation_borders() .
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 <i>Orientation</i> section for more detail.

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

```
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	<i>Side boxplots</i>
-------------------	----------------------

Description

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

Usage

```
geom_xsideboxplot(  
  mapping = NULL,  
  data = NULL,  
  stat = "boxplot",  
  position = "dodge2",  
  ...,  
  outliers = TRUE,  
  outlier.colour = NULL,  
  outlier.color = NULL,  
  outlier.fill = NULL,  
  outlier.shape = NULL,  
  outlier.size = NULL,  
  outlier.stroke = 0.5,  
  outlier.alpha = NULL,  
  whisker.colour = NULL,  
  whisker.color = NULL,  
  whisker.linetype = NULL,  
  whisker.linewidth = NULL,  
  staple.colour = NULL,  
  staple.color = NULL,  
  staple.linetype = NULL,  
  staple.linewidth = NULL,  
  median.colour = NULL,  
  median.color = NULL,  
  median.linetype = NULL,  
  median.linewidth = NULL,  
  box.colour = NULL,  
  box.color = NULL,  
  box.linetype = NULL,  
  box.linewidth = NULL,
```

```
    notch = FALSE,
    notchwidth = 0.5,
    staplewidth = 0,
    varwidth = FALSE,
    na.rm = FALSE,
    orientation = "x",
    show.legend = NA,
    inherit.aes = TRUE
  )

geom_ysideboxplot(
  mapping = NULL,
  data = NULL,
  stat = "boxplot",
  position = "dodge2",
  ...,
  outliers = TRUE,
  outlier.colour = NULL,
  outlier.color = NULL,
  outlier.fill = NULL,
  outlier.shape = NULL,
  outlier.size = NULL,
  outlier.stroke = 0.5,
  outlier.alpha = NULL,
  whisker.colour = NULL,
  whisker.color = NULL,
  whisker.linetype = NULL,
  whisker.linewidth = NULL,
  staple.colour = NULL,
  staple.color = NULL,
  staple.linetype = NULL,
  staple.linewidth = NULL,
  median.colour = NULL,
  median.color = NULL,
  median.linetype = NULL,
  median.linewidth = NULL,
  box.colour = NULL,
  box.color = NULL,
  box.linetype = NULL,
  box.linewidth = NULL,
  notch = FALSE,
  notchwidth = 0.5,
  staplewidth = 0,
  varwidth = FALSE,
  na.rm = FALSE,
  orientation = "y",
  show.legend = NA,
  inherit.aes = TRUE
```

)

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A <code>Stat</code> ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as <code>"jitter"</code>. • For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to layer()'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

	<ul style="list-style-type: none"> • When constructing a layer using a <code>stat_*</code>() function, the <code>...</code> argument can be used to pass on parameters to the <code>geom</code> part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The <code>geom</code>'s documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*</code>() function, the <code>...</code> argument can be used to pass on parameters to the <code>stat</code> part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The <code>stat</code>'s documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through <code>...</code>. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
<code>outliers</code>	Whether to display (TRUE) or discard (FALSE) outliers from the plot. Hiding or discarding outliers can be useful when, for example, raw data points need to be displayed on top of the boxplot. By discarding outliers, the axis limits will adapt to the box and whiskers only, not the full data range. If outliers need to be hidden and the axes needs to show the full data range, please use <code>outlier.shape = NA</code> instead.
<code>outlier.colour</code> , <code>outlier.color</code> , <code>outlier.fill</code> , <code>outlier.shape</code> , <code>outlier.size</code> , <code>outlier.stroke</code> , <code>outlier.alpha</code>	Default aesthetics for outliers. Set to NULL to inherit from the data's aesthetics.
<code>whisker.colour</code> , <code>whisker.color</code> , <code>whisker.linetype</code> , <code>whisker.linewidth</code>	Default aesthetics for the whiskers. Set to NULL to inherit from the data's aesthetics.
<code>staple.colour</code> , <code>staple.color</code> , <code>staple.linetype</code> , <code>staple.linewidth</code>	Default aesthetics for the staples. Set to NULL to inherit from the data's aesthetics. Note that staples don't appear unless the <code>staplewidth</code> argument is set to a non-zero size.
<code>median.colour</code> , <code>median.color</code> , <code>median.linetype</code> , <code>median.linewidth</code>	Default aesthetics for the median line. Set to NULL to inherit from the data's aesthetics.
<code>box.colour</code> , <code>box.color</code> , <code>box.linetype</code> , <code>box.linewidth</code>	Default aesthetics for the boxes. Set to NULL to inherit from the data's aesthetics.
<code>notch</code>	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.
<code>notchwidth</code>	For a notched box plot, width of the notch relative to the body (defaults to <code>notchwidth = 0.5</code>).
<code>staplewidth</code>	The relative width of staples to the width of the box. Staples mark the ends of the whiskers with a line.
<code>varwidth</code>	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 <code>weight</code> aesthetic).
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>orientation</code>	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 <i>Orientation</i> 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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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. annotation_borders() .

Value

XLayer or YLayer object to be added to a ggplot object

See Also

[geom_*sideviolin](#)

Examples

```
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

p1 + geom_xsideboxplot(aes(y = Combo_Index)) +
  geom_ysideboxplot(aes(x = Combo_Index)) +
  #when mixing continuous/discrete scales
  #use the following helper functions
  scale_xsidey_continuous() +
  scale_ysidex_continuous()

#sideboxplots with swapped orientation
#Note: The order of the layers affects the default
# scale type. If you were to omit the last two scales, the
# data labels may be affected
ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) +
  geom_xsideboxplot(aes(y = Species), orientation = "y") +
  geom_point() +
  scale_y_continuous() + scale_xsidey_discrete()

#If using the scale_(xsidey|ysidex)* functions are a bit cumbersome,
# Take extra care to recast your data types.
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_xsidedensity	Side density distributions
-------------------	----------------------------

Description

The `xside` and `yside` variants of `geom_density` is `geom_xsidedensity` and `geom_ysidedensity`.

Usage

```
geom_xsidedensity(  
  mapping = NULL,  
  data = NULL,  
  stat = "density",  
  position = "identity",  
  ...,  
  outline.type = "upper",  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 10,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  orientation = "x"  
)
```

```
geom_ysidedensity(  
  mapping = NULL,  
  data = NULL,  
  stat = "density",  
  position = "identity",  
  ...,  
  outline.type = "upper",  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 10,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  orientation = "y"  
)
```

Arguments

mapping	Set of aesthetic mappings created by <code>aes()</code> . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	Use to override the default connection between <code>geom_density()</code> and <code>stat_density()</code> .
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can <i>not</i> be passed through Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
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.

lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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. annotation_borders() .
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 <i>Orientation</i> section for more detail.

Value

XLayer or YLayer object to be added to a ggplot object

Examples

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

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

geom_xsidefreqpoly	<i>Side Frequency Polygons</i>
--------------------	--------------------------------

Description

The *xside* and *yside* variants of [geom_freqpoly](#) is [geom_xsidefreqpoly](#) and [geom_ysidefreqpoly](#).

Usage

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

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

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>.

	<ul style="list-style-type: none"> • For more information and other ways to specify the stat, see the layer stat documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through ... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
na.rm	<p>If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.</p>
show.legend	<p>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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.</p>
inherit.aes	<p>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>annotation_borders()</code>.</p>
orientation	<p>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</p>

given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
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)
```

geom_xsidefunction	<i>Side function plot</i>
--------------------	---------------------------

Description

The [xside](#) and [yside](#) variants of [geom_function](#)

Usage

```
geom_xsidefunction(
  mapping = NULL,
  data = NULL,
  stat = "function",
  position = "identity",
  ...,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

```
stat_xsidefunction(
  mapping = NULL,
  data = NULL,
  geom = "function",
  position = "identity",
  ...,
  fun,
  xlim = NULL,
  n = 101,
```

```

    args = list(),
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
  )

geom_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() . If specified and <code>inherit.aes = TRUE</code> (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 <code>stat_function()</code> , do not use.
stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: <ul style="list-style-type: none"> • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.

position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through ... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
arrow	Arrow specification, as created by <code>grid::arrow()</code> .
arrow.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).
linemitre	Line mitre limit (number greater than 1).
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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. annotation_borders() .
geom	The geometric object to use to display the data for this layer. When using a <code>stat_*()</code> function to construct a layer, the <code>geom</code> argument can be used to override the default coupling between stats and geoms. The <code>geom</code> argument accepts the following: <ul style="list-style-type: none"> • A Geom ggproto subclass, for example <code>GeomPoint</code>. • A string naming the geom. To give the geom as a string, strip the function name of the <code>geom_</code> prefix. For example, to use <code>geom_point()</code>, give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
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, specify the range of the function.
n	Number of points to interpolate along the x axis.
args	List of additional arguments passed on to the function defined by <code>fun</code> .
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

```
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_xsidedensity(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
```

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

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

```
geom_ysidehistogram(  
  mapping = NULL,  
  data = NULL,  
  stat = "bin",  
  position = "stack",  
  ...,  
  binwidth = NULL,  
  bins = NULL,  
  orientation = "y",  
  lineend = "butt",  
  linejoin = "mitre",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

Arguments

<code>mapping</code>	Set of aesthetic mappings created by <code>aes()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply <code>mapping</code> if there is no plot mapping.
<code>data</code>	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> .

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. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as "count".
- For more information and other ways to specify the stat, see the [layer stat](#) documentation.

position

A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as "jitter".
- For more information and other ways to specify the position, see the [layer position](#) documentation.

...

Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.

	<ul style="list-style-type: none"> • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through <code>...</code>. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
<code>binwidth</code>	<p>The width of the bins. Can be specified as a numeric value or as a function that takes <code>x</code> after scale transformation as input and returns a single numeric value. 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 <code>bins</code>, 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.</p> <p>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.</p>
<code>bins</code>	Number of bins. Overridden by <code>binwidth</code> . Defaults to 30.
<code>orientation</code>	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 <i>Orientation</i> section for more detail.
<code>lineend</code>	Line end style (round, butt, square).
<code>linejoin</code>	Line join style (round, mitre, bevel).
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<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. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<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. <code>annotation_borders()</code> .

Value

XLayer or YLayer object to be added to a ggplot object

Aesthetics

`geom_xsidehistogram` uses the same aesthetics as `geom_xsidebar()`

Examples

```
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	<i>Side label</i>
-----------------	-------------------

Description

The [xside](#) and [yside](#) variants of [geom_label](#).

Usage

```
geom_xsidelabel(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "nudge",  
  ...,  
  parse = FALSE,  
  label.padding = unit(0.25, "lines"),  
  label.r = unit(0.15, "lines"),  
  border.colour = NULL,  
  border.color = NULL,  
  text.colour = NULL,  
  text.color = NULL,  
  size.unit = "mm",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  label.size = lifecycle::deprecated()  
)
```

```
geom_ysidelabel(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "nudge",  
  ...,  
  parse = FALSE,  
  label.padding = unit(0.25, "lines"),  
  label.r = unit(0.15, "lines"),  
  border.colour = NULL,  
  border.color = NULL,  
  text.colour = NULL,  
  text.color = NULL,  
  size.unit = "mm",  
  na.rm = FALSE,
```

```

    show.legend = NA,
    inherit.aes = TRUE,
    label.size = lifecycle::deprecated()
  )

```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as <code>"jitter"</code>. • For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to layer()'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the

available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the stat part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as [key glyphs](#), to change the display of the layer in the legend.

<code>parse</code>	If TRUE, the labels will be parsed into expressions and displayed as described in <code>?plotmath</code> .
<code>label.padding</code>	Amount of padding around label. Defaults to 0.25 lines.
<code>label.r</code>	Radius of rounded corners. Defaults to 0.15 lines.
<code>border.colour</code> , <code>border.color</code>	Colour of label border. When NULL (default), the <code>colour</code> aesthetic determines the colour of the label border. <code>border.color</code> is an alias for <code>border.colour</code> .
<code>text.colour</code> , <code>text.color</code>	Colour of the text. When NULL (default), the <code>colour</code> aesthetic determines the colour of the text. <code>text.color</code> is an alias for <code>text.colour</code> .
<code>size.unit</code>	How the size aesthetic is interpreted: as millimetres ("mm", default), points ("pt"), centimetres ("cm"), inches ("in"), or picas ("pc").
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<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. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<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. <code>annotation_borders()</code> .
<code>label.size</code>	[Deprecated] Replaced by the <code>linewidth</code> aesthetic. Size of label border, in mm.

Value

XLayer or YLayer object to be added to a ggplot object

geom_xsideline	<i>Side line plot</i>
----------------	-----------------------

Description

The [xside](#) and [yside](#) of [geom_line](#). The [xside](#) and [yside](#) variants of [geom_path](#)

Usage

```
geom_xsideline(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  orientation = "x",  
  arrow = NULL,  
  arrow.fill = NULL,  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 10,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_ysideline(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  orientation = "y",  
  arrow = NULL,  
  arrow.fill = NULL,  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 10,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_xsidepath(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",
```



```

    position = "identity",
    ...,
    arrow = NULL,
    arrow.fill = NULL,
    lineend = "butt",
    linejoin = "round",
    linemitre = 10,
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
  )

geom_ysidepath(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A <code>Stat</code> ggproto subclass, for example <code>StatCount</code>.

	<ul style="list-style-type: none"> • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through ... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
orientation	<p>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 <i>Orientation</i> section for more detail.</p>
arrow	<p>Arrow specification, as created by <code>grid::arrow()</code>.</p>
arrow.fill	<p>fill colour to use for the arrow head (if closed). NULL means use colour aesthetic.</p>
lineend	<p>Line end style (round, butt, square).</p>

linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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. annotation_borders() .

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()
```

geom_xsidepoint	<i>Side Points</i>
-----------------	--------------------

Description

The ggside variants of [geom_point](#) is [geom_xsidepoint\(\)](#) and [geom_ysidepoint\(\)](#). Both variants inherit from [geom_point](#), thus the only difference is where the data is plotted. The xside variant will plot data along the x-axis, while the yside variant will plot data along the y-axis.

Usage

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

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

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as <code>"jitter"</code>. • For more information and other ways to specify the position, see the layer position documentation.

...	<p>Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through ... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<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. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<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. <code>annotation_borders()</code> .

Value

XLayer or YLayer object to be added to a ggplot object

Examples

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

geom_xsidesegment	Side line Segments
-------------------	--------------------

Description

The [xside](#) and [yside](#) of [geom_segment](#).

Usage

```
geom_xsidesegment(  
  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_ysidesegment(  
  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() . If specified and <code>inherit.aes = TRUE</code> (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 <code>NULL</code> , 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. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A Stat ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as "count".
- For more information and other ways to specify the stat, see the [layer stat](#) documentation.

position

A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as "jitter".
- For more information and other ways to specify the position, see the [layer position](#) documentation.

...

Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.

- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as [key glyphs](#), to change the display of the layer in the legend.

<code>arrow</code>	specification for arrow heads, as created by <code>grid::arrow()</code> .
<code>arrow.fill</code>	fill colour to use for the arrow head (if closed). NULL means use colour aesthetic.
<code>lineend</code>	Line end style (round, butt, square).
<code>linejoin</code>	Line join style (round, mitre, bevel).
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<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. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<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. <code>annotation_borders()</code> .

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
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)
lvls <- rownames(mat)[hc_x$order]
df1[["colclar"]] <- factor(df1[["colclar"]], levels = lvls)
dendrox <- 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 = dendrox$segments, aes(x = x, y = y, xend = xend, yend = yend))
```

geom_xsidetext

Side text

Description

The [xside](#) and [yside](#) variants of [geom_text](#).

Usage

```
geom_xsidetext(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "nudge",
  ...,
  parse = FALSE,
  check_overlap = FALSE,
  size.unit = "mm",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

```
geom_ysidetext(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "nudge",
  ...,
  parse = FALSE,
  check_overlap = FALSE,
  size.unit = "mm",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A <code>Stat</code> ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as <code>"jitter"</code>. • For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer.

An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.

- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as [key glyphs](#), to change the display of the layer in the legend.

<code>parse</code>	If TRUE, the labels will be parsed into expressions and displayed as described in <code>?plotmath</code> .
<code>check_overlap</code>	If TRUE, text that overlaps previous text in the same layer will not be plotted. <code>check_overlap</code> happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling <code>geom_text()</code> . Note that this argument is not supported by <code>geom_label()</code> .
<code>size.unit</code>	How the size aesthetic is interpreted: as millimetres ("mm", default), points ("pt"), centimetres ("cm"), inches ("in"), or picas ("pc").
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<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. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<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. annotation_borders() .

Value

XLayer or YLayer object to be added to a ggplot object

<code>geom_xsidetile</code>	<i>Side tile plot</i>
-----------------------------	-----------------------

Description

The [xside](#) and [yside](#) variants of [geom_tile](#)

Usage

```
geom_xsidetile(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  lineend = "butt",
  linejoin = "mitre",
```

```

na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE
)

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

```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position.

- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the [layer position](#) documentation.

... Other arguments passed on to [layer\(\)](#)'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept.
- The key_glyph argument of [layer\(\)](#) may also be passed on through ... This can be one of the functions described as [key glyphs](#), to change the display of the layer in the legend.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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. annotation_borders() .

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
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))
```

geom_xsideviolin

Side Violin plots

Description

The [xside](#) and [yside](#) variants of [geom_violin](#)

Usage

```
geom_xsideviolin(
  mapping = NULL,
  data = NULL,
  stat = "ydensity",
  position = "dodge",
  ...,
  trim = TRUE,
  bounds = c(-Inf, Inf),
  quantile.colour = NULL,
  quantile.color = NULL,
  quantile.linetype = 0L,
  quantile.linewidth = NULL,
  scale = "area",
  na.rm = FALSE,
  orientation = "x",
  show.legend = NA,
```

```

    inherit.aes = TRUE,
    draw_quantiles = lifecycle::deprecated()
  )

  geom_ysideviolin(
    mapping = NULL,
    data = NULL,
    stat = "ydensity",
    position = "dodge",
    ...,
    trim = TRUE,
    bounds = c(-Inf, Inf),
    quantile.colour = NULL,
    quantile.color = NULL,
    quantile.linetype = 0L,
    quantile.linewidth = NULL,
    scale = "area",
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE,
    orientation = "y",
    draw_quantiles = lifecycle::deprecated()
  )

```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	Use to override the default connection between geom_violin() and stat_ydensity() .
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:</p> <ul style="list-style-type: none"> • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use position_jitter(), give the position as <code>"jitter"</code>.

	<ul style="list-style-type: none"> For more information and other ways to specify the position, see the layer position documentation.
...	<p>Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through ... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
<code>trim</code>	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don't trim the tails.
<code>bounds</code>	Known lower and upper bounds for estimated data. Default <code>c(-Inf, Inf)</code> means that there are no (finite) bounds. If any bound is finite, boundary effect of default density estimation will be corrected by reflecting tails outside bounds around their closest edge. Data points outside of bounds are removed with a warning
<code>quantile.colour,</code> <code>quantile.linetype</code>	<div> <div><code>quantile.color,</code></div> <div><code>quantile.linewidth,</code></div> </div> <p>Default aesthetics for the quantile lines. Set to NULL to inherit from the data's aesthetics. By default, quantile lines are hidden and can be turned on by changing <code>quantile.linetype</code>.</p>
<code>scale</code>	if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>orientation</code>	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 <code>orientation</code> to either "x" or "y". See the <i>Orientation</i> section for more detail.
<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. It

can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

`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. [annotation_borders\(\)](#).

`draw_quantiles` **[Deprecated]** Previous specification of drawing quantiles.

Value

XLayer or YLayer object to be added to a ggplot object

See Also

[geom_*sideboxplot](#)

Examples

```
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

*ggside options***Description**

Set characteristics of side panels

Usage

```
ggside(
  x.pos = NULL,
  y.pos = NULL,
  scales = NULL,
  collapse = NULL,
  draw_x_on = NULL,
  draw_y_on = NULL,
  strip = NULL,
  respect_side_labels = NULL
)
```

Arguments

<code>x.pos</code>	x side panel can either take "top" or "bottom"
<code>y.pos</code>	y side panel can either take "right" or "left"
<code>scales</code>	Determines side panel's unaligned axis scale. Inputs are similar to <code>facet_* scales</code> 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.
<code>collapse</code>	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.
<code>draw_x_on, draw_y_on</code>	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.
<code>strip</code>	Determines if the strip should be rendered on the main plot or on their default locations. Only has an effect on <code>facet_grid</code> .
<code>respect_side_labels</code>	Valid arguments are "default", "x", "y", "all", and "none" Indicates if panel spacing should respect the axis labels. The default is to respect side panel labels except when xside labels are on the same side as the yside panel. Note: setting this parameter to "x" is to "respect the labels of the xside panel" and consequently the yside labels, if present, are not respected.

Value

a object of class 'ggside_options' or to be added to a ggplot

See Also

For more information regarding the ggside api: see [xside](#) or [yside](#)

ggside-deprecated	<i>Deprecated Functions</i>
-------------------	-----------------------------

Description

The following functions have been deprecated.

as_ggsideFacet <- [ggside_facet](#) as_ggsideCoord <- [ggside_coord](#) is.ggside <- [is_ggside](#) is.ggside_layer <- [is_ggside_layer](#) is.ggside_options <- [is_ggside_options](#) is.ggside_scale <- [is_ggside_scale](#)

ggside-scales	<i>Specifying side scales</i>
---------------	-------------------------------

Description

The [xside](#) and [yside](#) variants of [geoms](#) are plotted along the x-axis and y-axis respectively of their main panel's data mapping. The positional scale here is shared between the main panel and the side panel. The related positional scale type of the side panel, i.e. the y axis of the xside panel (xsidey) or the x axis of the yside panel (ysidex), is determined automatically by ggplot2 default scales. However, you can override this by using the [continuous](#) or [discrete](#) variants within ggside. This allows the user to select the scale type or transform most appropriate for their side panels.

ggside-scales-binned	<i>Position scales for binning continuous data ggside scales</i>
----------------------	--

Description

The [xside](#) and [yside](#) variants of [scale_x_binned/scale_y_binned](#). [scale_xsidey_binned](#) enables better control on how the y-axis is rendered on the xside panel and [scale_ysidex_binned](#) enables better control on how the x-axis is rendered on the yside panel.

Usage

```
scale_xsidey_binned(
  name = waiver(),
  n.breaks = 10,
  nice.breaks = TRUE,
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  oob = squish,
  na.value = NA_real_,
  right = TRUE,
  show.limits = FALSE,
  transform = "identity",
  guide = waiver(),
  position = "left"
)
```

```
scale_ysidex_binned(
  name = waiver(),
  n.breaks = 10,
  nice.breaks = TRUE,
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  oob = squish,
  na.value = NA_real_,
  right = TRUE,
  show.limits = FALSE,
  transform = "identity",
  guide = waiver(),
  position = "bottom"
)
```

Arguments

name	The name of the scale. Used as the axis or legend title. If <code>waiver()</code> , the default, the name of the scale is taken from the first mapping used for that aesthetic. If <code>NULL</code> , the legend title will be omitted.
n.breaks	The number of break points to create if breaks are not given directly.
nice.breaks	Logical. Should breaks be attempted placed at nice values instead of exactly evenly spaced between the limits. If <code>TRUE</code> (default) the scale will ask the transformation object to create breaks, and this may result in a different number of breaks than requested. Ignored if breaks are given explicitly.
breaks	One of: <ul style="list-style-type: none"> • <code>NULL</code> for no breaks

	<ul style="list-style-type: none"> • <code>waiver()</code> 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 <code>scales::extended_breaks()</code>). Note that for position scales, limits are provided after scale expansion. Also accepts rlang lambda function notation.
<code>labels</code>	<p>One of the options below. Please note that when <code>labels</code> is a vector, it is highly recommended to also set the <code>breaks</code> argument as a vector to protect against unintended mismatches.</p> <ul style="list-style-type: none"> • <code>NULL</code> for no labels • <code>waiver()</code> for the default labels computed by the transformation object • A character vector giving labels (must be same length as <code>breaks</code>) • An expression vector (must be the same length as <code>breaks</code>). See <code>?plotmath</code> for details. • A function that takes the <code>breaks</code> as input and returns labels as output. Also accepts rlang lambda function notation.
<code>limits</code>	<p>One of:</p> <ul style="list-style-type: none"> • <code>NULL</code> to use the default scale range • A numeric vector of length two providing limits of the scale. Use <code>NA</code> 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 <code>limit</code> argument in the coordinate system (see coord_cartesian()).
<code>expand</code>	<p>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 <code>expand</code> 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.</p>
<code>oob</code>	<p>One of:</p> <ul style="list-style-type: none"> • Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang lambda function notation. • The default (<code>scales::squish()</code>) squishes out of bounds values into range. • <code>scales::censor</code> for replacing out of bounds values with <code>NA</code>. • <code>scales::squish_infinite()</code> for squishing infinite values into range.
<code>na.value</code>	<p>Missing values will be replaced with this value.</p>
<code>right</code>	<p>Should the intervals be closed on the right (<code>TRUE</code>, default) or should the intervals be closed on the left (<code>FALSE</code>)? 'Closed on the right' means that values at break positions are part of the lower bin (open on the left), whereas they are part of the upper bin when intervals are closed on the left (open on the right).</p>
<code>show.limits</code>	<p>should the limits of the scale appear as ticks</p>

transform	<p>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".</p> <p>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 <code>transform_<name></code>. If transformations require arguments, you can call them from the scales package, e.g. <code>scales::transform_boxcox(p = 2)</code>. You can create your own transformation with <code>scales::new_transform()</code>.</p>
guide	A function used to create a guide or its name. See <code>guides()</code> 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::ScaleBinnedPosition`

Examples

```
ggplot(iris, aes(Sepal.Width, Sepal.Length)) +
  geom_point() +
  geom_xsidepoint(aes(y = Petal.Width, xcolour = Petal.Length)) +
  scale_xsidey_binned(n.breaks = 4) +
  scale_colour_steps(aesthetics = "xcolour", guide = guide_colorbar(available_aes = "xcolour")) +
  theme(ggside.panel.scale.x = .3)
```

ggside-scales-continuous

Position scales for continuous data ggside scales

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_ysidex_continuous` enables better control on how the x-axis is rendered on the yside panel.

Usage

```
scale_xsidey_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
```

```

    expand = waiver(),
    oob = scales::censor,
    na.value = NA_real_,
    transform = "identity",
    guide = waiver(),
    position = "left",
    sec.axis = waiver()
)

scale_xsidey_log10(...)

scale_xsidey_reverse(...)

scale_xsidey_sqrt(...)

scale_ysidex_log10(...)

scale_ysidex_reverse(...)

scale_ysidex_sqrt(...)

scale_ysidex_log10(...)

scale_ysidex_reverse(...)

scale_ysidex_sqrt(...)

```

Arguments

name	The name of the scale. Used as the axis or legend title. If <code>waiver()</code> , the default, the name of the scale is taken from the first mapping used for that aesthetic. If <code>NULL</code> , the legend title will be omitted.
breaks	One of: <ul style="list-style-type: none"> • <code>NULL</code> for no breaks • <code>waiver()</code> 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 <code>scales::extended_breaks()</code>). Note that for position scales, limits are provided after scale expansion. Also accepts rlang lambda function notation.
minor_breaks	One of: <ul style="list-style-type: none"> • <code>NULL</code> for no minor breaks • <code>waiver()</code> for the default breaks (none for discrete, one minor break between each major break for continuous) • A numeric vector of positions

	<ul style="list-style-type: none"> • A function that given the limits returns a vector of minor breaks. Also accepts rlang <code>lambda</code> function notation. When the function has two arguments, it will be given the limits and major break positions.
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 <code>breaks = waiver()</code> . Use <code>NULL</code> to use the default number of breaks given by the transformation.
labels	<p>One of the options below. Please note that when <code>labels</code> is a vector, it is highly recommended to also set the <code>breaks</code> argument as a vector to protect against unintended mismatches.</p> <ul style="list-style-type: none"> • <code>NULL</code> for no labels • <code>waiver()</code> for the default labels computed by the transformation object • A character vector giving labels (must be same length as <code>breaks</code>) • An expression vector (must be the same length as <code>breaks</code>). See <code>?plotmath</code> for details. • A function that takes the <code>breaks</code> as input and returns labels as output. Also accepts rlang <code>lambda</code> function notation.
limits	<p>One of:</p> <ul style="list-style-type: none"> • <code>NULL</code> to use the default scale range • A numeric vector of length two providing limits of the scale. Use <code>NA</code> to refer to the existing minimum or maximum • A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang <code>lambda</code> function notation. Note that setting limits on positional scales will remove data outside of the limits. If the purpose is to zoom, use the <code>limit</code> argument in the coordinate system (see <code>coord_cartesian()</code>).
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 <code>expansion()</code> to generate the values for the <code>expand</code> 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	<p>One of:</p> <ul style="list-style-type: none"> • Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang <code>lambda</code> function notation. • The default (<code>scales::censor()</code>) replaces out of bounds values with <code>NA</code>. • <code>scales::squish()</code> for squishing out of bounds values into range. • <code>scales::squish_infinite()</code> for squishing infinite values into range.
na.value	Missing values will be replaced with this value.
transform	<p>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".</p> <p>A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the <code>scales</code></p>

	package, and are called <code>transform_<name></code> . If transformations require arguments, you can call them from the scales package, e.g. <code>scales::transform_boxcox(p = 2)</code> . You can create your own transformation with <code>scales::new_transform()</code> .
<code>guide</code>	A function used to create a guide or its name. See <code>guides()</code> for more information.
<code>position</code>	For position scales, The position of the axis. <code>left</code> or <code>right</code> for y axes, <code>top</code> or <code>bottom</code> for x axes.
<code>sec.axis</code>	<code>sec_axis()</code> is used to specify a secondary axis.
<code>...</code>	Other arguments passed on to <code>scale_(ylx)side(xly)_continuous()</code>

Value

ggside_scale object inheriting from `ggplot2::ScaleContinuousPosition`

Examples

```
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_xsidedensity(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_*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.
```

ggside-scales-discrete

Position scales for discrete data ggside scales

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

...

Arguments passed on to `discrete_scale`**breaks** One of:

- NULL for no breaks
- `waiver()` for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang `lambda` function notation.

limits One of:

- NULL to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang `lambda` function notation.

drop Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

na.translate Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

na.value If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where NA is always placed at the far right.

aesthetics The names of the aesthetics that this scale works with.

minor_breaks One of:

- NULL for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang `lambda` function notation. When the function has two arguments, it will be given the limits and major break positions.

labels One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- 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)
- An expression vector (must be the same length as breaks). See `?plot-math` for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang `lambda` function notation.

call The call used to construct the scale for reporting messages.

	super	The super class to use for the constructed scale
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

```
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
```

ggside_coord

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

```
ggside_coord(coord)

## Default S3 method:
ggside_coord(coord)

## S3 method for class 'CoordCartesian'
ggside_coord(coord)

## S3 method for class 'CoordSide'
ggside_coord(coord)

## S3 method for class 'CoordTrans'
ggside_coord(coord)

## S3 method for class 'CoordFixed'
ggside_coord(coord)
```

Arguments

coord coord ggproto Object to replace

ggside_geom	<i>ggside geom constructor</i>
-------------	--------------------------------

Description

utility function to make a ggside Geom

Usage

```
ggside_geom(class_name = NULL, geom = NULL, side = NULL, ...)
```

Arguments

class_name	New class name for the ggproto object
geom	The Geom ggproto to inherit from
side	should the resulting object be configured for x or y
...	additional members to add to the ggproto class.

ggside_layer	<i>New ggside layer</i>
--------------	-------------------------

Description

utility function to make a ggside layer compatible with ggside internals

Usage

```
ggside_layer(
  geom = NULL,
  stat = NULL,
  data = NULL,
  mapping = NULL,
  position = NULL,
  params = list(),
  inherit.aes = TRUE,
  check.aes = TRUE,
  check.param = TRUE,
  show.legend = NA,
  key_glyph = NULL,
  side = NULL
)

as_ggside_layer(layer, side)
```

Arguments

geom	<p>The geometric object to use to display the data for this layer. When using a <code>stat_*()</code> function to construct a layer, the <code>geom</code> argument can be used to override the default coupling between stats and geoms. The <code>geom</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A Geom ggproto subclass, for example <code>GeomPoint</code>. • A string naming the geom. To give the geom as a string, strip the function name of the <code>geom_</code> prefix. For example, to use <code>geom_point()</code>, give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count".

	<ul style="list-style-type: none"> For more information and other ways to specify the stat, see the layer stat documentation.
data	<p>The data to be displayed in this layer. There are three options:</p> <p>If NULL, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
mapping	Set of aesthetic mappings created by <code>aes()</code> . If specified and <code>inherit.aes = TRUE</code> (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.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:</p> <ul style="list-style-type: none"> The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". For more information and other ways to specify the position, see the layer position documentation.
params	Additional parameters to the geom and stat.
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>annotation_borders()</code> .
check.aes, check.param	If TRUE, the default, will check that supplied parameters and aesthetics are understood by the geom or stat. Use FALSE to suppress the checks.
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
key_glyph	A legend key drawing function or a string providing the function name minus the <code>draw_key_</code> prefix. See draw_key for details.
side	should the resulting <code>ggplot2_layer</code> be configured for x or y side
layer	a <code>LayerInstance</code> object made from layer

ggside_layout	<i>Construct ggside layout</i>
---------------	--------------------------------

Description

Creates a new layout object required for ggside functionality

Usage

```
ggside_layout(layout)
```

Arguments

layout	a ggproto Layout object
--------	-------------------------

is_ggside	<i>Check ggside objects</i>
-----------	-----------------------------

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

parse_side_aes	<i>Extending base ggproto classes for ggside</i>
----------------	--

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

```
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](#)

position_rescale	<i>Rescale x or y onto new range in margin</i>
------------------	--

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

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

```
position_yrescale(
  rescale = "y",
```



```

    midpoint = NULL,
    range = NULL,
    location = NULL,
    instance = NULL
  )

  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	<i>Scales for the *colour aesthetics</i>
---------------	--

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	<i>scale_ycolour_hue</i>
-------------------	--------------------------

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	<i>scale_yfill_hue</i>
-----------------	------------------------

Description

- scale_yfill_hue
- scale_yfill_manual
- scale_yfill_gradient
- scale_yfill_discrete
- scale_yfill_continuous

stat_summarise	<i>Summarise by grouping variable</i>
----------------	---------------------------------------

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() . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
geom	<p>The geometric object to use to display the data for this layer. When using a <code>stat_*()</code> function to construct a layer, the <code>geom</code> argument can be used to override the default coupling between stats and geoms. The <code>geom</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A <code>Geom</code> ggproto subclass, for example <code>GeomPoint</code>. • A string naming the geom. To give the geom as a string, strip the function name of the <code>geom_</code> prefix. For example, to use <code>geom_point()</code>, give the geom as <code>"point"</code>.

	<ul style="list-style-type: none"> For more information and other ways to specify the geom, see the layer geom documentation.
position	<p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:</p> <ul style="list-style-type: none"> The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". For more information and other ways to specify the position, see the layer position documentation.
...	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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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. annotation_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(tidyr)
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	<i>ggside custom themes</i>
-------------------	-----------------------------

Description

Theme elements to help customize the look and feel of [ggside](#)'s side panels.

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_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.
ggside.axis.text, ggside.axis.text.x, ggside.axis.text.y, ggside.axis.text.x.top, ggside.axis.text.x.bo
ggside.axis.line, ggside.axis.line.x, ggside.axis.line.y, ggside.axis.line.x.top, ggside.axis.line.x.bo
ggside.axis.ticks, ggside.axis.ticks.x, ggside.axis.ticks.y, ggside.axis.ticks.x.top, ggside.axis.ticks
ggside.axis.ticks.length, ggside.axis.ticks.length.x, ggside.axis.ticks.length.y, ggside.axis.ticks.le
ggside.axis.minor.ticks, ggside.axis.minor.ticks.x, ggside.axis.minor.ticks.y, ggside.axis.minor.ticks.
ggside.axis.minor.ticks.length, ggside.axis.minor.ticks.length.x, ggside.axis.minor.ticks.length.y, gg

Examples

```
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"))
```

xside	<i>The xside geometries</i>
-------	-----------------------------

Description

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

Value

geom_xside* return a XLayer object to be added to a ggplot

New Aesthetics

All `xside` 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 `xside` 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 `xside` geometry without conflicts. See more information in `vignette("ggside")`.

Exported Geometries

The following are the `xside` 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

[yside](#)

yside

The yside geometries

Description

`yside` refers to the api of `ggside`. 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("ggside")`.

#' @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**
 - class_definitions, 5
 - parse_side_aes, 64
 - position_rescale, 64
 - stat_summarise, 67
- aes(), 7, 9, 14, 17, 20, 23, 26, 30, 33, 36, 38, 41, 44, 47, 62, 68
- annotation_borders(), 11, 16, 19, 21, 25, 28, 31, 35, 37, 40, 43, 45, 49, 62, 69
- as_ggside, 3
- as_ggside_layer (ggside_layer), 61
- as_ggsideCoord (ggside-deprecated), 51
- as_ggsideFacet (ggside-deprecated), 51
- check_scales_collapse, 4
- class_definitions, 5
- class_ggside (class_definitions), 5
- class_ggside_layer (class_definitions), 5
- class_ggside_opt (class_definitions), 5
- class_ggside_scale (class_definitions), 5
- continuous, 51
- coord_cartesian(), 53, 56
- discrete, 51
- discrete_scale, 58
- draw_key, 62
- expansion(), 53, 56, 59
- FacetGrid, 4
- FacetNull, 4
- FacetWrap, 4
- fortify(), 7, 9, 14, 18, 20, 27, 30, 33, 36, 39, 42, 44, 47, 62, 68
- geom_abline, 5
- geom_abline (geom_xsideabline), 5
- geom_freqpoly (geom_xsidefreqpoly), 19
- geom_hline, 5
- geom_hline (geom_xsideabline), 5
- geom_sidebar (geom_xsidebar), 8
- geom_sidebar(), 28
- geom_sideboxplot, 49
- geom_sideboxplot (geom_xsideboxplot), 12
- geom_sidedensity (geom_xsidedensity), 17
- geom_sidefunction (geom_xsidefunction), 22
- geom_sidehistogram (geom_xsidehistogram), 25
- geom_sidelabel (geom_xsidelabel), 29
- geom_sideline (geom_xsideline), 32
- geom_sidepoint (geom_xsidepoint), 35
- geom_sidesegment (geom_xsidesegment), 38
- geom_sidetext (geom_xsidetext), 41
- geom_sidetile (geom_xsidetile), 43
- geom_sideviolin, 16
- geom_sideviolin (geom_xsideviolin), 46
- geom_vline, 5
- geom_vline (geom_xsideabline), 5
- geom_abline, 5
- geom_bar, 8
- geom_boxplot, 12
- geom_col, 8
- geom_density, 17
- geom_freqpoly, 19
- geom_function, 22
- geom_histogram, 26
- geom_hline, 5
- geom_label, 29
- geom_line, 32
- geom_path, 32
- geom_point, 35
- geom_segment, 38
- geom_text, 41

geom_tile, 43
 geom_violin, 46
 geom_vline, 5
 geom_xsideabline, 5
 geom_xsidebar, 8, 8, 73
 geom_xsideboxplot, 12, 12, 73
 geom_xsidecol, 8, 73
 geom_xsidecol (geom_xsidebar), 8
 geom_xsidedensity, 17, 17, 73
 geom_xsidefreqpoly, 19, 19, 73
 geom_xsidefunction, 22
 geom_xsidehistogram, 11, 25, 26, 73
 geom_xsidehline (geom_xsideabline), 5
 geom_xsidelabel, 29
 geom_xsideline, 32, 73
 geom_xsidepath, 73
 geom_xsidepath (geom_xsideline), 32
 geom_xsidepoint, 35, 73
 geom_xsidepoint(), 35
 geom_xsidesegment, 38
 geom_xsidetext, 41, 73
 geom_xsidetile, 43, 73
 geom_xsideviolin, 46, 73
 geom_xsidevline (geom_xsideabline), 5
 geom_ysideabline (geom_xsideabline), 5
 geom_ysidebar, 8, 74
 geom_ysidebar (geom_xsidebar), 8
 geom_ysideboxplot, 12, 74
 geom_ysideboxplot (geom_xsideboxplot), 12
 geom_ysidecol, 8, 74
 geom_ysidecol (geom_xsidebar), 8
 geom_ysidedensity, 17, 74
 geom_ysidedensity (geom_xsidedensity), 17
 geom_ysidefreqpoly, 19, 74
 geom_ysidefreqpoly (geom_xsidefreqpoly), 19
 geom_ysidefunction (geom_xsidefunction), 22
 geom_ysidehistogram, 11, 26, 74
 geom_ysidehistogram (geom_xsidehistogram), 25
 geom_ysidehline (geom_xsideabline), 5
 geom_ysidelabel (geom_xsidelabel), 29
 geom_ysideline, 74
 geom_ysideline (geom_xsideline), 32
 geom_ysidepath, 74
 geom_ysidepath (geom_xsideline), 32
 geom_ysidepoint, 74
 geom_ysidepoint (geom_xsidepoint), 35
 geom_ysidepoint(), 35
 geom_ysidesegment (geom_xsidesegment), 38
 geom_ysidetext, 74
 geom_ysidetext (geom_xsidetext), 41
 geom_ysidetile, 74
 geom_ysidetile (geom_xsidetile), 43
 geom_ysideviolin, 74
 geom_ysideviolin (geom_xsideviolin), 46
 geom_ysidevline (geom_xsideabline), 5
 geoms, 51
 GeomXsideabline (parse_side_aes), 64
 GeomXsidebar (parse_side_aes), 64
 GeomXsideboxplot (parse_side_aes), 64
 GeomXsidecol (parse_side_aes), 64
 GeomXsidedensity (parse_side_aes), 64
 GeomXsidefunction (parse_side_aes), 64
 GeomXsidehline (parse_side_aes), 64
 GeomXsidelabel (parse_side_aes), 64
 GeomXsideline (parse_side_aes), 64
 GeomXsidepath (parse_side_aes), 64
 GeomXsidepoint (parse_side_aes), 64
 GeomXsidesegment (parse_side_aes), 64
 GeomXsidetext (parse_side_aes), 64
 GeomXsidetile (parse_side_aes), 64
 GeomXsideviolin (parse_side_aes), 64
 GeomXsidevline (parse_side_aes), 64
 GeomYsideabline (parse_side_aes), 64
 GeomYsidebar (parse_side_aes), 64
 GeomYsideboxplot (parse_side_aes), 64
 GeomYsidecol (parse_side_aes), 64
 GeomYsidedensity (parse_side_aes), 64
 GeomYsidefunction (parse_side_aes), 64
 GeomYsidehline (parse_side_aes), 64
 GeomYsidelabel (parse_side_aes), 64
 GeomYsideline (parse_side_aes), 64
 GeomYsidepath (parse_side_aes), 64
 GeomYsidepoint (parse_side_aes), 64
 GeomYsidesegment (parse_side_aes), 64
 GeomYsidetext (parse_side_aes), 64
 GeomYsidetile (parse_side_aes), 64
 GeomYsideviolin (parse_side_aes), 64
 GeomYsidevline (parse_side_aes), 64
 ggplot(), 7, 9, 14, 18, 20, 26, 30, 33, 36, 38, 42, 44, 47, 62, 68

- ggplot2, [4](#), [73](#), [74](#)
- ggplot2's class_ggplot, [5](#)
- ggproto, [64](#)
- ggside, [50](#), [70](#)
- ggside-deprecated, [51](#)
- ggside-ggproto-facets
 - (check_scales_collapse), [4](#)
- ggside-ggproto-geoms (parse_side_aes), [64](#)
- ggside-layers, [5](#)
- ggside-options, [5](#)
- ggside-options (ggside), [50](#)
- ggside-scales, [5](#), [51](#)
- ggside-scales-binned, [51](#)
- ggside-scales-continuous, [54](#)
- ggside-scales-discrete, [57](#)
- ggside-theme (theme_ggside_grey), [70](#)
- ggside_coord, [51](#), [59](#)
- ggside_facet, [51](#)
- ggside_facet (check_scales_collapse), [4](#)
- ggside_geom, [60](#)
- ggside_layer, [61](#)
- ggside_layout, [63](#)
- ggside_options (ggside), [50](#)
- ggside_scales (ggside-scales), [51](#)
- grid::arrow(), [24](#), [34](#), [40](#)
- guides(), [54](#), [57](#), [59](#)

- is.ggside (ggside-deprecated), [51](#)
- is.ggside_layer (ggside-deprecated), [51](#)
- is.ggside_options (ggside-deprecated), [51](#)
- is.ggside_scale (ggside-deprecated), [51](#)
- is_ggside, [51](#), [63](#)
- is_ggside_layer, [51](#)
- is_ggside_layer (is_ggside), [63](#)
- is_ggside_options, [51](#)
- is_ggside_options (is_ggside), [63](#)
- is_ggside_scale, [51](#)
- is_ggside_scale (is_ggside), [63](#)

- key glyphs, [7](#), [10](#), [15](#), [18](#), [21](#), [24](#), [28](#), [31](#), [34](#), [37](#), [40](#), [43](#), [45](#), [48](#)

- lambda, [53](#), [55](#), [56](#), [58](#)
- layer, [62](#), [64](#), [69](#)
- layer geom, [25](#), [61](#), [69](#)
- layer position, [8](#), [10](#), [14](#), [18](#), [21](#), [24](#), [27](#), [30](#), [34](#), [36](#), [39](#), [42](#), [45](#), [48](#), [62](#), [69](#)
- layer stat, [10](#), [14](#), [21](#), [23](#), [27](#), [30](#), [34](#), [36](#), [39](#), [42](#), [44](#), [62](#)
- layer(), [7](#), [10](#), [14](#), [15](#), [18](#), [21](#), [24](#), [27](#), [28](#), [30](#), [31](#), [34](#), [37](#), [39](#), [40](#), [42](#), [43](#), [45](#), [48](#)
- length, [69](#)

- parse_side_aes, [64](#)
- position_rescale, [64](#)
- position_xrescale (position_rescale), [64](#)
- position_yrescale (position_rescale), [64](#)
- PositionRescale (position_rescale), [64](#)

- rlang::as_function(), [25](#)

- scale_x_binned, [51](#)
- scale_x_continuous, [54](#)
- scale_x_discrete, [57](#)
- scale_xcolor (scale_xcolour), [65](#)
- scale_xcolor_continuous (scale_xcolour), [65](#)
- scale_xcolor_discrete (scale_xcolour), [65](#)
- scale_xcolor_gradientn (scale_xcolour), [65](#)
- scale_xcolor_manual (scale_xcolour), [65](#)
- scale_xcolour, [65](#)
- scale_xcolour_continuous (scale_xcolour), [65](#)
- scale_xcolour_discrete (scale_xcolour), [65](#)
- scale_xcolour_gradient (scale_xcolour), [65](#)
- scale_xcolour_gradientn (scale_xcolour), [65](#)
- scale_xcolour_hue (scale_xcolour), [65](#)
- scale_xcolour_manual (scale_xcolour), [65](#)
- scale_xfill, [66](#)
- scale_xfill_continuous (scale_xfill), [66](#)
- scale_xfill_discrete (scale_xfill), [66](#)
- scale_xfill_gradient (scale_xfill), [66](#)
- scale_xfill_gradientn (scale_xfill), [66](#)
- scale_xfill_hue (scale_xfill), [66](#)
- scale_xfill_manual (scale_xfill), [66](#)
- scale_xsidey_binned, [51](#)
- scale_xsidey_binned (ggside-scales-binned), [51](#)
- scale_xsidey_continuous, [54](#)
- scale_xsidey_continuous (ggside-scales-continuous), [54](#)

- scale_xsidey_discrete, [57](#)
- scale_xsidey_discrete
 - (ggside-scales-discrete), [57](#)
- scale_xsidey_log10
 - (ggside-scales-continuous), [54](#)
- scale_xsidey_reverse
 - (ggside-scales-continuous), [54](#)
- scale_xsidey_sqrt
 - (ggside-scales-continuous), [54](#)
- scale_y_binned, [51](#)
- scale_y_continuous, [54](#)
- scale_y_discrete, [57](#)
- scale_ycolor (scale_xcolour), [65](#)
- scale_ycolor_continuous
 - (scale_ycolour_hue), [67](#)
- scale_ycolor_discrete
 - (scale_ycolour_hue), [67](#)
- scale_ycolor_gradientn
 - (scale_ycolour_hue), [67](#)
- scale_ycolor_manual
 - (scale_ycolour_hue), [67](#)
- scale_ycolour (scale_xcolour), [65](#)
- scale_ycolour_continuous
 - (scale_ycolour_hue), [67](#)
- scale_ycolour_discrete
 - (scale_ycolour_hue), [67](#)
- scale_ycolour_gradient
 - (scale_ycolour_hue), [67](#)
- scale_ycolour_gradientn
 - (scale_ycolour_hue), [67](#)
- scale_ycolour_hue, [67](#)
- scale_ycolour_manual
 - (scale_ycolour_hue), [67](#)
- scale_yfill (scale_xfill), [66](#)
- scale_yfill_continuous
 - (scale_yfill_hue), [67](#)
- scale_yfill_discrete (scale_yfill_hue), [67](#)
- scale_yfill_gradient (scale_yfill_hue), [67](#)
- scale_yfill_gradientn (scale_xfill), [66](#)
- scale_yfill_hue, [67](#)
- scale_yfill_manual (scale_yfill_hue), [67](#)
- scale_ysidex_binned, [51](#)
- scale_ysidex_binned
 - (ggside-scales-binned), [51](#)
- scale_ysidex_continuous, [54](#)
- scale_ysidex_continuous
 - (ggside-scales-continuous), [54](#)
- scale_ysidex_discrete, [57](#)
- scale_ysidex_discrete
 - (ggside-scales-discrete), [57](#)
- scale_ysidex_log10
 - (ggside-scales-continuous), [54](#)
- scale_ysidex_reverse
 - (ggside-scales-continuous), [54](#)
- scale_ysidex_sqrt
 - (ggside-scales-continuous), [54](#)
- scales::censor, [53](#)
- scales::censor(), [56](#)
- scales::extended_breaks(), [53](#), [55](#)
- scales::new_transform(), [54](#), [57](#)
- scales::squish(), [53](#), [56](#)
- scales::squish_infinite(), [53](#), [56](#)
- sec_axis(), [57](#)
- sidePanelLayout
 - (check_scales_collapse), [4](#)
- stat_summarise, [67](#)
- stat_summarize (stat_summarise), [67](#)
- stat_xsidefunction
 - (geom_xsidefunction), [22](#)
- stat_ysidefunction
 - (geom_xsidefunction), [22](#)
- StatSummarise (stat_summarise), [67](#)
- StatSummarize (stat_summarise), [67](#)
- theme_ggside_bw (theme_ggside_grey), [70](#)
- theme_ggside_classic
 - (theme_ggside_grey), [70](#)
- theme_ggside_dark (theme_ggside_grey), [70](#)
- theme_ggside_gray (theme_ggside_grey), [70](#)
- theme_ggside_grey, [70](#)
- theme_ggside_light (theme_ggside_grey), [70](#)
- theme_ggside_linedraw
 - (theme_ggside_grey), [70](#)
- theme_ggside_minimal
 - (theme_ggside_grey), [70](#)
- theme_ggside_void (theme_ggside_grey), [70](#)
- theme_grey, [71](#)
- transformation object, [53](#), [55](#)
- xside, [5](#), [8](#), [12](#), [17](#), [19](#), [22](#), [26](#), [29](#), [32](#), [38](#), [41](#), [43](#), [46](#), [51](#), [54](#), [57](#), [72](#), [74](#)

yside, [5](#), [8](#), [12](#), [17](#), [19](#), [22](#), [26](#), [29](#), [32](#), [38](#), [41](#),
[43](#), [46](#), [51](#), [54](#), [57](#), [73](#), [73](#)