Package 'ggside'

July 22, 2025

Type Package

Title Side Grammar Graphics

Version 0.3.1

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 7.3.1

VignetteBuilder knitr

Depends ggplot2 (>= 3.5.0)

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

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

Config/testthat/edition 3

Collate 'z-depricated.R' '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' 'side-layout-.r' 'utils-side-coord.R'
'side-coord-cartesian.R' 'plot-construction.R' 'ggplot_add.R'
'add_gg.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' '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' 'zzz.R'

NeedsCompilation no

Author Justin Landis [aut, cre] (ORCID:
       <<https://orcid.org/0000-0001-5501-4934>>)

Repository CRAN

Date/Publication 2024-03-01 09:12:37 UTC

Contents

---

as_ggside   *Explicit conversion to ggside object*

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'
as_ggside(x, ggside = NULL, ...)
```

Arguments

x	an object to convert
...	unused argument
ggside	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

---

geom_xsideabline   *Side Reference lines*

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,
  ...,
  yintercept,
  na.rm = FALSE,
  show.legend = NA
)

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

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

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

Arguments

mapping	Set of aesthetic mappings created by [aes()]().
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
...	Other arguments passed on to [layer()](). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
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,
  width = NULL,
  na.rm = FALSE,
```

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

geom_ysidebar(
    mapping = NULL,
    data = NULL,
    stat = "count",
    position = "stack",
    ...,
    just = 0.5,
    width = NULL,
    na.rm = FALSE,
    orientation = "y",
    show.legend = NA,
    inherit.aes = TRUE
)

geom_xsidecol(
    mapping = NULL,
    data = NULL,
    position = "stack",
    ...,
    just = 0.5,
    width = NULL,
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
)

geom_ysidecol(
    mapping = NULL,
    data = NULL,
    position = "stack",
    ...,
    just = 0.5,
    width = NULL,
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE,
    orientation = "y"
)
```

Arguments

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

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

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

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

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

stat   The statistical transformation to use on the data for this layer, either as a ggproto
                    Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g.
                    "count" rather than "stat_count")

position   Position adjustment, either as a string naming the adjustment (e.g. "jitter" to
                    use position_jitter), or the result of a call to a position adjustment function.
                    Use the latter if you need to change the settings of the adjustment.

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

just   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. position_dodge().

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

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

orientation   The orientation of the layer. The default (NA) automatically determines the ori-
                    entation 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 *Orienta-
                    tion* section for more detail.

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

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

Value

XLayer or YLayer object to be added to a ggplot object

Aesthetics

Required aesthetics are in bold.

- x
- y

- fill *or* xfill Fill color of the xsidebar

- fill *or* yfill Fill color of the ysidebar

- width specifies the width of each bar

- height specifies the height of each bar

- alpha Transparency level of xfill or yfill

- size size of the border line.

See Also

[geom_xsidehistogram](), [geom_ysidehistogram]()

Examples

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

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

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

---

geom_xsideboxplot   *Side boxplots*

Description

The [xside]() and [yside]() variants of [geom_boxplot]() is [geom_xsideboxplot]() and [geom_ysideboxplot]().

Usage

```
geom_xsideboxplot(
  mapping = NULL,
  data = NULL,
  stat = "boxplot",
  position = "dodge2",
  ...,
  outliers = TRUE,
  outlier.colour = NULL,
  outlier.color = NULL,
  outlier.fill = NULL,
  outlier.shape = 19,
```

```
    outlier.size = 1.5,
    outlier.stroke = 0.5,
    outlier.alpha = NULL,
    notch = FALSE,
    notchwidth = 0.5,
    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 = 19,
    outlier.size = 1.5,
    outlier.stroke = 0.5,
    outlier.alpha = 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 inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.

	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to [layer()](). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
outliers	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 outlier.shape = NA instead.

outlier.colour, outlier.color, outlier.fill, outlier.shape, outlier.size, outlier.stroke, outlier.alpha

	Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.
	In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
notch	If FALSE (default) make a standard box plot. If TRUE, make a notched box plot. Notches are used to compare groups; if the notches of two boxes do not overlap, this suggests that the medians are significantly different.
notchwidth	For a notched box plot, width of the notch relative to the body (defaults to notchwidth = 0.5).
staplewidth	The relative width of staples to the width of the box. Staples mark the ends of the whiskers with a line.
varwidth	If FALSE (default) make a standard box plot. If TRUE, boxes are drawn with widths proportional to the square-roots of the number of observations in the groups (possibly weighted, using the weight aesthetic).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. [borders()]().

Value

XLayer or YLayer object to be added to a ggplot object

See Also

geom_*sideviolin

Examples

```
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: They order of the layers are 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",
  ...,
  na.rm = FALSE,
  orientation = "x",
  show.legend = NA,
  inherit.aes = TRUE,
  outline.type = "upper"
)

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

Arguments

mapping	Set of aesthetic mappings created by [aes()](). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	Use to override the default connection between geom_density() and stat_density().
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to [layer()](). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also

na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
outline.type	Type of the outline of the area; "both" draws both the upper and lower lines, "upper"/"lower" draws the respective lines only. "full" draws a closed polygon around the area.

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
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   *Side Frequency Polygons*

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
)

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

Arguments

mapping	Set of aesthetic mappings created by [aes()](#). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()](#).
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()](#) for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to [layer()](#). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
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   *Side function plot*

Description

The xside and yside variants of geom_function

Usage

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

stat_xsidefunction(
  mapping = NULL,
  data = NULL,
  geom = "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 inherit.aes =
            TRUE (the default), it is combined with the default mapping at the top level of
            the plot. You must supply mapping if there is no plot mapping.

data   Ignored by stat_function(), do not use.

stat   The statistical transformation to use on the data for this layer, either as a ggproto
            Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g.
            "count" rather than "stat_count")

position   Position adjustment, either as a string naming the adjustment (e.g. "jitter" to
            use position_jitter), or the result of a call to a position adjustment function.
            Use the latter if you need to change the settings of the adjustment.

...   Other arguments passed on to [layer()](). These are often aesthetics, used to set
            an aesthetic to a fixed value, like colour = "red" or size = 3. They may also

	be parameters to the paired geom/stat.
na.rm	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
geom	The geometric object to use to display the data, either as a `ggproto` Geom subclass or as a string naming the geom stripped of the `geom_` prefix (e.g. `"point"` rather than `"geom_point"`)
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 `fun`.
ylim	Optionally, restrict the range of the function to this range (y-axis)

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
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
```

---

geom_xsidehistogram   *Side Histograms*

Description

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

Usage

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

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

Arguments

mapping	Set of aesthetic mappings created by [aes()](). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")

position	Position adjustment, either as a string naming the adjustment (e.g. `"jitter"` to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to [`layer()`](). These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
binwidth	The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in `bins`, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data.
	The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.
bins	Number of bins. Overridden by `binwidth`. Defaults to 30.
na.rm	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting `orientation` to either `"x"` or `"y"`. See the *Orientation* section for more detail.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. [`borders()`]().

Value

XLayer or YLayer object to be added to a ggplot object

Aesthetics

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

Examples

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

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

geom_xsidelabel *Side label*

Description

The xside and yside variants of geom_label.

Usage

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

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

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string, or the result of a call to a position adjustment function. Cannot be jointly specified with nudge_x or nudge_y.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.
nudge_x, nudge_y	
	Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.
label.padding	Amount of padding around label. Defaults to 0.25 lines.
label.r	Radius of rounded corners. Defaults to 0.15 lines.
label.size	Size of label border, in mm.
size.unit	How the size aesthetic is interpreted: as millimetres ("mm", default), points ("pt"), centimetres ("cm"), inches ("in"), or picas ("pc").
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

XLayer or YLayer object to be added to a ggplot object

---

geom_xsideline *Side line plot*

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

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

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

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

Arguments

mapping	Set of aesthetic mappings created by [aes()](). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()](). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
arrow	Arrow specification, as created by grid::arrow().

Value

XLayer or YLayer object to be added to a ggplot object

Examples

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

---

geom_xsidepoint   *Side Points*

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 inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to [layer()](). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. [borders()]().

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
ggplot(diamonds, aes(depth, table, alpha = .2)) +
  geom_point() +
  geom_ysidepoint(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 inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to [layer()](). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
arrow	specification for arrow heads, as created by [grid::arrow()]().
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).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. [borders()]().

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
library(dplyr)
library(tidyr)
library(ggdendro)
```

```
#dendrogram with geom_*sidesegment
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 = "identity",
  ...,
  parse = FALSE,
  nudge_x = 0,
  nudge_y = 0,
  check_overlap = FALSE,
  size.unit = "mm",
  na.rm = FALSE,
```

```
    show.legend = NA,
    inherit.aes = TRUE
)

geom_ysidetext(
    mapping = NULL,
    data = NULL,
    stat = "identity",
    position = "identity",
    ...,
    parse = FALSE,
    nudge_x = 0,
    nudge_y = 0,
    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 inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string, or the result of a call to a position adjustment function. Cannot be jointly specified with nudge_x or nudge_y.
...	Other arguments passed on to [layer()](). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.
nudge_x, nudge_y	
	Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.

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

size.unit   How the size aesthetic is interpreted: as millimetres ("mm", default), points ("pt"), centimetres ("cm"), inches ("in"), or picas ("pc").

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

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

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

Value

XLayer or YLayer object to be added to a ggplot object

---

geom_xsidetile   *Side tile plot*

Description

The xside and yside variants of geom_tile

Usage

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

geom_ysidetile(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
```

```
    linejoin = "mitre",
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by [aes()](). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to [layer()](). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
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.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. [borders()]().

Value

XLayer or YLayer object to be added to a ggplot object

Examples

```
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",
  ...,
  draw_quantiles = NULL,
  trim = TRUE,
  bounds = c(-Inf, Inf),
  scale = "area",
  na.rm = FALSE,
  orientation = "x",
  show.legend = NA,
  inherit.aes = TRUE
)

geom_ysideviolin(
  mapping = NULL,
  data = NULL,
```

```
    stat = "ydensity",
    position = "dodge",
    ...,
    draw_quantiles = NULL,
    trim = TRUE,
    bounds = c(-Inf, Inf),
    scale = "area",
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE,
    orientation = "y"
)
```

Arguments

mapping	Set of aesthetic mappings created by [aes()](). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	Use to override the default connection between geom_violin() and stat_ydensity().
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to [layer()](). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
draw_quantiles	If not(NULL) (default), draw horizontal lines at the given quantiles of the density estimate.
trim	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don't trim the tails.
bounds	Known lower and upper bounds for estimated data. Default c(-Inf, Inf) 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
scale	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.

na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

XLayer or YLayer object to be added to a ggplot object

See Also

geom_*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 varialbe 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

x.pos	x side panel can either take "top" or "bottom"
y.pos	y side panel can either take "right" or "left"
scales	Determines side panel's unaligned axis scale. Inputs are similar to facet_* scales function. Default is set to "fixed", but "free_x", "free_y" and "free" are acceptable inputs. For example, xside panels are aligned to the x axis of the main panel. Setting "free" or "free_y" will cause all y scales of the x side Panels to be independent.
collapse	Determines if side panels should be collapsed into a single panel. Set "x" to collapse all x side panels, set "y" to collapse all y side panels, set "all" to collapse both x and y side panels.
draw_x_on, draw_y_on	
	Determines where the axis is rendered. For example: By default, the bottom x-axis is rendered on the bottom most panel per column. If set to "main", then the axis is rendered on the bottom of the bottom most main panel. If set to "side", then the x-axis is rendered on the bottom of the bottom most side panel(s). You may apply this logic to all axis positions.
strip	Determines if the strip should be rendered on the main plot or on their default locations. Only has an effect on facet_grid.

respect_side_labels

> 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   *Deprecated Functions*

Description

The following functions have been deprecated.

as_ggsideFacet <- ggside_facet as_ggsideCoord <- ggside_coord

---

ggside-scales-binned   *Position scales for binning continuous data ggside scales*

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 `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

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 TRUE (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:

  - NULL for no breaks
  - waiver() for the default breaks computed by the [transformation object](#)
  - A numeric vector of positions
  - A function that takes the limits as input and returns breaks as output (e.g., a function returned by [scales::extended_breaks()](#)). Also accepts rlang [lambda](#) function notation.

labels
: One of:

  - NULL for no labels
  - waiver() for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as breaks)
  - An expression vector (must be the same length as breaks). See ?plotmath for details.

- A function that takes the breaks as input and returns labels as output. Also accepts rlang [lambda](#) function notation.

limits   One of:

- NULL to use the default scale range
- A numeric vector of length two providing limits of the scale. Use NA to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang [lambda](#) function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see [coord_cartesian()](#)).

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

oob   One of:

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

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

right   Should the intervals be closed on the right (TRUE, default) or should the intervals be closed on the left (FALSE)? '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).

show.limits   should the limits of the scale appear as ticks

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

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called transform_<name>. If transformations require arguments, you can call them from the scales package, e.g. [scales::transform_boxcox(p = 2)](#). You can create your own transformation with [scales::new_transform()](#).

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::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 `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

breaks
: One of:

  - `NULL` for no breaks
  - `waiver()` for the default breaks computed by the [transformation object](#)
  - A numeric vector of positions
  - A function that takes the limits as input and returns breaks as output (e.g., a function returned by [`scales::extended_breaks()`](#)). Also accepts rlang [lambda](#) function notation.

minor_breaks
: One of:

  - `NULL` for no minor breaks
  - `waiver()` for the default breaks (one minor break between each major break)
  - A numeric vector of positions
  - A function that given the limits returns a vector of minor breaks. Also accepts rlang [lambda](#) function notation. When the function has two arguments, it will be given the limits and major breaks.

n.breaks
: An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.

labels
: One of:

  - `NULL` for no labels
  - `waiver()` for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as `breaks`)
  - An expression vector (must be the same length as breaks). See ?plotmath for details.
  - A function that takes the breaks as input and returns labels as output. Also accepts rlang [lambda](#) function notation.

limits
: One of:

  - `NULL` to use the default scale range
  - A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum

- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang [lambda](#) function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see [coord_cartesian()](#)).

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

oob   One of:

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

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

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

             A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called transform_<name>. If transformations require arguments, you can call them from the scales package, e.g. [scales::transform_boxcox(p](#) [= 2)](#). You can create your own transformation with [scales::new_transform()](#).

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.

sec.axis   [sec_axis()](#) is used to specify a secondary axis.

...   Other arguments passed on to scale_(y|x)side(x|y)_continuous()

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](#)

        palette A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., [scales::pal_hue()](#)).

        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 uses all the levels in the factor.

na.translate Unlike continuous scales, discrete scales can easily show miss-
ing 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 al-
ways placed at the far right.

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

scale_name **[Deprecated]** The name of the scale that should be used for error
messages associated with this scale.

labels One of:

- NULL for no labels
- waiver() for the default labels computed by the transformation object
- A character vector giving labels (must be same length as breaks)
- 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 informa-
                tion.

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   *ggside geom constructor*

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`   *New ggside layer*

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`	The geometric object to use to display the data, either as a `ggproto` Geom subclass or as a string naming the geom stripped of the `geom_` prefix (e.g. `"point"` rather than `"geom_point"`)
`stat`	The statistical transformation to use on the data for this layer, either as a `ggproto` Geom subclass or as a string naming the stat stripped of the `stat_` prefix (e.g. `"count"` rather than `"stat_count"`)
`data`	The data to be displayed in this layer. There are three options:
	If `NULL`, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().

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

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

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

position   Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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. borders().

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.

key_glyph   A legend key drawing function or a string providing the function name minus the draw_key_ prefix. See draw_key for details.

side   should the resulting ggplot2_layer be configured for x or y side

layer   a LayerInstance object made from layer

---

ggside_layout   *Construct ggside layout*

Description

Creates a new layout object required for ggside functionality

Usage

```
ggside_layout(layout)
```

Arguments

layout   a ggproto Layout object

is.ggside	*Check ggside objects*

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	*Extending base ggproto classes for ggside*

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   *Rescale x or y onto new range in margin*

Description

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

Usage

```
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 | *Scales for the *colour aesthetics* |

Description

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

Value

returns a ggproto object to be added to a ggplot

Related Functions

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

---

scale_xfill *Scales for the *fill aesthetics*

Description

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

Value

returns a ggproto object to be added to a ggplot

Related Functions

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

---

scale_ycolour_hue *scale_ycolour_hue*

Description

scale_ycolour_hue

scale_ycolour_manual

scale_ycolour_gradient

scale_ycolour_discrete

scale_ycolour_discrete

scale_ycolour_continuous

scale_ycolour_continuous

---

scale_yfill_hue   *scale_yfill_hue*

Description

scale_yfill_hue

scale_yfill_manual

scale_yfill_gradient

scale_yfill_discrete

scale_yfill_continuous

---

stat_summarise   *Summarise by grouping variable*

Description

Applies a function to a specified grouping variable

Usage

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

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

Arguments

mapping	Set of aesthetic mappings created by [aes()](). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options:
	If NULL, the default, the data is inherited from the plot data as specified in the call to [ggplot()]().
	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()]() for which variables will be created.
	A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data, either as a ggproto Geom subclass or as a string naming the geom stripped of the geom_ prefix (e.g. "point" rather than "geom_point")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	additional arguments to pass to [layer]().
fun	Summarising function to use. If no function provided it will default to [length]().
args	List of additional arguments passed to the function.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. [borders()]().

Format

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

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

Value

A Layer object to be added to a ggplot

Aesthetics

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

Examples

```
library(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   *ggside custom themes*

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.len

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, ggs

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   *The xside geometries*

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