

Package ‘ggalign’

October 18, 2025

Title A 'ggplot2' Extension for Composable Visualization

Version 1.2.0

Description A 'ggplot2' extension providing an integrative framework for composable visualization, enabling the creation of complex multi-plot layouts such as insets, circular arrangements, and multi-panel compositions. Built on the grammar of graphics, it offers tools to align, stack, and nest plots, simplifying the construction of richly annotated figures for high-dimensional data contexts—such as genomics, transcriptomics, and microbiome studies—by making it easy to link related plots, overlay clustering results, or highlight shared patterns.

License MIT + file LICENSE

URL <https://github.com/Yunuuuu/ggalign>,
<https://yunuuuu.github.io/ggalign/>

BugReports <https://github.com/Yunuuuu/ggalign/issues>

Depends ggplot2 (>= 4.0.0)

Imports S7, vctrs (>= 0.5.0), rlang, cli, grDevices, grid, gtable,
scales, methods, stats, utils, lifecycle

Suggests gridGraphics, ragg, magick, testthat (>= 3.0.0), vdiff (>= 1.0.6)

Enhances patchwork, ggrastr, maftools

ByteCompile true

Config/Needs/check R.utils, patchwork, ape, lattice, ComplexHeatmap,
pheatmap

Config/Needs/website patchwork, bench, pheatmap, gplots,
ComplexHeatmap

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Language en-GB

Collate 'active.R' 'alignpatch-.R' 'ggplot-theme.R'
 'alignpatch-theme.R' 'utils-ggplot.R' 'alignpatch-tags.R'
 'alignpatch-title.R' 'alignpatch-area.R'
 'import-standalone-pkg.R' 'utils-grid.R' 'alignpatch-design.R'
 'alignpatch-alignpatches.R' 'alignpatch-ggplot2.R'
 'alignpatch-build.R' 'alignpatch-free-align.R'
 'alignpatch-free-border.R' 'alignpatch-free-guide.R'
 'alignpatch-free-lab.R' 'alignpatch-free-space.R'
 'alignpatch-free-vp.R' 'alignpatch-grob.R'
 'alignpatch-guides.R' 'alignpatch-inset.R'
 'alignpatch-patchwork.R' 'alignpatch-update.R'
 'alignpatch-wrap.R' 'attributes.R' 'craftsman.R'
 'craft-align-.R' 'craft-align-hclust.R'
 'craft-align-dendrogram.R' 'craft-align-group.R'
 'craft-align-kmeans.R' 'craft-align-order.R'
 'craft-align-order2.R' 'craft-align-phylo.R' 'craft-cross-.R'
 'craft-cross-link.R' 'craft-cross-mark.R' 'craft-cross-none.R'
 'scheme-.R' 'craftbox.R' 'domain.R' 'fortify-data-frame-.R'
 'fortify-data-frame-dendrogram.R' 'fortify-data-frame-phylo.R'
 'fortify-matrix-.R' 'fortify-matrix-list.R'
 'fortify-matrix-maftools.R' 'fortify-matrix-matrix.R'
 'genomic-helper.R' 'ggalign-package.R' 'ggalign.R' 'ggcross.R'
 'ggfree.R' 'ggmark.R' 'ggplot-coord-circle.R'
 'ggplot-facet-sector.R' 'ggplot-geom-draw.R'
 'ggplot-geom-gshape.R' 'ggplot-geom-magick.R'
 'ggplot-geom-pie.R' 'ggplot-geom-rect3d.R'
 'ggplot-geom-subrect.R' 'ggplot-helper.R' 'grid-grob-magick.R'
 'import-standalone-assert.R' 'import-standalone-obj-type.R'
 'import-standalone-purrr.R' 'import-standalone-tibble.R'
 'layer-order.R' 'layout-.R' 'layout-align.R'
 'layout-quad-scope.R' 'layout-operator.R' 'layout-chain-.R'
 'layout-chain-circle-.R' 'layout-chain-circle-build.R'
 'layout-chain-circle-genomic.R' 'layout-chain-circle-switch.R'
 'layout-chain-stack-.R' 'layout-chain-stack-build.R'
 'layout-chain-stack-composer.R' 'layout-chain-stack-cross.R'
 'layout-chain-stack-genomic.R' 'layout-chain-stack-switch.R'
 'layout-domain.R' 'layout-heatmap-.R' 'layout-heatmap-build.R'
 'layout-heatmap-oncplot.R' 'layout-quad-.R'
 'layout-quad-build.R' 'layout-quad-operator.R'
 'layout-quad-switch.R' 'layout-quad-upset.R' 'link.R' 'mark.R'
 'object-name.R' 'pair-links.R' 'plot-ideogram.R'
 'raster-magick.R' 'rasterise.R' 'scheme-align.R'
 'scheme-data.R' 'scheme-theme.R' 'tune.R' 'utils-assert.R'
 'utils-rd.R' 'utils.R' 'zzz.R'

NeedsCompilation no

Author Yun Peng [aut, cre] (ORCID: <<https://orcid.org/0000-0003-2801-3332>>),
 Shixiang Wang [aut] (ORCID: <<https://orcid.org/0000-0001-9855-7357>>),
 Guangchuang Yu [ths] (ORCID: <<https://orcid.org/0000-0002-6485-8781>>)

Maintainer Yun Peng <yunyunp96@163.com>

Repository CRAN

Date/Publication 2025-10-18 15:50:02 UTC

Contents

.link_draw	5
.mark_draw	6
active	7
alignpatches	7
align_dendro	10
align_group	12
align_hclust	13
align_kmeans	14
align_order	15
align_order2	16
align_phylo	18
area	19
as_grob.formula	20
as_grob.ggalien::alignpatches	21
as_grob.ggplot	22
as_grob.gList	23
as_grob.grob	23
as_grob.Heatmap	24
as_grob.patch	25
as_grob.patchwork	26
as_grob.pheatmap	27
as_grob.recordedplot	28
as_grob.trellis	29
circle_genomic	31
circle_layout	32
circle_switch	34
continuous_limits	35
coord_circle	36
cross_link	37
cross_mark	38
cross_none	39
draw_key_gshape	40
element_vec	41
facet_sector	42
fortify_data_frame	43
fortify_data_frame.character	44
fortify_data_frame.default	45
fortify_data_frame.dendrogram	46
fortify_data_frame.GRanges	48
fortify_data_frame.matrix	49
fortify_data_frame.phylo	50

fortify_matrix	52
fortify_matrix.default	53
fortify_matrix.GISTIC	53
fortify_matrix.list_upset	55
fortify_matrix.MAF	56
fortify_matrix.matrix	58
fortify_matrix.matrix_oncoplot	59
fortify_matrix.matrix_upset	60
free_align	61
genomic_density	64
genomic_dist	65
geom_draw	66
geom_gshape	69
geom_magick	72
geom_pie	75
geom_rect3d	78
geom_subrect	81
ggalign	85
ggalignGrob	87
ggalign_attr	88
ggalign_data_set	89
ggalign_stat	89
ggcross	90
ggfree	91
ggmark	92
ggoncoplot	94
ggupset	96
ggwrap	98
hclust2	99
heatmap_layout	100
inset	102
is_layout	103
layer_order	104
layout-operator	104
layout_design	106
layout_tags	107
layout_theme	109
layout_title	117
link_draw	118
link_line	119
link_tetragon	120
magickGrob	120
mark_draw	121
mark_line	122
mark_tetragon	123
mark_triangle	123
memo_order	124
new_tune	124

no_expansion	125
order2	125
pair_links	126
patch_title	127
plot_ideogram	129
quad_active	130
quad_layout	133
quad_scope	135
quad_switch	137
raster_magick	138
read_example	139
scale_gshape_manual	140
scale_z_continuous	142
scheme_align	143
scheme_data	145
scheme_theme	145
stack_cross	154
stack_genomic	155
stack_layout	156
stack_switch	158
theme_no_axes	159
tune	160
tune.list	161
tune.MAF	161
tune.matrix	162

Index 163

<i>.link_draw</i>	<i>Define the links to connect a pair of observations</i>
-------------------	---

Description

A base version of `link_draw()`, optimized for performance. This function serves as the foundation for building other `link_*` functions that manage the drawing of links between pairs of observations.

Usage

```
.link_draw(.draw, ...)
```

Arguments

<code>.draw</code>	A function used to draw the links. The function must return a <code>grob()</code> object. If the function does not return a valid grob, no drawing will occur. The input data for the function contains a list, where each item is a list of two data frames: one for the left hand coordinates ("hand1") and one for the right hand observations coordinates ("hand2").
--------------------	--

... [<dyn-dots>](#) A list of formulas, where each side of the formula should be an integer or character index of the original data, or a `range_link()` object defining the linked observations. Use `NULL` to indicate no link on that side. You can also combine these by wrapping them into a single `list()`. If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with `I()` to use the ordering from the layout. You can also use `waiver()` to inherit values from the other group.

See Also

[link_draw\(\)](#)

.mark_draw

Define the links to connect the marked observations

Description

A base version of [mark_draw](#), designed for performance optimization. This function is used to build other `mark_*` functions that manage the drawing of links between marked observations.

Usage

```
.mark_draw(.draw, ...)
```

Arguments

`.draw` A function used to draw the links. The function must return a [grob\(\)](#) object. If the function does not return a valid grob, nothing will be drawn. The input data for the function contains a list, where each item is a list of two data frames: one for the panel side coordinates ("panel") and one for the marked observations coordinates ("link").

... [<dyn-dots>](#) A list of formulas, where each side of the formula should be an integer or character index of the original data, or a `range_link()` object defining the linked observations. Use `NULL` to indicate no link on that side. You can also combine these by wrapping them into a single `list()`. If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with `I()` to use the ordering from the layout. You can also use `waiver()` to inherit values from the other group.

See Also

[mark_draw\(\)](#)

active	<i>Plot Adding Context Settings</i>
--------	-------------------------------------

Description

[Experimental]

These settings control the behavior of the plot when added to a layout, as well as the arrangement of individual plot areas within the layout.

Usage

```
active(order = NA_integer_, use = NA, name = NA_character_)
```

Arguments

order	An integer specifying the order of the plot area within the layout.
use	A logical (TRUE/FALSE) indicating whether to set the active context to the current plot when added to a layout. If TRUE, any subsequent ggplot elements will be applied to this plot.
name	A string specifying the plot's name, useful for switching active contexts through the what argument in functions like quad_anno() / stack_switch() .

Details

By default, the active context is set only for functions that add plot areas. This allows other ggplot2 elements-such as geoms, stats, scales, or themes- to be seamlessly added to the current plot area.

The default ordering of the plot areas is from top to bottom or from left to right, depending on the layout orientation. However, users can customize this order using the order argument.

alignpatches	<i>Arrange multiple plots into a grid</i>
--------------	---

Description

An internal S7 class that represents a collection of aligned plots along with their layout configuration, titles, tags, and theme.

Usage

```
alignpatches(
  ...,
  ncol = NULL,
  nrow = NULL,
  byrow = TRUE,
  widths = NA,
  heights = NA,
  area = NULL,
  guides = waiver(),
  theme = NULL,
  design = NULL
)

align_plots(
  ...,
  ncol = NULL,
  nrow = NULL,
  byrow = TRUE,
  widths = NA,
  heights = NA,
  area = NULL,
  guides = waiver(),
  theme = NULL,
  design = NULL
)
```

Arguments

...	<dyn-dots> A list of plots, ususally the ggplot object. Use NULL to indicate an empty spacer. Each input must implement the <code>patch()</code> method.
ncol, nrow	The number of columns and rows in the grid. Defaults to NULL. If both are NULL, the layout dimensions are determined automatically using the same logic as <code>facet_wrap()</code> .
byrow	A logical value indicating whether plots should be filled in row-major order (TRUE) or column-major order (FALSE). Defaults to TRUE.
widths, heights	The relative widths and heights of each column and row in the grid. These values are recycled to match the grid dimensions. The special value NA is treated as a unit of 1null, unless a fixed-aspect plot is included — in that case, the affected dimension will expand or contract to maintain the aspect ratio of the plot. Defaults to NA.
area	A specification of the area layout. Can be defined either as a character string or as a combination of calls to <code>area()</code> . Defaults to NULL.
guides	A string with one or more of "t", "l", "b", "r", and "i" indicating which side of guide legends should be collected. Defaults to <code>waiver()</code> , which inherits from the parent layout. If there is no parent layout, or if NULL is provided, no guides will be collected.

theme	A theme() object used to customize various elements of the layout. By default, the theme will inherit from the parent layout.
design	An alias for area, retained for backward compatibility.

Value

An alignpatches object.

Properties

- **plots:** A list of plot objects.
- **layout:** A list specifying layout options, including:
 - ncol, nrow, byrow: grid layout parameters.
 - widths, heights: relative dimensions of rows/columns.
 - area: custom area specification.
 - guides: guide handling.
- **titles:** A list specifying title options (title, subtitle, caption).
- **tags:** A list specifying tag options (tags, sep, prefix, suffix).
- **theme:** A theme configuration object.

See Also

- [layout_design\(\)](#)
- [layout_title\(\)](#)
- [layout_theme\(\)](#)
- [layout_tags\(\)](#)

Examples

```
# directly copied from patchwork
p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
  geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
  geom_bar(aes(gear)) +
  facet_wrap(~cyl)
p4 <- ggplot(mtcars) +
  geom_bar(aes(carb))
p5 <- ggplot(mtcars) +
  geom_violin(aes(cyl, mpg, group = cyl))

# Either add the plots as single arguments
align_plots(p1, p2, p3, p4, p5)

# Or use bang-bang-bang to add a list
align_plots(!!!list(p1, p2, p3), p4, p5)
```

```
# Match plots to areas by name
area <- "#BB
      AA#"
align_plots(B = p1, A = p2, area = area)

# Compare to not using named plot arguments
align_plots(p1, p2, area = area)
```

align_dendro	<i>Plot dendrogram tree</i>
--------------	-----------------------------

Description

Plot dendrogram tree

Usage

```
align_dendro(
  mapping = aes(),
  ...,
  distance = "euclidean",
  method = "complete",
  use_missing = "pairwise.complete.obs",
  reorder_dendrogram = FALSE,
  merge_dendrogram = FALSE,
  reorder_group = FALSE,
  k = NULL,
  h = NULL,
  cutree = NULL,
  plot_dendrogram = TRUE,
  plot_cut_height = NULL,
  center = FALSE,
  type = "rectangle",
  root = NULL,
  size = NULL,
  data = NULL,
  active = NULL,
  no_axes = deprecated()
)
```

Arguments

mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...	<dyn-dots> Additional arguments passed to geom_segment() .

distance	A string of distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Correlation coefficient can be also used, including "pearson", "spearman" or "kendall". In this way, 1 - cor will be used as the distance. In addition, you can also provide a <code>dist</code> object directly or a function return a <code>dist</code> object. Use NULL, if you don't want to calculate the distance.
method	A string of the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). You can also provide a function which accepts the calculated distance (or the input matrix if distance is NULL) and returns a <code>hclust</code> object. Alternative, you can supply an object which can be coerced to <code>hclust</code> .
use_missing	An optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Only used when distance is a correlation coefficient string.
reorder_dendrogram	A single boolean value indicating whether to reorder the dendrogram based on the means. Alternatively, you can provide a custom function that accepts an <code>hclust</code> object and the data used to generate the tree, returning either an <code>hclust</code> or <code>dendrogram</code> object. Default is FALSE.
merge_dendrogram	A single boolean value, indicates whether we should merge multiple dendrograms, only used when previous groups have been established. Default: FALSE.
reorder_group	A single boolean value, indicates whether we should do Hierarchical Clustering between groups, only used when previous groups have been established. Default: FALSE.
k	An integer scalar indicates the desired number of groups.
h	A numeric scalar indicates heights where the tree should be cut.
cutree	A function used to cut the <code>hclust</code> tree. It should accept four arguments: the <code>hclust</code> tree object, distance (only applicable when method is a string or a function for performing hierarchical clustering), k (the number of clusters), and h (the height at which to cut the tree). By default, <code>cutree()</code> is used.
plot_dendrogram	A boolean value indicates whether plot the dendrogram tree.
plot_cut_height	A boolean value indicates whether plot the cut height.
center	A boolean value. if TRUE, nodes are plotted centered with respect to all leaves/tips in the branch. Otherwise (default), plot them in the middle of the direct child nodes.
type	A string indicates the plot type, "rectangle" or "triangle".
root	A length one string or numeric indicates the root branch.
size	The relative size of the plot, can be specified as a <code>unit()</code> . Note that for <code>circle_layout()</code> , all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.

data	A matrix-like object. By default, it inherits from the layout matrix.
active	A <code>active()</code> object that defines the context settings when added to a layout.
no_axes	[Deprecated] Please add <code>theme()</code> directly to the ggplot instead.

ggplot2 specification

`align_dendro` initializes a ggplot data and mapping.

The internal ggplot object will always use a default mapping of `aes(x = .data$x, y = .data$y)`.

The default ggplot data is the node coordinates with edge data attached in `ggalign` attribute, in addition, a `geom_segment` layer with a data frame of the edge coordinates will be added when `plot_dendrogram = TRUE`.

See `fortify_data_frame.dendrogram()` for details.

Discrete Axis Alignment

It is important to note that we consider rows as observations, meaning `vec_size(data)/NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

Examples

```
# align_dendro will always add a plot area
ggheatmap(matrix(rnorm(81), nrow = 9)) +
  anno_top() +
  align_dendro()
ggheatmap(matrix(rnorm(81), nrow = 9)) +
  anno_top() +
  align_dendro(k = 3L)
```

align_group	<i>Group and align observations based on a group vector</i>
-------------	---

Description

[Stable]

Splits observations into groups, with slice ordering based on group levels.

Usage

```
align_group(group, active = NULL)
```

Arguments

group	A character define the groups of the observations.
active	A <code>active()</code> object that defines the context settings when added to a layout.

Examples

```
set.seed(1L)
small_mat <- matrix(rnorm(81), nrow = 9)
ggheatmap(small_mat) +
  anno_top() +
  align_group(sample(letters[1:4], ncol(small_mat), replace = TRUE))
```

align_hclust

*Reorder or Group observations based on hierarchical clustering***Description****[Stable]**

This function aligns observations within the layout according to a hierarchical clustering tree, enabling reordering or grouping of elements based on clustering results.

Usage

```
align_hclust(
  distance = "euclidean",
  method = "complete",
  use_missing = "pairwise.complete.obs",
  reorder_dendrogram = FALSE,
  reorder_group = FALSE,
  k = NULL,
  h = NULL,
  cutree = NULL,
  data = NULL,
  active = NULL
)
```

Arguments

- | | |
|----------|---|
| distance | A string of distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Correlation coefficient can be also used, including "pearson", "spearman" or "kendall". In this way, 1 - cor will be used as the distance. In addition, you can also provide a dist object directly or a function return a dist object. Use NULL, if you don't want to calculate the distance. |
| method | A string of the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). You can also provide a function which accepts the calculated distance (or the input matrix if distance is NULL) and returns a hclust object. Alternative, you can supply an object which can be coerced to hclust . |

use_missing	An optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Only used when distance is a correlation coefficient string.
reorder_dendrogram	A single boolean value indicating whether to reorder the dendrogram based on the means. Alternatively, you can provide a custom function that accepts an hclust object and the data used to generate the tree, returning either an hclust or dendrogram object. Default is FALSE.
reorder_group	A single boolean value, indicates whether we should do Hierarchical Clustering between groups, only used when previous groups have been established. Default: FALSE.
k	An integer scalar indicates the desired number of groups.
h	A numeric scalar indicates heights where the tree should be cut.
cutree	A function used to cut the hclust tree. It should accept four arguments: the hclust tree object, distance (only applicable when method is a string or a function for performing hierarchical clustering), k (the number of clusters), and h (the height at which to cut the tree). By default, cutree() is used.
data	A matrix-like object. By default, it inherits from the layout matrix.
active	A active() object that defines the context settings when added to a layout.

Discrete Axis Alignment

It is important to note that we consider rows as observations, meaning `vec_size(data)/NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

See Also

[hclust2\(\)](#)

Examples

```
# align_hclust won't add a dendrogram
ggheatmap(matrix(rnorm(81), nrow = 9)) +
  anno_top() +
  align_hclust(k = 3L)
```

align_kmeans

Split observations by k-means clustering groups.

Description

[Stable]

Aligns and groups observations based on k-means clustering, enabling observation splits by cluster groups.

Usage

```
align_kmeans(..., data = NULL, active = NULL)
```

Arguments

...	Arguments passed on to <code>stats::kmeans</code> <code>iter.max</code> the maximum number of iterations allowed. <code>nstart</code> if centers is a number, how many random sets should be chosen? <code>algorithm</code> character: may be abbreviated. Note that "Lloyd" and "Forgy" are alternative names for one algorithm. <code>trace</code> logical or integer number, currently only used in the default method ("Hartigan-Wong"): if positive (or true), tracing information on the progress of the algorithm is produced. Higher values may produce more tracing information.
data	A numeric matrix to be used by k-means. By default, it will inherit from the layout matrix.
active	A <code>active()</code> object that defines the context settings when added to a layout.

Discrete Axis Alignment

It is important to note that we consider rows as observations, meaning `vec_size(data)/NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
  anno_top() +
  align_kmeans(3L)
```

<code>align_order</code>	<i>Order observations based on weights</i>
--------------------------	--

Description**[Stable]**

Ordering observations based on summary weights or a specified ordering character or integer index.

Usage

```
align_order(
  weights = rowMeans,
  ...,
  reverse = FALSE,
  strict = TRUE,
  data = NULL,
  active = NULL
)
```

Arguments

weights	A summary function which accepts a data and returns the weights for each observations. Alternatively, you can provide an ordering index as either an integer or a character. Since characters have been designated as character indices, if you wish to specify a function name as a string, you must enclose it with <code>I()</code> .
...	<dyn-dots> Additional arguments passed to function provided in weights argument.
reverse	A boolean value. Should the sort order be in reverse?
strict	A boolean value indicates whether the order should be strict. If previous groups has been established, and strict is FALSE, this will reorder the observations in each group.
data	A matrix, data frame, or atomic vector used as the input for the weights function. Alternatively, you can specify a function (including purrr-like lambda syntax) that will be applied to the layout matrix, transforming it as necessary for weight calculations. By default, it will inherit from the layout matrix.
active	A <code>active()</code> object that defines the context settings when added to a layout.

Discrete Axis Alignment

It is important to note that we consider rows as observations, meaning `vec_size(data)/NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
  anno_left() +
  align_order(I("rowMeans"))
```

align_order2

Reorders layout observations based on specific statistics.

Description

Reorders layout observations based on specific statistics.

Usage

```
align_order2(
  stat,
  ...,
  reverse = FALSE,
  strict = TRUE,
  data = NULL,
  active = NULL
)
```

Arguments

stat	A statistical function which accepts a data and returns the statistic, which we'll call <code>order2()</code> to extract the ordering information.
...	<dyn-dots> Additional arguments passed to function provided in stat argument.
reverse	A boolean value. Should the sort order be in reverse?
strict	A boolean value indicates whether the order should be strict. If previous groups has been established, and strict is FALSE, this will reorder the observations in each group.
data	A matrix, data frame, or atomic vector used as the input for the stat function. Alternatively, you can specify a function (including purrr-like lambda syntax) that will be applied to the layout matrix, transforming it as necessary for statistic calculations. By default, it will inherit from the layout matrix.
active	A <code>active()</code> object that defines the context settings when added to a layout.

Details

[Experimental]

The `align_order2()` function differs from `align_order()` in that the `weights` argument in `align_order()` must return atomic weights for each observation. In contrast, the `stat` argument in `align_order2()` can return more complex structures, such as `hclust` or `dendrogram`, among others.

Typically, you can achieve the functionality of `align_order2()` using `align_order()` by manually extracting the ordering information from the statistic.

Discrete Axis Alignment

It is important to note that we consider rows as observations, meaning `vec_size(data)/NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

See Also

[order2\(\)](#)

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +  
  anno_left() +  
  align_order2(hclust2)
```

align_phylo

*Plot Phylogenetics tree***Description**

Plot Phylogenetics tree

Usage

```
align_phylo(
  phylo,
  ...,
  mapping = aes(),
  split = FALSE,
  ladderize = NULL,
  type = "rectangle",
  center = FALSE,
  tree_type = NULL,
  root = NULL,
  active = NULL,
  size = NULL,
  no_axes = deprecated()
)
```

Arguments

phylo	A phylo object.
...	<dyn-dots> Additional arguments passed to geom_segment() .
mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
split	A logical scalar indicating whether to split the phylogenetic tree into separate subtrees when multiple panel groups are present.
ladderize	A single string of "left" or "right", indicating whether to ladderize the tree. Ladderizing arranges the tree so that the smallest clade is positioned on the "right" or the "left". By default, NULL means the tree will not be ladderized.
type	A string indicates the plot type, "rectangle" or "triangle".
center	A boolean value. if TRUE, nodes are plotted centered with respect to all leaves/tips in the branch. Otherwise (default), plot them in the middle of the direct child nodes.
tree_type	A single string, one of "phylogram" or "cladogram", indicating the type of tree. • phylogram: Represents a phylogenetic tree where branch lengths indicate evolutionary distance or time. • cladogram: Represents a tree where branch lengths are not used, or the branches do not reflect evolutionary time.

	Usually, you don't need to modify this.
root	A length one string or numeric indicates the root branch.
active	A <code>active()</code> object that defines the context settings when added to a layout.
size	The relative size of the plot, can be specified as a <code>unit()</code> . Note that for <code>circle_layout()</code> , all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.
no_axes	[Deprecated] Please add <code>theme()</code> directly to the ggplot instead.

area	<i>Define the plotting areas</i>
------	----------------------------------

Description

This is a small helper used to specify a single area in a rectangular grid that should contain a plot. Objects constructed with `area()` can be concatenated together with `c()` in order to specify multiple areas.

Usage

```
area(t, l, b = t, r = l)
```

Arguments

t, b	The top and bottom bounds of the area in the grid
l, r	The left and right bounds of the area in the grid

Details

The grid that the areas are specified in reference to enumerate rows from top to bottom, and columns from left to right. This means that t and l should always be less or equal to b and r respectively. Instead of specifying area placement with a combination of `area()` calls, it is possible to instead pass in a single string

```
areas <- c(area(1, 1, 2, 1),
           area(2, 3, 3, 3))
```

is equivalent to

```
areas <- -"A##
          A#B
          ##B"
```

Value

A `galign_area` object.

Examples

```

p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
  geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
  geom_bar(aes(gear)) +
  facet_wrap(~cyl)

layout <- c(
  area(1, 1),
  area(1, 3, 3),
  area(3, 1, 3, 2)
)

# Show the layout to make sure it looks as it should
plot(layout)

# Apply it to a alignpatches
align_plots(p1, p2, p3, design = layout)

```

as_grob.formula	<i>Convert Object into a Grob</i>
-----------------	-----------------------------------

Description

Convert Object into a Grob

Usage

```

## S3 method for class 'formula'
as_grob(x, ..., device = NULL, name = NULL)

## S3 method for class '`function`'
as_grob(x, ..., device = NULL, name = NULL)

```

Arguments

x	An object to be converted into a grob .
...	Graphical Parameters passed on to par() .
device	A function that opens a graphics device for <code>grid.echo()</code> to work on. By default this is an off-screen, in-memory device based on the pdf device. This default device may not be satisfactory when using custom fonts.
name	A character identifier.

Value

A [grob](#) object.

as_grob method collections

- `as_grob.grob()`
- `as_grob.gList()`
- `as_grob.patch_ggplot()`
- `as_grob.ggplot()`
- `as_grob.ggalgn::alignpatches()`
- `as_grob.patchwork()`
- `as_grob.patch()`
- `as_grob.formula()`
- `as_grob.recordedplot()`
- `as_grob.trellis()`
- `as_grob.Heatmap()`
- `as_grob.pheatmap()`

See Also`plot()`

Other `as_grob`: `as_grob.Heatmap()`, `as_grob.gList()`, `as_grob.ggalgn::alignpatches()`, `as_grob.ggplot()`, `as_grob.grob()`, `as_grob.patch()`, `as_grob.patch_ggplot()`, `as_grob.patchwork()`, `as_grob.pheatmap()`, `as_grob.recordedplot()`, `as_grob.trellis()`

`as_grob.ggalgn::alignpatches`
Convert Object into a Grob

Description

Convert Object into a Grob

Usage

```
## S3 method for class '`ggalign::alignpatches`'  
as_grob(x, ...)
```

Arguments

<code>x</code>	An object to be converted into a <code>grob</code> .
<code>...</code>	Not used currently.

Value

A `grob` object.

See Also

[align_plots\(\)](#)

Other as_grob: [as_grob.Heatmap\(\)](#), [as_grob.formula\(\)](#), [as_grob.gList\(\)](#), [as_grob.ggplot\(\)](#), [as_grob.grob\(\)](#), [as_grob.patch\(\)](#), [as_grob.patch_ggplot\(\)](#), [as_grob.patchwork\(\)](#), [as_grob.pheatmap\(\)](#), [as_grob.recordedplot\(\)](#), [as_grob.trellis\(\)](#)

as_grob.ggplot	<i>Convert Object into a Grob</i>
----------------	-----------------------------------

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'ggplot'
as_grob(x, ...)
```

Arguments

x	An object to be converted into a grob .
...	Not used currently.

Value

A [grob](#) object.

See Also

[ggplot](#)

Other as_grob: [as_grob.Heatmap\(\)](#), [as_grob.formula\(\)](#), [as_grob.gList\(\)](#), [as_grob.ggalign::alignpatches\(\)](#), [as_grob.grob\(\)](#), [as_grob.patch\(\)](#), [as_grob.patch_ggplot\(\)](#), [as_grob.patchwork\(\)](#), [as_grob.pheatmap\(\)](#), [as_grob.recordedplot\(\)](#), [as_grob.trellis\(\)](#)

as_grob.gList	<i>Convert Object into a Grob</i>
---------------	-----------------------------------

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'gList'  
as_grob(x, ...)
```

Arguments

x	An object to be converted into a grob .
...	Additional arguments passed on to gTree .

Value

A [grob](#) object.

See Also

Other as_grob: [as_grob.Heatmap\(\)](#), [as_grob.formula\(\)](#), [as_grob.ggaligned::alignpatches\(\)](#), [as_grob.ggplot\(\)](#), [as_grob.grob\(\)](#), [as_grob.patch\(\)](#), [as_grob.patch_ggplot\(\)](#), [as_grob.patchwork\(\)](#), [as_grob.pheatmap\(\)](#), [as_grob.recordedplot\(\)](#), [as_grob.trellis\(\)](#)

as_grob.grob	<i>Convert Object into a Grob</i>
--------------	-----------------------------------

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'grob'  
as_grob(x, ...)
```

Arguments

x	An object to be converted into a grob .
...	Not used currently.

Value

A [grob](#) object.

See Also

Other as_grob: [as_grob.Heatmap\(\)](#), [as_grob.formula\(\)](#), [as_grob.gList\(\)](#), [as_grob.ggalignment::alignpatches\(\)](#), [as_grob.ggplot\(\)](#), [as_grob.patch\(\)](#), [as_grob.patch_ggplot\(\)](#), [as_grob.patchwork\(\)](#), [as_grob.pheatmap\(\)](#), [as_grob.recordedplot\(\)](#), [as_grob.trellis\(\)](#)

as_grob.Heatmap	<i>Convert Object into a Grob</i>
-----------------	-----------------------------------

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'Heatmap'
as_grob(x, ..., device = NULL)

## S3 method for class 'HeatmapList'
as_grob(x, ..., device = NULL)

## S3 method for class 'HeatmapAnnotation'
as_grob(x, ..., device = NULL)
```

Arguments

x	An object to be converted into a grob .
...	Additional arguments passed to draw() .
device	A function that opens a graphics device for temporary rendering. By default this is an off-screen, in-memory device based on the pdf device, but this default device may not be satisfactory when using custom fonts.

Value

A [grob](#) object.

as_grob method collections

- [as_grob.grob\(\)](#)
- [as_grob.gList\(\)](#)
- [as_grob.patch_ggplot\(\)](#)
- [as_grob.ggplot\(\)](#)

- [as_grob.ggalignment::alignpatches\(\)](#)
- [as_grob.patchwork\(\)](#)
- [as_grob.patch\(\)](#)
- [as_grob.formula\(\)](#)
- [as_grob.recordedplot\(\)](#)
- [as_grob.trellis\(\)](#)
- [as_grob.Heatmap\(\)](#)
- [as_grob.pheatmap\(\)](#)

See Also

- [Heatmap\(\)](#)
- [HeatmapAnnotation\(\)](#)

Other as_grob: [as_grob.formula\(\)](#), [as_grob.gList\(\)](#), [as_grob.ggalignment::alignpatches\(\)](#), [as_grob.ggplot\(\)](#), [as_grob.grob\(\)](#), [as_grob.patch\(\)](#), [as_grob.patch_ggplot\(\)](#), [as_grob.patchwork\(\)](#), [as_grob.pheatmap\(\)](#), [as_grob.recordedplot\(\)](#), [as_grob.trellis\(\)](#)

as_grob.patch	<i>Convert Object into a Grob</i>
---------------	-----------------------------------

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'patch'
as_grob(x, ...)
```

Arguments

x	An object to be converted into a grob .
...	Additional arguments passed on to patchGrob .

Value

A [grob](#) object.

as_grob method collections

- [as_grob.grob\(\)](#)
- [as_grob.gList\(\)](#)
- [as_grob.patch_ggplot\(\)](#)
- [as_grob.ggplot\(\)](#)
- [as_grob.ggalignment::alignpatches\(\)](#)
- [as_grob.patchwork\(\)](#)
- [as_grob.patch\(\)](#)
- [as_grob.formula\(\)](#)
- [as_grob.recordedplot\(\)](#)
- [as_grob.trellis\(\)](#)
- [as_grob.Heatmap\(\)](#)
- [as_grob.pheatmap\(\)](#)

See Also

[patch](#)

Other as_grob: [as_grob.Heatmap\(\)](#), [as_grob.formula\(\)](#), [as_grob.gList\(\)](#), [as_grob.ggalignment::alignpatches\(\)](#), [as_grob.ggplot\(\)](#), [as_grob.grob\(\)](#), [as_grob.patch_ggplot\(\)](#), [as_grob.patchwork\(\)](#), [as_grob.pheatmap\(\)](#), [as_grob.recordedplot\(\)](#), [as_grob.trellis\(\)](#)

as_grob.patchwork	<i>Convert Object into a Grob</i>
-------------------	-----------------------------------

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'patchwork'
as_grob(x, ...)
```

Arguments

x	An object to be converted into a grob .
...	Additional arguments passed on to patchworkGrob .

Value

A [grob](#) object.

as_grob method collections

- [as_grob.grob\(\)](#)
- [as_grob.gList\(\)](#)
- [as_grob.patch_ggplot\(\)](#)
- [as_grob.ggplot\(\)](#)
- [as_grob.ggalien::alignpatches\(\)](#)
- [as_grob.patchwork\(\)](#)
- [as_grob.patch\(\)](#)
- [as_grob.formula\(\)](#)
- [as_grob.recordedplot\(\)](#)
- [as_grob.trellis\(\)](#)
- [as_grob.Heatmap\(\)](#)
- [as_grob.pheatmap\(\)](#)

See Also

[patchwork](#)

Other as_grob: [as_grob.Heatmap\(\)](#), [as_grob.formula\(\)](#), [as_grob.gList\(\)](#), [as_grob.ggalien::alignpatches\(\)](#), [as_grob.ggplot\(\)](#), [as_grob.grob\(\)](#), [as_grob.patch\(\)](#), [as_grob.patch_ggplot\(\)](#), [as_grob.pheatmap\(\)](#), [as_grob.recordedplot\(\)](#), [as_grob.trellis\(\)](#)

as_grob.pheatmap	<i>Convert Object into a Grob</i>
------------------	-----------------------------------

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'pheatmap'
as_grob(x, ...)
```

Arguments

x	An object to be converted into a grob .
...	Additional arguments passed to specific methods.

Value

A [grob](#) object.

as_grob method collections

- `as_grob.grob()`
- `as_grob.gList()`
- `as_grob.patch_ggplot()`
- `as_grob.ggplot()`
- `as_grob.ggalien::alignpatches()`
- `as_grob.patchwork()`
- `as_grob.patch()`
- `as_grob.formula()`
- `as_grob.recordedplot()`
- `as_grob.trellis()`
- `as_grob.Heatmap()`
- `as_grob.pheatmap()`

See Also

`pheatmap()`

Other as_grob: `as_grob.Heatmap()`, `as_grob.formula()`, `as_grob.gList()`, `as_grob.ggalien::alignpatches()`, `as_grob.ggplot()`, `as_grob.grob()`, `as_grob.patch()`, `as_grob.patch_ggplot()`, `as_grob.patchwork()`, `as_grob.recordedplot()`, `as_grob.trellis()`

`as_grob.recordedplot` *Convert Object into a Grob*

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'recordedplot'
as_grob(x, ..., device = NULL)
```

Arguments

- | | |
|---------------------|---|
| <code>x</code> | An object to be converted into a grob . |
| <code>...</code> | Additional arguments passed to specific methods. |
| <code>device</code> | A function that opens a graphics device for <code>grid.echo()</code> to work on. By default this is an off-screen, in-memory device based on the pdf device. This default device may not be satisfactory when using custom fonts. |

Value

A `grob` object.

as_grob method collections

- `as_grob.grob()`
- `as_grob.gList()`
- `as_grob.patch_ggplot()`
- `as_grob.ggplot()`
- `as_grob.ggalien::alignpatches()`
- `as_grob.patchwork()`
- `as_grob.patch()`
- `as_grob.formula()`
- `as_grob.recordedplot()`
- `as_grob.trellis()`
- `as_grob.Heatmap()`
- `as_grob.pheatmap()`

See Also

`recordPlot()`

Other `as_grob`: `as_grob.Heatmap()`, `as_grob.formula()`, `as_grob.gList()`, `as_grob.ggalien::alignpatches()`, `as_grob.ggplot()`, `as_grob.grob()`, `as_grob.patch()`, `as_grob.patch_ggplot()`, `as_grob.patchwork()`, `as_grob.pheatmap()`, `as_grob.trellis()`

`as_grob.trellis`*Convert Object into a Grob*

Description

Convert Object into a Grob

Usage

```
## S3 method for class 'trellis'  
as_grob(x, ..., device = NULL)
```

Arguments

<code>x</code>	An object to be converted into a grob .
<code>...</code>	Arguments passed on to grid::grid.grabExpr
<code>warn</code>	An integer specifying the amount of warnings to emit. 0 means no warnings, 1 means warn when it is certain that the grab will not faithfully represent the original scene. 2 means warn if there's any possibility that the grab will not faithfully represent the original scene.
<code>wrap</code>	A logical indicating how the output should be captured. If TRUE, each non-grob element on the display list is captured by wrapping it in a grob.
<code>wrap.grobs</code>	A logical indicating whether, if we are wrapping elements (<code>wrap=TRUE</code>), we should wrap grobs (or just wrap viewports).
<code>width,height</code>	Size of the device used for temporary rendering.
<code>device</code>	A function that opens a graphics device for temporary rendering. By default this is an off-screen, in-memory device based on the pdf device, but this default device may not be satisfactory when using custom fonts.

Value

A [grob](#) object.

as_grob method collections

- [as_grob.grob\(\)](#)
- [as_grob.gList\(\)](#)
- [as_grob.patch_ggplot\(\)](#)
- [as_grob.ggplot\(\)](#)
- [as_grob.ggalen::alignpatches\(\)](#)
- [as_grob.patchwork\(\)](#)
- [as_grob.patch\(\)](#)
- [as_grob.formula\(\)](#)
- [as_grob.recordedplot\(\)](#)
- [as_grob.trellis\(\)](#)
- [as_grob.Heatmap\(\)](#)
- [as_grob.pheatmap\(\)](#)

See Also

[trellis](#)

Other `as_grob`: [as_grob.Heatmap\(\)](#), [as_grob.formula\(\)](#), [as_grob.gList\(\)](#), [as_grob.ggalen::alignpatches\(\)](#), [as_grob.ggplot\(\)](#), [as_grob.grob\(\)](#), [as_grob.patch\(\)](#), [as_grob.patch_ggplot\(\)](#), [as_grob.patchwork\(\)](#), [as_grob.pheatmap\(\)](#), [as_grob.recordedplot\(\)](#)

Description

`circle_genomic()` constructs a circular layout specifically for genomic data. It is a specialized variant of `circle_continuous()` that applies default axis limits and coerces the first column of each plot's data to use chromosome (seqname) identifiers-matching those in the layout data-as factor levels.

Usage

```
circle_genomic(
  data,
  ...,
  radial = NULL,
  direction = "outward",
  sector_spacing = NULL,
  theme = NULL
)
```

Arguments

<code>data</code>	The input data, which can be: • A character string ("hg19" or "hg38") to load a predefined cytoband reference. • A <code>data.frame</code> with at least three columns: chromosome, start, and end positions. • A genomic object convertible via <code>fortify_data_frame()</code>.
<code>...</code>	Additional arguments passed to specific methods or <code>fortify_data_frame()</code> .
<code>radial</code>	A <code>coord_circle()/coord_radial()</code> object that defines the global parameters for coordinate across all plots in the layout. The parameters <code>start</code> , <code>end</code> , <code>direction</code> , and <code>expand</code> will be inherited and applied uniformly to all plots within the layout. The parameters <code>theta</code> and <code>r.axis.inside</code> will always be ignored and will be set to "x" and TRUE, respectively, for all plots.
<code>direction</code>	A single string of "inward" or "outward", indicating the direction in which the plot is added. • outward: The plot is added from the inner to the outer. • inward: The plot is added from the outer to the inner.
<code>sector_spacing</code>	The size of spacing between different panel. A numeric of the radians or a <code>rel()</code> object.
<code>theme</code>	A <code>theme()</code> object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, <code>panel.border</code> , and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.

Value

A circle_layout object representing the genomic layout.

circle_layout	<i>Arrange plots in a circular layout</i>
---------------	---

Description**[Experimental]**

If limits is provided, a continuous variable will be required and aligned in the direction specified (circle_continuous). Otherwise, a discrete variable will be required and aligned (circle_discrete).

Usage

```
circle_layout(
  data = NULL,
  ...,
  radial = NULL,
  direction = "outward",
  sector_spacing = NULL,
  limits = waiver(),
  theme = NULL,
  spacing_theta = deprecated()
)
```

```
circle_discrete(
  data = NULL,
  ...,
  radial = NULL,
  direction = "outward",
  sector_spacing = NULL,
  theme = NULL,
  spacing_theta = deprecated()
)
```

```
circle_continuous(
  data = NULL,
  ...,
  radial = NULL,
  direction = "outward",
  sector_spacing = NULL,
  limits = NULL,
  theme = NULL,
  spacing_theta = deprecated()
)
```

Arguments

data	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout: • If <code>limits</code> is not provided, <code>fortify_matrix()</code> will be used to get a matrix. • If <code>limits</code> is specified, <code>fortify_data_frame()</code> will be used to get a data frame.
...	Additional arguments passed to <code>fortify_data_frame()</code> or <code>fortify_matrix()</code> .
radial	A <code>coord_circle()/coord_radial()</code> object that defines the global parameters for coordinate across all plots in the layout. The parameters <code>start</code> , <code>end</code> , <code>direction</code> , and <code>expand</code> will be inherited and applied uniformly to all plots within the layout. The parameters <code>theta</code> and <code>r.axis.inside</code> will always be ignored and will be set to "x" and TRUE, respectively, for all plots.
direction	A single string of "inward" or "outward", indicating the direction in which the plot is added. • outward: The plot is added from the inner to the outer. • inward: The plot is added from the outer to the inner.
sector_spacing	The size of spacing between different panel. A numeric of the radians or a <code>rel()</code> object.
limits	A <code>continuous_limits()</code> object specifying the left/lower limit and the right/upper limit of the scale. Used to align the continuous axis.
theme	A <code>theme()</code> object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, <code>panel.border</code> , and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.
spacing_theta	[Deprecated] Please use <code>sector_spacing</code> instead.

Value

A `CircleLayout` object.

Examples

```
set.seed(123)

small_mat <- matrix(rnorm(56), nrow = 7)
rownames(small_mat) <- paste0("row", seq_len(nrow(small_mat)))
colnames(small_mat) <- paste0("column", seq_len(ncol(small_mat)))

# circle_layout
# same for circle_discrete()
circle_layout(small_mat) +
  galign() +
  geom_tile(aes(y = .column_index, fill = value)) +
  scale_fill_viridis_c() +
  align_dendro(aes(color = branch), k = 3L) +
  scale_color_brewer(palette = "Dark2")
```

```

# same for circle_continuous()
circle_layout(mpg, limits = continuous_limits(c(3, 5))) +
  ggalign(mapping = aes(displ, hwy, colour = class)) +
  geom_point(size = 2) +
  ggalign(mapping = aes(displ, hwy, colour = class)) +
  geom_point(size = 2) &
  scale_color_brewer(palette = "Dark2") &
  theme_bw()

# circle_discrete()
# direction outward
circle_discrete(small_mat) +
  align_dendro(aes(color = branch), k = 3L) +
  scale_color_brewer(palette = "Dark2") +
  ggalign() +
  geom_tile(aes(y = .column_index, fill = value)) +
  scale_fill_viridis_c()

# direction inward
circle_discrete(small_mat, direction = "inward") +
  ggalign() +
  geom_tile(aes(y = .column_index, fill = value)) +
  scale_fill_viridis_c() +
  align_dendro(aes(color = branch), k = 3L) +
  scale_color_brewer(palette = "Dark2")

# circle_continuous()
circle_continuous(mpg, limits = continuous_limits(c(3, 5))) +
  ggalign(mapping = aes(displ, hwy, colour = class)) +
  geom_point(size = 2) +
  ggalign(mapping = aes(displ, hwy, colour = class)) +
  geom_point(size = 2) &
  scale_color_brewer(palette = "Dark2") &
  theme_bw()

```

circle_switch

Determine the active context of circle layout

Description

[Stable]

Usage

```
circle_switch(radial = waiver(), direction = NULL, what = waiver(), ...)
```

Arguments

radial A `coord_circle()/coord_radial()` object that defines the global parameters for coordinate across all plots in the layout. The parameters start, end,

	direction, and expand will be inherited and applied uniformly to all plots within the layout. The parameters <code>theta</code> and <code>r.axis.inside</code> will always be ignored and will be set to "x" and TRUE, respectively, for all plots.
direction	A single string of "inward" or "outward", indicating the direction in which the plot is added. • outward: The plot is added from the inner to the outer. • inward: The plot is added from the outer to the inner.
what	What should get activated for the <code>circle_layout()</code> ? A single number or string of the plot elements in the layout. If NULL, will remove any active context.
...	These dots are for future extensions and must be empty.

Value

A `circle_switch` object which can be added to `circle_layout()`.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 7)
rownames(small_mat) <- paste0("row", seq_len(nrow(small_mat)))
colnames(small_mat) <- paste0("column", seq_len(ncol(small_mat)))
circle_discrete(small_mat) +
  ggalign() +
  geom_tile(aes(y = .column_index, fill = value)) +
  scale_fill_viridis_c() +
  align_dendro(aes(color = branch), k = 3L) +
  scale_color_brewer(palette = "Dark2")
```

continuous_limits	<i>Set continuous limits for the layout</i>
-------------------	---

Description

To align continuous axes, it is important to keep the limits consistent across all plots in the layout. You can set the limits by passing a function directly to the `limits` or `xlim/ylim` argument, using ... only. Alternatively, you can add a `ContinuousDomain` object to the layout. For the `quad_layout()` function, you must specify x/y arguments. For other layouts, you should pass the limits using ... directly.

Usage

```
continuous_limits(...)
```

Arguments

... A list of two numeric values, specifying the left/lower limit and the right/upper limit of the scale.

coord_circle

Polar Coordinates with Enhanced Controls

Description

An extended version of [coord_radial\(\)](#), providing additional customization options.

Usage

```
coord_circle(
  theta = "x",
  start = 0,
  end = NULL,
  thetalim = NULL,
  rlim = NULL,
  expand = FALSE,
  direction = 1,
  clip = "off",
  r.axis.inside = NULL,
  rotate.angle = FALSE,
  inner.radius = 0,
  outer.radius = 0.95
)
```

Arguments

theta	variable to map angle to (x or y)
start	Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction.
end	Position from 12 o'clock in radians where plot ends, to allow for partial polar coordinates. The default, NULL, is set to $\text{start} + 2 * \pi$.
thetalim, rlim	Limits for the theta and r axes.
expand	If TRUE, the default, adds a small expansion factor to the limits to prevent overlap between data and axes. If FALSE, limits are taken directly from the scale.
direction	1, clockwise; -1, anticlockwise
clip	Should drawing be clipped to the extent of the plot panel? A setting of "on" (the default) means yes, and a setting of "off" means no. For details, please see coord_cartesian() .
r.axis.inside	One of the following: • NULL (default) places the axis next to the panel if start and end arguments form a full circle and inside the panel otherwise. • TRUE to place the radius axis inside the panel. • FALSE to place the radius axis next to the panel.

- A numeric value, setting a theta axis value at which the axis should be placed inside the panel. Can be given as a length 2 vector to control primary and secondary axis placement separately.
- rotate.angle If TRUE, transforms the angle aesthetic in data in accordance with the computed theta position. If FALSE (default), no such transformation is performed. Can be useful to rotate text geoms in alignment with the coordinates.
- inner.radius A numeric in $[0, 1)$ indicates the inner radius.
- outer.radius A numeric in $(0, 1]$ indicates the outer radius. `coord_radial()` by default uses 0.8.

Examples

```
ggplot(mtcars, aes(displacement, mpg)) +
  geom_point() +
  coord_circle(
    start = -0.4 * pi, end = 0.4 * pi,
    inner.radius = 0.3, outer.radius = 1
  )
ggplot(mtcars, aes(displacement, mpg)) +
  geom_point() +
  coord_circle(
    start = -0.4 * pi, end = 0.4 * pi,
    inner.radius = 0.3, outer.radius = 0.5
  )
```

cross_link

Add a plot to connect selected observations

Description

Add a plot to connect selected observations

Usage

```
cross_link(
  link,
  data = waiver(),
  ...,
  on_top = TRUE,
  obs_size = 1,
  inherit_index = NULL,
  inherit_panel = NULL,
  inherit_nobs = NULL,
  size = NULL,
  active = NULL
)
```

Arguments

link	A <code>link_draw()</code> object that defines how to draw the links, such as <code>link_line()</code> .
data	The dataset to use for the layout. By default, <code>fortify_matrix()</code> will convert the data to a matrix. This argument allows you to change the layout data. If not specified, the original data will be used.
...	<dyn-dots> Additional arguments passed to <code>fortify_matrix()</code> .
on_top	A boolean value indicating whether to draw the link on top of the plot panel (TRUE) or below (FALSE).
obs_size	A single numeric value that indicates the size of a single observation, ranging from (0, 1].
inherit_index	A boolean value indicating whether to inherit the ordering index. If TRUE, will match the layout ordering index with the data names.
inherit_panel	A boolean value indicating whether to inherit the panel group. If TRUE, will match the layout panel with the data names.
inherit_nobs	A boolean value indicating whether to inherit the number of observations (nobs). If TRUE, the data input must be compatible with the layout data.
size	The relative size of the plot, can be specified as a <code>unit()</code> . Note that for <code>circle_layout()</code> , all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.
active	A <code>active()</code> object that defines the context settings when added to a layout.

ggplot2 Specification

The `cross_link` function initializes a ggplot object but does not initialize any data. Using `scheme_data()` to change the internal data if needed.

cross_mark	<i>Add a plot to annotate observations</i>
------------	--

Description

Add a plot to annotate observations

Usage

```
cross_mark(
  mark,
  data = waiver(),
  ...,
  obs_size = 1,
  inherit_index = NULL,
  inherit_panel = NULL,
  inherit_nobs = NULL,
  size = NULL,
  active = NULL
)
```

Arguments

mark	A <code>mark_draw()</code> object to define how to draw the links. Like <code>mark_line()</code> , <code>mark_tetragon()</code> . Note the names of the pair links will be used to define the panel names so must be unique.
data	The dataset to use for the layout. By default, <code>fortify_matrix()</code> will convert the data to a matrix. This argument allows you to change the layout data. If not specified, the original data will be used.
...	<dyn-dots> Additional arguments passed to <code>fortify_matrix()</code> .
obs_size	A single numeric value that indicates the size of a single observation, ranging from (0, 1].
inherit_index	A boolean value indicating whether to inherit the ordering index. If TRUE, will match the layout ordering index with the data names.
inherit_panel	A boolean value indicating whether to inherit the panel group. If TRUE, will match the layout panel with the data names.
inherit_nobs	A boolean value indicating whether to inherit the number of observations (nobs). If TRUE, the data input must be compatible with the layout data.
size	The relative size of the plot, can be specified as a <code>unit()</code> . Note that for <code>circle_layout()</code> , all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.
active	A <code>active()</code> object that defines the context settings when added to a layout.

ggplot2 Specification

The `cross_mark` function initializes a ggplot object. The underlying data contains following columns:

- `.panel`: the panel for the aligned axis. It means x-axis for vertical stack layout (including top and bottom annotation), y-axis for horizontal stack layout (including left and right annotation).
- `.names` (`vec_names()`) and `.index` (`vec_size()/NROW()`): a character names (only applicable when names exists) and an integer of index of the original data.
- `.hand`: A factor with levels `c("left", "right")` for horizontal stack layouts, or `c("top", "bottom")` for vertical stack layouts, indicating the position of the linked observations.

You can use `scheme_data()` to modify the internal data if needed.

cross_none

Reset layout ordering and panel group

Description

Reset layout ordering and panel group

Usage

```
cross_none(
  data = waiver(),
  ...,
  inherit_index = NULL,
  inherit_panel = NULL,
  inherit_nobs = NULL
)
```

Arguments

<code>data</code>	The dataset to use for the layout. By default, <code>fortify_matrix()</code> will convert the data to a matrix. This argument allows you to change the layout data. If not specified, the original data will be used.
<code>...</code>	<dyn-dots> Additional arguments passed to <code>fortify_matrix()</code> .
<code>inherit_index</code>	A boolean value indicating whether to inherit the ordering index. If TRUE, will match the layout ordering index with the data names.
<code>inherit_panel</code>	A boolean value indicating whether to inherit the panel group. If TRUE, will match the layout panel with the data names.
<code>inherit_nobs</code>	A boolean value indicating whether to inherit the number of observations (nobs). If TRUE, the data input must be compatible with the layout data.

<code>draw_key_gshape</code>	<i>Key glyphs for legends</i>
------------------------------	-------------------------------

Description

Each geom has an associated function that draws the key when the geom needs to be displayed in a legend. These functions are called `draw_key_*`(), where `*` stands for the name of the respective key glyph. The key glyphs can be customized for individual geoms by providing a geom with the `key_glyph` argument. The `draw_key_gshape` function provides this interface for custom key glyphs used with `geom_gshape()`.

Usage

```
draw_key_gshape(data, params, size)
```

Arguments

<code>data</code>	A single row data frame containing the scaled aesthetics to display in this key
<code>params</code>	A list of additional parameters supplied to the geom.
<code>size</code>	Width and height of key in mm.

Value

A grid grob.

Examples

```
p <- ggplot(economics, aes(date, psavert, color = "savings rate"))
# key glyphs can be specified by their name
p + geom_line(key_glyph = "timeseries")

# key glyphs can be specified via their drawing function
p + geom_line(key_glyph = draw_key_rect)
```

element_vec

Apply a function to the fields of an element object

Description

For an [element](#) object, some fields are vectorized, while others are not. This function allows you to apply a function to the vectorized fields.

The following helper functions are available:

- `element_vec_fields`: Identify which fields are vectorized. Developers should implement this when creating new element classes.
- `element_vec`: Apply a custom function `.fn` to vectorized fields.
- `element_rep`: Applies [rep\(\)](#).
- `element_rep_len`: Applies [rep_len\(\)](#).
- `element_vec_recycle`: Applies [vec_recycle\(\)](#).
- `element_vec_rep`: Applies [vec_rep\(\)](#).
- `element_vec_rep_each`: Applies [vec_rep_each\(\)](#).
- `element_vec_slice`: Applies [vec_slice\(\)](#).

Usage

```
element_vec_fields(.el, ...)
```

```
element_vec(.el, .fn, ...)
```

```
element_rep(.el, ...)
```

```
element_rep_len(.el, length.out, ...)
```

```
element_vec_recycle(.el, size, ...)
```

```
element_vec_rep(.el, times, ...)
```

```
element_vec_rep_each(.el, times, ...)
```

```
element_vec_slice(.el, i, ...)
```

Arguments

<code>.el</code>	An element object.
<code>...</code>	Additional arguments passed on to <code>fn</code> .
<code>.fn</code>	The function to be applied to the vectorized fields of the element object.
<code>length.out</code>	Non-negative integer. The desired length of the output vector. Other inputs will be coerced to a double vector and the first element taken. Ignored if NA or invalid.
<code>size</code>	Desired output size.
<code>times</code>	For <code>vec_rep()</code> , a single integer for the number of times to repeat the entire vector. For <code>vec_rep_each()</code> , an integer vector of the number of times to repeat each element of <code>x</code> . <code>times</code> will be recycled to the size of <code>x</code> .
<code>i</code>	An integer, character or logical vector specifying the locations or names of the observations to get/set. Specify TRUE to index all elements (as in <code>x[[]]</code>), or NULL, FALSE or <code>integer()</code> to index none (as in <code>x[NULL]</code>).

facet_sector

Polar coordinates with Facet support

Description

Draw each panel in a sector of the polar coordinate system. If `facet_sector()` is used in a `ggplot`, the coordinate system must be created with [coord_circle\(\)](#) or [coord_radial\(\)](#).

Usage

```
facet_sector(
  facets,
  sector_spacing = pi/180,
  drop = TRUE,
  radial = deprecated(),
  spacing_theta = deprecated()
)
```

Arguments

<code>facets</code>	A set of variables or expressions quoted by vars() and defining faceting groups on the rows or columns dimension. The variables can be named (the names are passed to <code>labeller</code>). For compatibility with the classic interface, can also be a formula or character vector. Use either a one sided formula, <code>~a + b</code> , or a character vector, <code>c("a", "b")</code> .
<code>sector_spacing</code>	The size of spacing between different panel. A numeric of the radians or a rel() object.

drop	If TRUE, the default, all factor levels not used in the data will automatically be dropped. If FALSE, all factor levels will be shown, regardless of whether or not they appear in the data.
radial	[Deprecated] Please add the coordinate system directly to the ggplot instead.
spacing_theta	[Deprecated] Please use sector_spacing instead.

Examples

```
ggplot(mtcars, aes(displacement, mpg)) +  
  geom_point() +  
  facet_sector(vars(cyl)) +  
  coord_circle(  
    start = -0.4 * pi, end = 0.4 * pi, inner.radius = 0.3,  
    outer.radius = 0.8, expand = TRUE  
  )
```

fortify_data_frame	<i>Build a data frame</i>
--------------------	---------------------------

Description

[Stable]

This function converts various objects to a data frame.

Usage

```
fortify_data_frame(data, ..., data_arg = NULL, call = NULL)
```

Arguments

data	An object to be converted to a data frame.
...	Arguments passed to methods.
data_arg	The argument name for data. Developers can use it to improve messages. Not used by the user.
call	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Value

A data frame.

fortify_data_frame method collections

- `fortify_data_frame.default()`
- `fortify_data_frame.character()`
- `fortify_data_frame.GRanges()`
- `fortify_data_frame.matrix()`
- `fortify_data_frame.dendrogram()`
- `fortify_data_frame.phylo()`

`fortify_data_frame.character`

Build a data frame

Description

[Stable]

This function converts various objects to a data frame.

Usage

```
## S3 method for class 'character'
fortify_data_frame(data, ..., data_arg = NULL, call = NULL)

## S3 method for class 'factor'
fortify_data_frame(data, ..., data_arg = NULL, call = NULL)

## S3 method for class 'numeric'
fortify_data_frame(data, ..., data_arg = NULL, call = NULL)

## S3 method for class 'logical'
fortify_data_frame(data, ..., data_arg = NULL, call = NULL)

## S3 method for class 'complex'
fortify_data_frame(data, ..., data_arg = NULL, call = NULL)
```

Arguments

<code>data</code>	An object to be converted to a data frame.
<code>...</code>	These dots are for future extensions and must be empty.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Value

A data frame with following columns:

- .names: the names for the vector (only applicable if names exist).
- value: the actual value of the vector.

See Also

Other `fortify_data_frame()` methods: `fortify_data_frame.GRanges()`, `fortify_data_frame.default()`, `fortify_data_frame.dendrogram()`, `fortify_data_frame.matrix()`, `fortify_data_frame.phylo()`

`fortify_data_frame.default`*Build a data frame*

Description

[Stable]

This function converts various objects to a data frame.

Usage

```
## Default S3 method:  
fortify_data_frame(data, ..., data_arg = NULL, call = NULL)
```

Arguments

<code>data</code>	An object to be converted to a data frame.
<code>...</code>	Additional arguments passed to <code>fortify()</code> .
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Details

By default, it calls `fortify()` to build the data frame.

See Also

Other `fortify_data_frame()` methods: `fortify_data_frame.GRanges()`, `fortify_data_frame.character()`, `fortify_data_frame.dendrogram()`, `fortify_data_frame.matrix()`, `fortify_data_frame.phylo()`

fortify_data_frame.dendrogram

Build a data frame

Description

[Stable]

This function converts various objects to a data frame.

Usage

```
## S3 method for class 'dendrogram'
fortify_data_frame(
  data,
  ...,
  priority = "right",
  center = FALSE,
  type = "rectangle",
  leaf_pos = NULL,
  leaf_braches = NULL,
  reorder_branches = TRUE,
  branch_gap = NULL,
  root = NULL,
  double = TRUE,
  data_arg = NULL,
  call = NULL
)

## S3 method for class 'hclust'
fortify_data_frame(data, ...)
```

Arguments

data	A hclust or a dendrogram object.
...	Additional arguments passed to dendrogram method.
priority	A string of "left" or "right". if we draw from right to left, the left will override the right, so we take the "left" as the priority. If we draw from left to right, the right will override the left, so we take the "right" as priority. This is used by align_dendro() to provide support of facet operation in ggplot2.
center	A boolean value. if TRUE, nodes are plotted centered with respect to all leaves/tips in the branch. Otherwise (default), plot them in the middle of the direct child nodes.
type	A string indicates the plot type, "rectangle" or "triangle".
leaf_pos	The x-coordinates of the leaf node. Must be the same length of the number of observations in data.

leaf_braches	Branches of the leaf node. Must be the same length of the number of observations in data. Usually come from cutree .
reorder_branches	A single boolean value, indicates whether reorder the provided leaf_braches based on the actual index.
branch_gap	A single numeric value indicates the gap between different branches.
root	A length one string or numeric indicates the root branch.
double	A single logical value indicating whether horizontal lines should be doubled when segments span multiple branches. If TRUE, the horizontal lines will be repeated for each branch that the segment spans. If FALSE, only one horizontal line will be drawn. This is used by align_dendro() to provide support of facet operation in ggplot2.
data_arg	The argument name for data. Developers can use it to improve messages. Not used by the user.
call	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Value

A data frame with the node coordinates:

- .panel: Similar with panel column, but always give the correct panel for usage of the ggplot facet.
- .index: the original index in the tree for the the node
- label: node label text
- x and y: x-axis and y-axis coordinates for the node
- branch: which branch the node is. You can use this column to color different groups.
- panel: which panel the node is, if we split the plot into panel using [facet_grid](#), this column will show which panel the node is from. Note: some nodes may fall outside panel (between two panels), so there are possible NA values in this column.
- leaf: A logical value indicates whether the node is a leaf.

ggalign attributes

edge: A data frame for edge coordinates:

- .panel: Similar with panel column, but always give the correct panel for usage of the ggplot facet.
- x and y: x-axis and y-axis coordinates for the start node of the edge.
- xend and yend: the x-axis and y-axis coordinates of the terminal node for edge.
- branch: which panel the edge is. You can use this column to color different groups.
- panel1 and panel2: The panel1 and panel2 columns have the same functionality as panel, but they are specifically for the edge data and correspond to both nodes of each edge.

See Also

Other `fortify_data_frame()` methods: `fortify_data_frame.GRanges()`, `fortify_data_frame.character()`, `fortify_data_frame.default()`, `fortify_data_frame.matrix()`, `fortify_data_frame.phylo()`

Examples

```
fortify_data_frame(hclust(dist(USArrests), "ave"))
```

```
fortify_data_frame.GRanges
```

Build a data frame

Description

[Stable]

This function converts various objects to a data frame.

Usage

```
## S3 method for class 'GRanges'
fortify_data_frame(data, ..., data_arg = NULL, call = NULL)
```

Arguments

<code>data</code>	An object to be converted to a data frame.
<code>...</code>	Arguments passed to methods.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Value

A data frame with at least following columns:

- `seqnames`: The sequence (e.g., chromosome) names.
- `start`: The start positions of the ranges.
- `end`: The end positions of the ranges.
- `width`: The width of each range.

See Also

Other `fortify_data_frame()` methods: `fortify_data_frame.character()`, `fortify_data_frame.default()`, `fortify_data_frame.dendrogram()`, `fortify_data_frame.matrix()`, `fortify_data_frame.phylo()`

fortify_data_frame.matrix
Build a data frame

Description

[Stable]

This function converts various objects to a data frame.

Usage

```
## S3 method for class 'matrix'
fortify_data_frame(data, lvls = NULL, ..., data_arg = NULL, call = NULL)

## S3 method for class 'DelayedMatrix'
fortify_data_frame(data, ...)

## S3 method for class 'Matrix'
fortify_data_frame(data, ...)
```

Arguments

data	A matrix-like object.
lvls	A logical value indicating whether to restore factor levels using those stored in ggalign_lvls() , or a character vector specifying custom levels for the value column. If levels are provided or restored, the value column will be returned as a factor.
...	These dots are for future extensions and must be empty.
data_arg	The argument name for data. Developers can use it to improve messages. Not used by the user.
call	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Value

Matrix will be transformed into a long-form data frame, where each row represents a unique combination of matrix indices and their corresponding values. The resulting data frame will contain the following columns:

- `.row_names` and `.row_index`: the row names (only applicable when names exist) and an integer representing the row index of the original matrix.
- `.column_names` and `.column_index`: the column names (only applicable when names exist) and column index of the original matrix.
- `value`: the matrix value, returned as a factor if levels are specified or restored.

See Also

Other `fortify_data_frame()` methods: `fortify_data_frame.GRanges()`, `fortify_data_frame.character()`, `fortify_data_frame.default()`, `fortify_data_frame.dendrogram()`, `fortify_data_frame.phylo()`

`fortify_data_frame.phylo`

Build a data frame

Description**[Stable]**

This function converts various objects to a data frame.

Usage

```
## S3 method for class 'phylo'
fortify_data_frame(
  data,
  ...,
  priority = "right",
  center = FALSE,
  type = "rectangle",
  tree_type = NULL,
  tip_pos = NULL,
  tip_clades = NULL,
  reorder_clades = TRUE,
  clade_gap = NULL,
  root = NULL,
  double = TRUE,
  data_arg = NULL,
  call = NULL
)
```

Arguments

<code>data</code>	A hclust or a dendrogram object.
<code>...</code>	These dots are for future extensions and must be empty.
<code>priority</code>	A string of "left" or "right". if we draw from right to left, the left will override the right, so we take the "left" as the priority. If we draw from left to right, the right will override the left, so we take the "right" as priority. This is used by align_dendro() to provide support of facet operation in <code>ggplot2</code> .
<code>center</code>	A boolean value. if TRUE, nodes are plotted centered with respect to all leaves/tips in the branch. Otherwise (default), plot them in the middle of the direct child nodes.
<code>type</code>	A string indicates the plot type, "rectangle" or "triangle".

tree_type	A single string, one of "phylogram" or "cladogram", indicating the type of tree. • phylogram: Represents a phylogenetic tree where branch lengths indicate evolutionary distance or time. • cladogram: Represents a tree where branch lengths are not used, or the branches do not reflect evolutionary time. Usually, you don't need to modify this.
tip_pos	The x-coordinates of the tip. Must be the same length of the number of tips in tree.
tip_clades	Clades of the tips. Must be the same length of the number of tips in data.
reorder_clades	A single boolean value, indicates whether reorder the provided tip_clades based on the actual ordering index.
clade_gap	A single numeric value indicates the gap between different clades.
root	A length one string or numeric indicates the root branch.
double	A single logical value indicating whether horizontal lines should be doubled when segments span multiple branches. If TRUE, the horizontal lines will be repeated for each branch that the segment spans. If FALSE, only one horizontal line will be drawn. This is used by align_dendro() to provide support of facet operation in ggplot2.
data_arg	The argument name for data. Developers can use it to improve messages. Not used by the user.
call	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Value

A data frame with the node coordinates:

- .panel: Similar with panel column, but always give the correct panel for usage of the ggplot facet.
- .index: the original index in the tree for the the tip/node.
- label: the tip/node label text.
- x and y: x-axis and y-axis coordinates for the tip/node.
- clade: which clade the node is. You can use this column to color different clades.
- panel: which panel the node is, if we split the plot into panel using [facet_grid](#), this column will show which panel the node is from. Note: some nodes may fall outside panel (between two panels), so there are possible NA values in this column.
- tip: A logical value indicates whether current node is a tip.

ggalign attributes

edge: A data frame for edge coordinates:

- .panel: Similar with panel column, but always give the correct panel for usage of the ggplot facet.

- x and y: x-axis and y-axis coordinates for the start node of the edge.
- xend and yend: the x-axis and y-axis coordinates of the terminal node for edge.
- clade: which panel the edge is. You can use this column to color different groups.
- panel1 and panel2: The panel1 and panel2 columns have the same functionality as panel, but they are specifically for the edge data and correspond to both nodes of each edge.

See Also

Other `fortify_data_frame()` methods: `fortify_data_frame.GRanges()`, `fortify_data_frame.character()`, `fortify_data_frame.default()`, `fortify_data_frame.dendrogram()`, `fortify_data_frame.matrix()`

fortify_matrix	<i>Build a Matrix</i>
----------------	-----------------------

Description

[Stable]

This function converts various objects into a matrix format. By default, it calls `as.matrix()` to build a matrix.

Usage

```
fortify_matrix(data, ..., data_arg = NULL, call = NULL)
```

Arguments

<code>data</code>	An object to be converted into a matrix.
<code>...</code>	Additional arguments passed to methods.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Value

A matrix.

fortify_matrix method collections

- `fortify_matrix.default()`
- `fortify_matrix.list_upset()`
- `fortify_matrix.MAF()`
- `fortify_matrix.GISTIC()`
- `fortify_matrix.matrix()`
- `fortify_matrix.matrix_upset()`
- `fortify_matrix.matrix_oncoplot()`

`fortify_matrix.default`*Build a Matrix*

Description

By default, it calls `as.matrix()` to build a matrix.

Usage

```
## Default S3 method:  
fortify_matrix(data, ..., data_arg = NULL, call = NULL)
```

Arguments

<code>data</code>	An object to be converted into a matrix.
<code>...</code>	These dots are for future extensions and must be empty.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

Value

A matrix.

See Also

Other `fortify_matrix()` methods: `fortify_matrix.GISTIC()`, `fortify_matrix.MAF()`, `fortify_matrix.list_upset`, `fortify_matrix.matrix()`, `fortify_matrix.matrix_oncoplot()`, `fortify_matrix.matrix_upset()`

`fortify_matrix.GISTIC` *Build a matrix from a maftools object*

Description

Build a matrix from a maftools object

Usage

```
## S3 method for class 'GISTIC'
fortify_matrix(
  data,
  ...,
  n_top = NULL,
  bands = NULL,
  ignored_bands = NULL,
  sample_anno = NULL,
  remove_empty_samples = TRUE,
  data_arg = NULL,
  call = NULL
)
```

Arguments

<code>data</code>	A GISTIC object.
<code>...</code>	These dots are for future extensions and must be empty.
<code>n_top</code>	A single number indicates how many top bands to be drawn.
<code>bands</code>	An atomic character defines the bands to draw.
<code>ignored_bands</code>	An atomic character defines the bands to be ignored.
<code>sample_anno</code>	A data frame of sample clinical features to be added.
<code>remove_empty_samples</code>	A single boolean value indicating whether to drop samples without any genomic alterations.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

ggalign attributes

- `sample_anno`: sample clinical informations provided in `sample_anno`.
- `sample_summary`: sample copy number summary informations. See `data@cnv.summary` for details.
- `cytoband_summary`: cytoband summary informations. See `data@cytoband.summary` for details.
- `gene_summary`: gene summary informations. See `data@gene.summary` for details.
- `summary`: A data frame of summary information. See `data@summary` for details.

See Also

Other `fortify_matrix()` methods: `fortify_matrix.MAF()`, `fortify_matrix.default()`, `fortify_matrix.list_upse`, `fortify_matrix.matrix()`, `fortify_matrix.matrix_oncoplot()`, `fortify_matrix.matrix_upset()`

fortify_matrix.list_upset*Build a Matrix for UpSet plot*

Description

[Experimental]

This function converts a list into a matrix format suitable for creating an UpSet plot. It always returns a matrix for a horizontal UpSet plot.

Usage

```
## S3 method for class 'list_upset'  
fortify_matrix(data, mode = "distinct", ..., data_arg = NULL, call = NULL)
```

Arguments

data	A list of sets.
mode	A string of "distinct", "intersect", or "union" indicates the mode to define the set intersections. Check https://jokergoo.github.io/ComplexHeatmap-reference/book/upset-plot.html#upset-mode for details.
...	These dots are for future extensions and must be empty.
data_arg	The argument name for data. Developers can use it to improve messages. Not used by the user.
call	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

ggalign attributes

- intersection_sizes: An integer vector indicating the size of each intersection.
- set_sizes: An integer vector indicating the size of each set.

See Also

[tune.list\(\)](#)

Other `fortify_matrix()` methods: `fortify_matrix.GISTIC()`, `fortify_matrix.MAF()`, `fortify_matrix.default()`, `fortify_matrix.matrix()`, `fortify_matrix.matrix_oncoplot()`, `fortify_matrix.matrix_upset()`

fortify_matrix.MAF *Build a Matrix for OncoPrint*

Description

Convert MAF object to a matrix:

- `fortify_matrix.MAF`: Extract genomic alterations for genes.
- `fortify_matrix.MAF_pathways`: Extract genomic alterations for pathways. [tune.MAF\(\)](#) helps convert MAF object to a MAF_pathways object.

Usage

```
## S3 method for class 'MAF'
fortify_matrix(
  data,
  ...,
  genes = NULL,
  n_top = NULL,
  remove_empty_genes = TRUE,
  remove_empty_samples = TRUE,
  collapse_vars = TRUE,
  use_syn = TRUE,
  missing_genes = "error",
  data_arg = NULL,
  call = NULL
)

## S3 method for class 'MAF_pathways'
fortify_matrix(
  data,
  ...,
  pathdb = "smgdb",
  remove_empty_pathways = TRUE,
  remove_empty_samples = TRUE,
  data_arg = NULL,
  call = NULL
)
```

Arguments

<code>data</code>	A MAF object.
<code>...</code>	These dots are for future extensions and must be empty.
<code>genes</code>	An atomic character defines the genes to draw.
<code>n_top</code>	A single number indicates how many top genes to be drawn.

<code>remove_empty_genes</code>	A single boolean value indicates whether to drop genes without any genomic alterations.
<code>remove_empty_samples</code>	A single boolean value indicates whether to drop samples without any genomic alterations.
<code>collapse_vars</code>	A single boolean value indicating whether to collapse multiple alterations in the same sample and gene into a single value "Multi_Hit". Alternatively, you can provide a single string indicates the collapsed values.
<code>use_syn</code>	A single boolean value indicates whether to include synonymous variants when Classifies SNPs into transitions and transversions.
<code>missing_genes</code>	A string, either "error" or "remove", specifying the action for handling missing genes.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.
<code>pathdb</code>	A string of "smgbp" or "sigpw", or a named list of genes to define the pathways.
<code>remove_empty_pathways</code>	A single boolean value indicates whether to drop pathways without any genomic alterations.

ggalign attributes

For `fortify_matrix.MAF`:

- `gene_summary`: A data frame of gene summary informations. See `maftools::getGeneSummary()` for details.
- `sample_summary`: A data frame of sample summary informations. See `maftools::getSampleSummary()` for details.
- `sample_anno`: A data frame of sample clinical informations. See `maftools::getClinicalData()` for details.
- `variant_weights`: A data frame of variant weights. Each gene in a sample is assigned a total weight of 1. When multiple variants occur in the same gene-sample pair, the weight for each variant reflects its proportion of the total.
- `n_genes`: Total number of genes.
- `n_samples`: Total number of samples.
- `titv`: A list of data frame with Transitions and Transversions summary. See `maftools::titv()` for details.

The levels of Variant_Classification will be stored in `ggalign_lvls()`. If they do not exist, alphabetical ordering will be used.

For `fortify_matrix.MAF_pathways`:

- `gene_list`: the pathway contents.

- `pathway_summary`: pathway summary informations. See `maftools::pathways()` for details.
- `sample_summary`: sample summary informations. See `maftools::getSampleSummary()` for details.
- `sample_anno`: sample clinical informations. See `maftools::getClinicalData()` for details.

See Also

Other `fortify_matrix()` methods: `fortify_matrix.GISTIC()`, `fortify_matrix.default()`, `fortify_matrix.list_upset()`, `fortify_matrix.matrix()`, `fortify_matrix.matrix_oncoplot()`, `fortify_matrix.matrix_upset()`

`fortify_matrix.matrix` *Build a matrix*

Description

Build a matrix

Usage

```
## S3 method for class 'matrix'
fortify_matrix(data, ..., data_arg = NULL, call = NULL)
```

Arguments

<code>data</code>	A matrix object.
<code>...</code>	These dots are for future extensions and must be empty.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

shape

- `upset`: `fortify_matrix.matrix_upset()`
- `oncoplot`: `fortify_matrix.matrix_oncoplot()`

See Also

Other `fortify_matrix()` methods: `fortify_matrix.GISTIC()`, `fortify_matrix.MAF()`, `fortify_matrix.default()`, `fortify_matrix.list_upset()`, `fortify_matrix.matrix_oncoplot()`, `fortify_matrix.matrix_upset()`

```
fortify_matrix.matrix_oncoplot
```

Build a Matrix for OncoPrint

Description

Converts a matrix suitable for creating an OncoPrint. `tune.matrix()` helps convert matrix object to a `matrix_oncoplot` object.

Usage

```
## S3 method for class 'matrix_oncoplot'
fortify_matrix(
  data,
  ...,
  genes = NULL,
  n_top = NULL,
  remove_empty_genes = TRUE,
  remove_empty_samples = TRUE,
  missing_genes = "error",
  data_arg = NULL,
  call = NULL
)
```

Arguments

<code>data</code>	A matrix where each row represents an genes, and each column represents samples. The values in the matrix indicate whether the element is part of the set.
<code>...</code>	These dots are for future extensions and must be empty.
<code>genes</code>	An atomic character defines the genes to draw.
<code>n_top</code>	A single number indicates how many top genes to be drawn.
<code>remove_empty_genes</code>	A single boolean value indicats whether to drop genes without any genomic alterations.
<code>remove_empty_samples</code>	A single boolean value indicats whether to drop samples without any genomic alterations.
<code>missing_genes</code>	A string, either "error" or "remove", specifying the action for handling missing genes.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

ggalign attributes

- gene_summary: An integer vector of the altered samples for each gene.
- sample_summary: An integer vector of the altered genes for each sample.
- n_genes: Total number of genes.
- n_samples: Total number of samples.

See Also

`tune.matrix()`

Other `fortify_matrix()` methods: `fortify_matrix.GISTIC()`, `fortify_matrix.MAF()`, `fortify_matrix.default()`, `fortify_matrix.list_upset()`, `fortify_matrix.matrix()`, `fortify_matrix.matrix_upset()`

`fortify_matrix.matrix_upset`

Build a Matrix for UpSet plot

Description

Converts a matrix suitable for creating an UpSet plot. `tune.matrix()` helps convert matrix object to a `matrix_upset` object.

Usage

```
## S3 method for class 'matrix_upset'
fortify_matrix(data, ..., data_arg = NULL, call = NULL)
```

Arguments

<code>data</code>	A matrix where each row represents an element, and each column defines a set. The values in the matrix indicate whether the element is part of the set. Any non-missing value signifies that the element exists in the set.
<code>...</code>	Arguments passed on to <code>fortify_matrix.list_upset</code>
<code>mode</code>	A string of "distinct", "intersect", or "union" indicates the mode to define the set intersections. Check https://jokergoo.github.io/ComplexHeatmap-reference/book/upset-plot.html#upset-mode for details.
<code>data_arg</code>	The argument name for data. Developers can use it to improve messages. Not used by the user.
<code>call</code>	The execution environment where data and other arguments for the method are collected. Developers can use it to improve messages. Not used by the user.

ggalign attributes

- intersection_sizes: An integer vector indicating the size of each intersection.
- set_sizes: An integer vector indicating the size of each set.

See Also

[tune.matrix\(\)](#)

Other [fortify_matrix\(\)](#) methods: [fortify_matrix.GISTIC\(\)](#), [fortify_matrix.MAF\(\)](#), [fortify_matrix.default\(\)](#), [fortify_matrix.list_upset\(\)](#), [fortify_matrix.matrix\(\)](#), [fortify_matrix.matrix_oncoplot\(\)](#)

free_align

Free from alignment

Description

[align_plots](#) will try to align plot panels, and every elements of the plot, following functions remove these restrictions:

- `free_align`: if we want to compose plots without alignment of some panel axes (panel won't be aligned). we can wrap the plot with `free_align`.
- `free_border`: attaches borders (e.g., axis titles, tick marks) directly to the plot panel. This keeps them visually close to the panel during alignment.
- `free_lab()`: Similar to `free_border()`, but only attaches axis titles and tick labels, not full borders. It's mainly included for completeness; in most cases, combining `free_border()` and `free_space()` is sufficient.
- `free_space`: Removing the ggplot element sizes when aligning.
- `free_vp`: Customize the [viewport](#) when aligning.
- `free_guide`: If we want to override the behaviour of the overall guides behaviour, we can wrap the plot with `free_guide`.

Usage

```
free_align(plot, axes = "tlbr")
```

```
free_border(plot, borders = "tlbr")
```

```
free_guide(plot, guides = "tlbr")
```

```
free_lab(plot, labs = "tlbr")
```

```
free_space(plot, spaces = "tlbr")
```

```
free_vp(plot, x = 0.5, y = 0.5, width = NA, height = NA, ...)
```

Arguments

`plot` A [ggplot](#) or [alignpatches](#) object.

`axes` Which axes shouldn't be aligned? A string containing one or more of "t", "l", "b", and "r".

borders	Which border shouldn't be aligned? A string containing one or more of "t", "l", "b", and "r".
guides	A string containing one or more of "t", "l", "b", "r", and "i" indicates which side of guide legends should be collected for the plot. If NULL, no guide legends will be collected.
labs	Which axis labs to be free? A string containing one or more of "t", "l", "b", and "r".
spaces	Which border spaces should be removed? A string containing one or more of "t", "l", "b", and "r".
x	A numeric vector or unit object specifying x-location.
y	A numeric vector or unit object specifying y-location.
width	A numeric vector or unit object specifying width.
height	A numeric vector or unit object specifying height.
...	Arguments passed on to <code>grid::viewport</code>
default.units	A string indicating the default units to use if x, y, width, or height are only given as numeric vectors.
just	A string or numeric vector specifying the justification of the viewport relative to its (x, y) location. If there are two values, the first value specifies horizontal justification and the second value specifies vertical justification. Possible string values are: "left", "right", "centre", "center", "bottom", and "top". For numeric values, 0 means left alignment and 1 means right alignment.
gp	An object of class "gpar", typically the output from a call to the function <code>gpar</code> . This is basically a list of graphical parameter settings.
clip	One of "on", "inherit", or "off", indicating whether to clip to the extent of this viewport, inherit the clipping region from the parent viewport, or turn clipping off altogether. For back-compatibility, a logical value of TRUE corresponds to "on" and FALSE corresponds to "inherit". May also be a grob (or a gTree) that describes a clipping path or the result of a call to <code>as.path</code> .
mask	One of "none" (or FALSE) or "inherit" (or TRUE) or a grob (or a gTree) or the result of call to <code>as.mask</code> . This specifies that the viewport should have no mask, or it should inherit the mask of its parent, or it should have its own mask, as described by the grob.
xscale	A numeric vector of length two indicating the minimum and maximum on the x-scale. The limits may not be identical.
yscale	A numeric vector of length two indicating the minimum and maximum on the y-scale. The limits may not be identical.
angle	A numeric value indicating the angle of rotation of the viewport. Positive values indicate the amount of rotation, in degrees, anticlockwise from the positive x-axis.
layout	A Grid layout object which splits the viewport into subregions.
layout.pos.row	A numeric vector giving the rows occupied by this viewport in its parent's layout.

`layout.pos.col` A numeric vector giving the columns occupied by this viewport in its parent's layout.

`name` A character value to uniquely identify the viewport once it has been pushed onto the viewport tree.

Value

- `free_align`: A modified version of plot with a `galign_free_align` class.
- `free_border`: A modified version of plot with a `galign_free_border` class.
- `free_guide`: A modified version of plot with a `galign_free_guide` class.
- `free_lab`: A modified version of plot with a `galign_free_lab` class.
- `free_space`: A modified version of plot with a `galign_free_space` class.
- `free_vp`: A modified version of plot with a `galign_free_vp` class.

Examples

```
# directly copied from `patchwork`
# Sometimes you have a plot that defies good composition alignment, e.g. due
# to long axis labels
p1 <- ggplot(mtcars) +
  geom_bar(aes(y = factor(gear), fill = factor(gear))) +
  scale_y_discrete(
    "",
    labels = c(
      "3 gears are often enough",
      "But, you know, 4 is a nice number",
      "I would def go with 5 gears in a modern car"
    )
  )

# When combined with other plots it ends up looking bad
p2 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp))

align_plots(p1, p2, ncol = 1L)

# We can fix this by using `free_align`
align_plots(free_align(p1), p2, ncol = 1L)

# If we still want the panels to be aligned to the right, we can choose to
# free only the left side
align_plots(free_align(p1, axes = "l"), p2, ncol = 1L)

# We could use `free_lab` to fix the layout in a different way
align_plots(p1, free_lab(p2), ncol = 1L)

# `free_border` is similar with `free_lab`, they have a distinction in terms
```

```

# of placement on either the top or bottom side of the panel. Specifically,
# the top side contains the `title` and `subtitle`, while the bottom side
# contains the `caption`. free_lab() does not attach these elements in the
# panel area.
p3 <- ggplot(mtcars) +
  geom_point(aes(hp, wt, colour = mpg)) +
  ggtitle("Plot 3")
p_axis_top <- ggplot(mtcars) +
  geom_point(aes(mpg, disp)) +
  ggtitle("Plot axis in top") +
  scale_x_continuous(position = "top")
align_plots(p_axis_top, free_lab(p3))
align_plots(p_axis_top, free_border(p3))

# Another issue is that long labels can occupy much spaces
align_plots(NULL, p1, p2, p2)

# This can be fixed with `free_space`
align_plots(NULL, free_space(p1, "1"), p2, p2)

```

genomic_density

Calculate Genomic Region Density

Description

Computes the density or count of genomic regions in sliding or fixed windows across the genome. The density can be reported as the percentage of uncovered bases or the number of overlapping regions within each window.

Usage

```

genomic_density(
  region,
  window_size = 1e+07,
  n_window = NULL,
  overlap = TRUE,
  mode = c("coverage", "count"),
  seqlengths = NULL
)

```

Arguments

region	A data frame with at least 3 columns: chromosome, start, and end. - Column 1: character or factor, chromosome name. - Column 2: numeric, start position (must be <= end). - Column 3: numeric, end position.
--------	---

window_size	Numeric, the width of each window (default is 1e+07). Ignored if n_window is specified.
n_window	Integer, the number of windows per chromosome. If provided, overrides window_size and evenly splits the chromosome into n_window (non-overlapping) or 2*n_window - 1 (overlapping) windows.
overlap	Logical, whether to use overlapping windows (default TRUE). Overlapping windows are spaced by half the window size.
mode	Character, either "coverage" or "count": • "count": reports the number of regions overlapping each window. • "coverage": reports the fraction of each window covered by regions.
seqlengths	Optional named vector of chromosome lengths. If missing, the maximum end value in the input is used as the chromosome length.

Details

This function splits the input by chromosome and tiles the genomic space into windows, optionally overlapping. For each window, it calculates:

- the number of regions that overlap it (if mode = "count"), or
- the fraction of bases covered by any region (if mode = "percent").

Value

A data frame containing the first three columns from region, plus a fourth column density, which represents either the region count or the coverage percentage, depending on mode.

Examples

```
region <- data.frame(
  chr = rep("chr1", 3),
  start = c(100, 5000000, 15000000),
  end = c(2000000, 7000000, 17000000)
)
genomic_density(region, window_size = 1e7, mode = "count")
genomic_density(region, n_window = 3, overlap = FALSE, mode = "coverage")
```

genomic_dist

Calculate inter-region distances for genomic rainfall plots

Description

This function computes distances between adjacent genomic regions, grouped by chromosome. Useful for visualizing clustering or dispersion of genomic features.

Usage

```
genomic_dist(region, mode = NULL)
```

Arguments

region	A data frame with at least 3 columns: chromosome, start, and end.
mode	How to assign distance for intermediate regions: one of "min", "max", "mean", "left", or "right".

Details

The distance between two adjacent regions is calculated as the number of bases between the **end position of the upstream region** and the **start position of the downstream region**. If two regions overlap or are adjacent (≤ 1 bp apart), the distance is set to 0. The resulting distance is assigned to each region according to the selected mode:

- "left": assign the distance to the upstream region
- "right": assign to the downstream region
- "min" / "max" / "mean": for intermediate regions, calculate the minimum, maximum, or average of the distances to neighboring regions

Value

A data frame with an additional dist column.

geom_draw

Layer with Grid or Function

Description

Draw a ggplot2 layer using a grob or a function.

Usage

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

Arguments

draw	Either a grob object or a function (can be purrr-style) that accepts at least one argument (a data frame of transformed coordinates) and returns a grob .
mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot(). A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).
type	A single string of "group" or "panel", "group" draws geoms with <code>draw_group</code> , which displays multiple observations as one geometric object, and "panel" draws geoms with <code>draw_panel</code> , displaying individual graphical objects for each observation (row). Default: "group".
stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following: • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer() 's <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through Unknown arguments that are not part of the 4 categories below are ignored.

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

<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>show.legend</code>	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<code>inherit.aes</code>	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>annotation_borders()</code> .

Details

If you want to combine the functionality of multiple geoms, it can typically be achieved by preparing the data for each geom inside the `draw_*()` call and sending it off to the different geoms, collecting the output in a `grid::gList` (a list of grobs) for `draw_group()` or a `grid::gTree` (a grob containing multiple child grobs) for `draw_panel()`.

See Also

<https://ggplot2.tidyverse.org/reference/ggplot2-ggproto.html>

Examples

```
text <- grid::textGrob(
  "ggdraw",
  x = c(0, 0, 0.5, 1, 1),
  y = c(0, 1, 0.5, 0, 1),
  hjust = c(0, 0, 0.5, 1, 1),
  vjust = c(0, 1, 0.5, 0, 1)
```

```
)
ggplot(data.frame(x = 1, y = 2)) +
  geom_draw(text)
```

geom_gshape

Layer with a customized shape graphic using grid functions.

Description

[Questioning]

geom_gshape depends on the new aesthetics gshape (shape with grid functions), which should always be provided with [scale_gshape_manual\(\)](#), in which, we can provide a list of grobs or functions that define how each value should be drawn. Any ggplot2 aesthetics can be used as the arguments.

Usage

```
geom_gshape(
  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 | <p>The data to be displayed in this layer. There are three options:</p> <p>If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p> |
| stat | The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: |

	• A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*</code>() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*</code>() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of layer() may also be passed on through ... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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

Life cycle

We're unsure whether this function is truly necessary, which is why it is marked as questioning. So far, we've found that `geom_subrect()` and `geom_subtile()` handle most use cases effectively.

Aesthetics

`geom_gshape()` understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- `x`
- `y`
- `gshape`
- `alpha` → NA
- `colour` → "black"
- `fill` → NA
- `group` → inferred
- `linetype` → 1
- `linewidth` → 0.5
- `shape` → 19
- `size` → 1.5
- `stroke` → 0.5

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

Examples

```
library(grid)
ggplot(data.frame(value = letters[seq_len(5)], y = seq_len(5))) +
  geom_gshape(aes(x = 1, y = y, gshape = value, fill = value)) +
  scale_gshape_manual(values = list(
    a = function(x, y, width, height, fill) {
      rectGrob(x, y,
        width = width, height = height,
        gp = gpar(fill = fill),
        default.units = "native"
      )
    },
    b = function(x, y, width, height, fill) {
      rectGrob(x, y,
        width = width, height = height,
        gp = gpar(fill = fill),
        default.units = "native"
      )
    },
  ))
```

```

c = function(x, y, width, height, fill) {
  rectGrob(x, y,
    width = width, height = height,
    gp = gpar(fill = fill),
    default.units = "native"
  )
},
d = function(x, y, width, height, shape) {
  gList(
    pointsGrob(x, y, pch = shape),
    # To ensure the rectangle color is shown in the legends, you
    # must explicitly provide a color argument and include it in
    # the `gpar()` of the graphical object
    rectGrob(x, y, width, height,
      gp = gpar(col = "black", fill = NA)
    )
  )
},
e = function(xmin, xmax, ymin, ymax) {
  segmentsGrob(
    xmin, ymin,
    xmax, ymax,
    gp = gpar(lwd = 2)
  )
}
)) +
scale_fill_brewer(palette = "Dark2") +
theme_void()

```

geom_magick

Draw images as point shapes using magick

Description

Reads an image with **magick**, applies optional processing, and uses the result as the graphical shape for points in a plot.

This is useful when you want to replace the usual point symbols with arbitrary images while keeping full control over their placement, size, and interpolation.

Usage

```

geom_magick(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  magick = NULL,

```

```

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

```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot(). A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).
stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following: • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as <code>"jitter"</code>. • For more information and other ways to specify the position, see the layer position documentation.
...	These dots are for future extensions and must be empty.
magick	A function (purrr-style formula is accepted) that takes an image_read() object as input and returns an object compatible with as.raster() . You can use any of the <code>image_*</code> () functions from the magick package to process the raster image.

magick_params	Additional arguments passed on to magick
interpolate	A logical value indicating whether to linearly interpolate the image (the alternative is to use nearest-neighbour interpolation, which gives a more blocky result).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders() .

Aesthetics

geom_magick() understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- **x**
- **y**
- image
- **alpha** → NA
- angle → 0
- **fill** → NA
- **group** → inferred
- hjust → 0.5
- **size** → via theme()
- vjust → 0.5

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

Examples

```
set.seed(123)
d <- data.frame(
  x = rnorm(10),
  y = rnorm(10),
  image = "https://jeroenooms.github.io/images/frink.png",
  fill = sample(c("A", "B", "C", "D"), 10, replace = TRUE),
  alpha = rnorm(10, mean = 0.5, sd = 0.1)
)
d$alpha <- pmax(pmin(d$alpha, 1), 0)
ggplot(d, aes(x, y)) +
  geom_magick(aes(image = image, fill = fill, alpha = alpha))
```

geom_pie

Pie charts

Description

Pie charts

Usage

```
geom_pie(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  clockwise = TRUE,  
  steps = 100,  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 10,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

Arguments

- | | |
|---------|---|
| mapping | Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. |
| data | <p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p> |
| stat | <p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none">• A <code>Stat</code> ggproto subclass, for example <code>StatCount</code>. |

	• A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following: • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through ... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
clockwise	A single boolean value indicates clockwise or not.
steps	An integer indicating the number of steps to generate the pie chart radian. Increasing this value results in a smoother pie circular.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

<code>show.legend</code>	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<code>inherit.aes</code>	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders() .

new aesthetics

- `angle`: the pie circle angle.
- `angle0`: the initial pie circle angle.
- `radius`: the circle radius.

Aesthetics

`geom_pie()` understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- `x`
- `y`
- `angle`
- `alpha` → NA
- `angle0` → 0
- `colour` → via `theme()`
- `fill` → via `theme()`
- `group` → inferred
- `linetype` → via `theme()`
- `linewidth` → via `theme()`
- `radius` → NULL

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

Examples

```
ggplot(data.frame(x = 1:10, y = 1:10, value = 1:10 / sum(1:10))) +
  geom_pie(aes(x, y, angle = value * 360))
```

geom_rect3d*Add z-aesthetic for geom_tile*

Description

Add z-aesthetic for geom_tile

Usage

```
geom_rect3d(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 10,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_tile3d(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 10,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and 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. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

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

`position`

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

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

`...`

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

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

	The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through <code>...</code>. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
<code>lineend</code>	Line end style (round, butt, square).
<code>linejoin</code>	Line join style (round, mitre, bevel).
<code>linemitre</code>	Line mitre limit (number greater than 1).
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>show.legend</code>	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<code>inherit.aes</code>	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>annotation_borders()</code> .

new aesthetics

- z: the third dimension (in the z direction), use `scale_z_continuous()` to control the ranges.
- theta: Angle between x-axis and z-axis.

Aesthetics

`geom_rect3d()` understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- `x` *or* `width` *or* `xmin` *or* `xmax`
- `y` *or* `height` *or* `ymin` *or* `ymax`
- `z`
- `alpha` → NA
- `colour` → via `theme()`
- `fill` → via `theme()`
- `group` → inferred
- `linetype` → via `theme()`
- `linewidth` → via `theme()`

`geom_tile3d()` understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- `x`
- `y`
- `z`
- `alpha` → NA
- `colour` → via `theme()`

- `fill` → via `theme()`
- `group` → inferred
- `height` → 1
- `linetype` → via `theme()`
- `linewidth` → via `theme()`
- `width` → 1

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(81), nrow = 9)
rownames(small_mat) <- paste0("row", seq_len(nrow(small_mat)))
colnames(small_mat) <- paste0("column", seq_len(ncol(small_mat)))
ggheatmap(small_mat,
  filling = FALSE,
  theme = theme(
    legend.box.spacing = unit(10, "mm"),
    plot.margin = margin(t = 15, unit = "mm")
  )
) +
  geom_tile3d(
    aes(fill = value, z = value, width = 0.8, height = 0.8),
    color = "black"
  ) +
  scale_fill_viridis_c(
    option = "plasma",
    breaks = scales::breaks_pretty(3L)
  ) +
  coord_cartesian(clip = "off")
```

geom_subrect

Subdivide Rectangles

Description

These geoms subdivide rectangles with shared borders into a grid. Both geoms achieve the same result but differ in how the rectangles are parameterized:

- `geom_subrect()`: Defines rectangles using their four corners (`xmin`, `xmax`, `ymin`, `ymax`).
- `geom_subtile()`: Defines rectangles using the center (`x`, `y`) and dimensions (`width`, `height`).

Usage

```
geom_subrect(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  byrow = FALSE,
  nrow = NULL,
  ncol = NULL,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  direction = deprecated()
)

geom_subtile(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  byrow = TRUE,
  nrow = NULL,
  ncol = NULL,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  direction = deprecated()
)
```

Arguments

- | | |
|---------|---|
| mapping | Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. |
| data | <p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return</p> |

	value must be a <code>data.frame</code> , and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).
stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following: • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
byrow	A single boolean value indicates whether we should arrange the divided rectangles in the row-major order.

nrow, ncol	A single positive integer specifying the number of rows or columns in the layout of the subdivided cell. By default, the layout dimensions are determined automatically using logic similar to facet_wrap() .
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders() .
direction	[Deprecated] A string specifying the arrangement direction: • "h"(horizontal): Creates a single row (one-row layout). • "v"(vertical): Creates a single column (one-column layout).

Aesthetics

`geom_subrect()` understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- **x** *or* width *or* **xmin** *or* **xmax**
- **y** *or* height *or* **ymin** *or* **ymax**
- **alpha** → NA
- **colour** → via `theme()`
- **fill** → via `theme()`
- **group** → inferred
- **linetype** → via `theme()`
- **linewidth** → via `theme()`

`geom_subtile()` understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- **x**
- **y**
- **alpha** → NA
- **colour** → via `theme()`
- **fill** → via `theme()`
- **group** → inferred
- height → 1
- **linetype** → via `theme()`
- **linewidth** → via `theme()`

- width → 1

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

Examples

```
# arranges by row
ggplot(data.frame(value = letters[seq_len(5)])) +
  geom_subtile(aes(x = 1, y = 1, fill = value), byrow = TRUE)

# arranges by column
ggplot(data.frame(value = letters[seq_len(9)])) +
  geom_subtile(aes(x = 1, y = 1, fill = value))

# one-row
ggplot(data.frame(value = letters[seq_len(4)])) +
  geom_subtile(aes(x = 1, y = 1, fill = value), nrow = 1)

# one-column
ggplot(data.frame(value = letters[seq_len(4)])) +
  geom_subtile(aes(x = 1, y = 1, fill = value), ncol = 1)
```

ggalign

Add ggplot by Aligning discrete or continuous variable

Description

[Stable]

`ggalign()` is similar to `ggplot` in that it initializes a `ggplot` data and mapping. `ggalign()` allowing you to provide data in various formats, including matrices, data frames, or simple vectors. By default, it will inherit from the layout. If a function, it will apply with the layout matrix. `ggalign()` focuses on integrating plots into a layout by aligning the axes.

Usage

```
ggalign(
  data = waiver(),
  mapping = aes(),
  ...,
  size = NULL,
  active = NULL,
  no_axes = deprecated()
)
```

Arguments

data	The following options can be used: • <code>NULL</code>: No data is set. • <code>waiver()</code>: Inherits the data from the layout matrix. • A function (including purrr-like lambda syntax): Applied to the layout matrix to transform the data before use. To transform the final plot data, please use <code>scheme_data()</code>. • A matrix, data.frame, or atomic vector.
mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...	<dyn-dots> Additional arguments passed to <code>fortify_data_frame()</code> .
size	The relative size of the plot, can be specified as a <code>unit()</code> . Note that for <code>circle_layout()</code> , all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.
active	A <code>active()</code> object that defines the context settings when added to a layout.
no_axes	[Deprecated] Please add <code>theme()</code> directly to the ggplot instead.

ggplot2 specification

ggalign initializes a ggplot object. The underlying data is created using `fortify_data_frame()`. Please refer to it for more details.

When aligning discrete variables, `ggalign()` always applies a default mapping for the axis of the data index in the layout. Specifically:

- `aes(y = .data$.y)` is used for the horizontal `stack_layout()` (including left and right annotations).
- `aes(x = .data$.x)` is used for the vertical `stack_layout()` (including top and bottom annotations) and `circle_layout()`.

The following columns will be added to the data frame to align discrete variables:

- `.panel`: The panel for the aligned axis. Refers to the x-axis for vertical `stack_layout()` (including top and bottom annotations), and the y-axis for horizontal `stack_layout()` (including left and right annotations).
- `.names` (`vec_names()`) and `.index` (`vec_size()/NROW()`): Character names (if available) and the integer index of the original data.
- `.x/.y` and `.discrete_x/.discrete_y`: Integer indices for x/y coordinates, and a factor of the data labels (only applicable when names exist).

It is recommended to use `.x/.y`, or `.discrete_x/.discrete_y` as the x/y mapping.

If the data inherits from `quad_layout()/ggheatmap()`, additional columns will be added:

- `.extra_panel`: Provides the panel information for the column (left or right annotation) or row (top or bottom annotation).
- `.extra_index`: The index information for the column (left or right annotation) or row (top or bottom annotation).

Discrete Axis Alignment

It is important to note that we consider rows as observations, meaning `vec_size(data)/NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +  
  anno_top() +  
  ggalign() +  
  geom_point(aes(y = value))  
  
ggheatmap(matrix(rnorm(81), nrow = 9)) +  
  anno_top(size = 0.5) +  
  align_dendro(k = 3L) +  
  ggalign(data = NULL, size = 0.2) +  
  geom_tile(aes(y = 1L, fill = .panel))
```

ggalignGrob*Generate a plot grob.*

Description

Generate a plot grob.

Usage

```
ggalignGrob(x)
```

Arguments

x An object to be converted into a [grob](#).

Value

A [grob\(\)](#) object.

Examples

```
ggalignGrob(ggplot())
```

`ggalign_attr`*Get Data from the Attribute Attached by ggalign*

Description

`ggalign_attr` retrieves supplementary information stored as attributes during the layout rendering process. These attributes-typically added during data transformation by functions such as `fortify_matrix()` or `fortify_data_frame()`-may contain filtered data, auxiliary metadata, or other context essential for downstream operations.

Factor level information, stored as a separate attribute, can be accessed via `ggalign_lvls`.

Usage

```
ggalign_attr(x, field = NULL, check = TRUE)
```

```
ggalign_lvls(x)
```

Arguments

<code>x</code>	Data used, typically inherited from the layout <code>quad_layout()/ggheatmap()</code> or <code>stack_layout()</code> object.
<code>field</code>	A string specifying the particular data to retrieve from the attached attribute. If <code>NULL</code> , the entire attached attribute list will be returned.
<code>check</code>	A boolean indicating whether to check if the <code>field</code> exists. If <code>TRUE</code> , an error will be raised if the specified <code>field</code> does not exist.

Details

Attributes attached to the data are especially useful when the input data is transformed in ways that limit access to the complete dataset. For example, `fortify_matrix.MAF()` might filter mutation data while adding attributes that retain important context, such as the total number of observations, for detailed or aggregated analyses. Additionally, it stores the levels of `Variant_Classification` for further usage.

Value

- `ggalign_attr`: The specified data from the attached supplementary data or `NULL` if it is unavailable.
- `ggalign_lvls`: The attached supplementary levels or `NULL` if it is unavailable.

ggalign_data_set	<i>Attach supplementary data and levels for ggalign</i>
------------------	---

Description

Attach supplementary data and levels for ggalign

Usage

```
ggalign_data_set(.data, ..., .lvls = NULL)
```

Arguments

.data	Input data for the layout.
...	<dyn-dots> A list of data to be attached.
.lvls	A character vector representing the attached levels.

Note

Used by developers in `fortify_matrix()`, `fortify_data_frame()`, and other related methods.

See Also

`ggalign_attr()/ggalign_lvls()`

ggalign_stat	<i>Get the statistics from the layout</i>
--------------	---

Description

Get the statistics from the layout

Usage

```
ggalign_stat(x, ...)

## S3 method for class '`ggalign::QuadLayout`'
ggalign_stat(x, position, ...)

## S3 method for class '`ggalign::StackLayout`'
ggalign_stat(x, what, ...)
```

Arguments

x	A <code>quad_layout()/ggheatmap()</code> or <code>stack_layout()</code> object.
...	Arguments passed to methods.
position	A string of "top", "left", "bottom", or "right".
what	A single number or string of the plot elements in the stack layout.

Value

The statistics

ggcross	<i>Connect two layout crosswise</i>
---------	-------------------------------------

Description

ggcross resets the layout ordering index of a `stack_cross()`. This allows you to add other `align_*` objects to define a new layout ordering index. Any objects added after `ggcross` will use this updated layout ordering index. This feature is particularly useful for creating tanglegram visualizations. `ggcross()` is an alias of `ggcross()`.

Usage

```
ggcross(mapping = aes(), size = NULL, active = NULL, no_axes = deprecated())
```

Arguments

mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
size	The relative size of the plot, can be specified as a <code>unit()</code> . Note that for <code>circle_layout()</code> , all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.
active	A <code>active()</code> object that defines the context settings when added to a layout.
no_axes	[Deprecated] Please add <code>theme()</code> directly to the <code>ggplot</code> instead.

ggplot2 specification

`ggcross()` initializes a `ggplot` data and mapping.

`ggcross()` always applies a default mapping for the axis of the data index in the layout. This mapping is `aes(y = .data$.y)` for horizontal stack layout (including left and right annotation) and `aes(x = .data$.x)` for vertical stack layout (including top and bottom annotation).

The data in the underlying `ggplot` object will contain following columns:

- `.panel`: The panel for the aligned axis. Refers to the x-axis for vertical `stack_layout()` (including top and bottom annotations), and the y-axis for horizontal `stack_layout()` (including left and right annotations).

- `.names` (`vec_names()`) and `.index` (`vec_size()/NROW()`): Character names (if available) and the integer index of the original data.
- `.hand`: a factor indicates the index groups.
- `.x/.y` and `.discrete_x/.discrete_y`: Integer indices for x/y coordinates, and a factor of the data labels (only applicable when names exist).

It is recommended to use `.x/.y`, or `.discrete_x/.discrete_y` as the x/y mapping.

ggfree

Add ggplot to layout without alignment

Description

[Experimental]

The `ggfree()` function allows you to incorporate a ggplot object into your layout. Unlike `ggalign()`, which aligns every axis value precisely, `ggfree()` focuses on integrating plots into the layout without enforcing strict axis alignment.

Usage

```
ggfree(data = waiver(), ..., size = NULL, active = NULL)
```

```
## Default S3 method:
```

```
ggfree(data = waiver(), mapping = aes(), ..., size = NULL, active = NULL)
```

Arguments

<code>data</code>	The following options can be used: • <code>NULL</code>: No data is set. • <code>waiver()</code>: Inherits the data from the layout matrix. • A function (including purrr-like lambda syntax): Applied to the layout matrix to transform the data before use. To transform the final plot data, please use <code>scheme_data()</code>. • A matrix, data frame, or atomic vector.
<code>...</code>	<dyn-dots> Additional arguments passed to <code>fortify_data_frame()</code> .
<code>size</code>	The relative size of the plot, can be specified as a <code>unit()</code> . Note that for <code>circle_layout()</code> , all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.
<code>active</code>	A <code>active()</code> object that defines the context settings when added to a layout.
<code>mapping</code>	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.

ggplot2 specification

ggalign initializes a ggplot object. The underlying data is created using `fortify_data_frame()`. Please refer to this method for more details.

When used in `quad_layout()/ggheatmap()`, if the data is inherited from the `quad_layout()` and the other direction aligns discrete variables, following columns will be added:

- `.extra_panel`: Provides the panel information for the column (left or right annotation) or row (top or bottom annotation).
- `.extra_index`: The index information for the column (left or right annotation) or row (top or bottom annotation).

Examples

```
ggheatmap(matrix(rnorm(56), nrow = 7)) +
  anno_top() +
  align_dendro() +
  ggfree(mtcars, aes(wt, mpg)) +
  geom_point()
```

ggmark

Add a plot to annotate selected observations

Description

Add a plot to annotate selected observations

Usage

```
ggmark(
  mark,
  data = waiver(),
  mapping = aes(),
  ...,
  group1 = NULL,
  group2 = NULL,
  obs_size = 1,
  size = NULL,
  active = NULL
)
```

Arguments

mark	A <code>mark_draw()</code> object to define how to draw the links. Like <code>mark_line()</code> , <code>mark_tetragon()</code> . Note the names of the pair links will be used to define the panel names so must be unique.
data	The following options can be used:

	• <code>NULL</code>: No data is set. • <code>waiver()</code>: Inherits the data from the layout matrix. • A function (including purrr-like lambda syntax): Applied to the layout matrix to transform the data before use. To transform the final plot data, please use <code>scheme_data()</code>. • A matrix, data.frame, or atomic vector.
mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...	<dyn-dots> Additional arguments passed to <code>fortify_data_frame()</code> .
group1, group2	A single boolean value indicating whether to use the panel group information from the layout as the paired groups. By default, if no specific observations are selected in mark, <code>ggmark()</code> will automatically connect all observations and group them according to the layout's defined groups.
obs_size	A single numeric value that indicates the size of a single observation, ranging from (0, 1].
size	The relative size of the plot, can be specified as a <code>unit()</code> . Note that for <code>circle_layout()</code> , all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.
active	A <code>active()</code> object that defines the context settings when added to a layout.

ggplot2 specification

`ggmark` initializes a `ggplot` object. The underlying data is created using `fortify_data_frame()`. Please refer to it for more details.

In addition, the following columns will be added to the data frame:

- `.panel`: the panel for the aligned axis. It means x-axis for vertical stack layout (including top and bottom annotation), y-axis for horizontal stack layout (including left and right annotation).
- `.names` (`vec_names()`) and `.index` (`vec_size()/NROW()`): a character names (only applicable when names exists) and an integer of index of the original data.
- `.hand`: A factor with levels `c("left", "right")` for horizontal stack layouts, or `c("top", "bottom")` for vertical stack layouts, indicating the position of the linked observations.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 7)
rownames(small_mat) <- paste0("row", seq_len(nrow(small_mat)))
colnames(small_mat) <- paste0("column", seq_len(ncol(small_mat)))

# mark_line
ggheatmap(small_mat) +
  theme(axis.text.x = element_text(hjust = 0, angle = -60)) +
  anno_right() +
  align_kmeans(3L) +
  ggmark(mark_line(I(1:3) ~ NULL)) +
```

```

    geom_boxplot(aes(.names, value)) +
    theme(plot.margin = margin(l = 0.1, t = 0.1, unit = "npc"))

# mark_tetragon
ggheatmap(small_mat) +
  theme(axis.text.x = element_text(hjust = 0, angle = -60)) +
  anno_right() +
  align_kmeans(3L) +
  ggmark(mark_tetragon(I(1:3) ~ NULL)) +
  geom_boxplot(aes(.names, value)) +
  theme(plot.margin = margin(l = 0.1, t = 0.1, unit = "npc"))

```

ggoncplot

Create an OncoPrint

Description

[Stable]

The `ggoncplot()` function generates oncoPrint visualizations that display genetic alterations in a matrix format. This function is especially useful for visualizing complex genomic data, such as mutations, copy number variations, and other genomic alterations in cancer research.

Usage

```

ggoncplot(
  data = NULL,
  mapping = aes(),
  ...,
  map_width = NULL,
  map_height = NULL,
  reorder_row = reorder_column,
  reorder_column = TRUE,
  remove_duplicates = FALSE,
  width = NA,
  height = NA,
  filling = waiver(),
  theme = NULL,
  active = NULL
)

```

Default S3 method:

```

ggoncplot(
  data = NULL,
  mapping = aes(),
  ...,
  map_width = NULL,
  map_height = NULL,
  reorder_row = reorder_column,

```

```

    reorder_column = TRUE,
    remove_duplicates = FALSE,
    width = NA,
    height = NA,
    filling = waiver(),
    theme = NULL,
    active = NULL
  )

```

Arguments

<code>data</code>	A character matrix which encodes the alterations, you can use ";", ":", " ", or " " to separate multiple alterations.
<code>mapping</code>	Default list of aesthetic mappings to use for main plot in the layout. If not specified, must be supplied in each layer added to the main plot.
<code>...</code>	Additional arguments passed to <code>fortify_matrix()</code> .
<code>map_width, map_height</code>	A named numeric value defines the width/height of each alterations.
<code>reorder_row</code>	A boolean value indicating whether to reorder the rows based on the frequency of alterations. You can set this to <code>FALSE</code> , then add <code>align_order(~rowSums(!is.na(.x)), reverse = TRUE)</code> to achieve the same result. You may also need to set <code>strit = FALSE</code> in <code>align_order()</code> if there are already groups.
<code>reorder_column</code>	A boolean value indicating whether to reorder the columns based on the characteristics of the alterations. You can set this to <code>FALSE</code> , then add <code>align_order2(memo_order)</code> to achieve the same result. You may also need to set <code>strit = FALSE</code> in <code>align_order2()</code> if there are already groups.
<code>remove_duplicates</code>	A logical value indicating whether to remove duplicated variants within the same cell.
<code>width, height</code>	The relative width/height of the main plot, can be a <code>unit</code> object.
<code>filling</code>	Same as <code>ggheatmap()</code> , but only "tile" can be used.
<code>theme</code>	A <code>theme()</code> object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, panel.border, and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.
<code>active</code>	A <code>active()</code> object that defines the context settings when added to a layout.

Details

`ggoncplot()` is a wrapper around the `ggheatmap()` function, designed to simplify the creation of OncoPrint-style visualizations. The function automatically processes the input character matrix by splitting the encoded alterations (delimited by ";", ":", " ", or "|") into individual genomic events and unnesting the columns for visualization.

Value

A `HeatmapLayout` object.

Examples

```
# A simple example from `ComplexHeatmap`
mat <- read.table(textConnection(
  "s1,s2,s3
g1,snv;indel,snv,indel
g2,,snv;indel,snv
g3,snv,,indel;snv"
), row.names = 1, header = TRUE, sep = ",", stringsAsFactors = FALSE)

ggoncoplot(mat, map_width = c(snv = 0.5), map_height = c(indel = 0.9)) +
  guides(fill = "none") +
  anno_top(size = 0.5) +
  ggalign() +
  geom_bar(aes(fill = value), data = function(x) {
    subset(x, !is.na(value))
  }) +
  anno_right(size = 0.5) +
  ggalign() +
  geom_bar(aes(fill = value), orientation = "y", data = function(x) {
    subset(x, !is.na(value))
  }) &
  scale_fill_brewer(palette = "Dark2", na.translate = FALSE)
```

ggupset

Create an UpSet plot

Description

[Experimental]

ggupset is a specialized version of [quad_discrete\(\)](#), which simplifies the creation of Upset plot.

Usage

```
ggupset(
  data = NULL,
  mapping = aes(),
  ...,
  direction = "h",
  point = NULL,
  line = NULL,
  rect = NULL,
  width = NA,
  height = NA,
  theme = NULL,
  active = NULL
)
```

Arguments

data	Data used to create the UpSet plot. <code>fortify_matrix()</code> will be used to convert the data to a matrix. Currently, only <code>fortify_matrix.list_upset</code> and <code>fortify_matrix.matrix_upset</code> are suitable for creating an UpSet plot.
mapping	Default list of aesthetic mappings to use for main plot in the layout. If not specified, must be supplied in each layer added to the main plot.
...	Additional arguments passed to <code>fortify_matrix()</code> .
direction	A string indicating the direction of the UpSet plot, "h"(horizontal) or "v"(vertical). In a vertical UpSet plot, the columns of the matrix correspond to the sets, and the rows correspond to the intersections. By default, the horizontal UpSet plot is used, where the rows of the matrix correspond to the sets and the columns correspond to the intersections.
point	A list of parameters passed to <code>geom_point()</code> .
line	A list of parameters passed to <code>geom_line()</code> .
rect	A list of parameters passed to <code>geom_rect()</code> .
width, height	The relative width/height of the main plot, can be a <code>unit</code> object.
theme	A <code>theme()</code> object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, <code>panel.border</code> , and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.
active	A <code>active()</code> object that defines the context settings when added to a layout.

ggplot2 specification

The data input will be converted to a matrix using `fortify_matrix()`, and the data in the underlying main plot will contain the following columns:

- `.panel_x` and `.panel_y`: the column and row panel groups.
- `.x` and `.y`: an integer index of x and y coordinates
- `.discrete_x` and `.discrete_y`: a factor of the data labels (only applicable when `.row_names` and `.column_names` exists).
- `.row_names` and `.column_names`: A character of the row and column names of the original matrix (only applicable when names exist).
- `.row_index` and `.column_index`: the row and column index of the original matrix.
- `value`: the actual matrix value.

Examples

```
set.seed(123)
lt <- list(
  a = sample(letters, 5),
  b = sample(letters, 10),
  c = sample(letters, 15)
)
ggupset(tune(lt)) +
```

```

scale_fill_manual(values = c("#F0F0F0", "white"), guide = "none") +
scale_color_manual(values = c("grey", "black"), guide = "none") +
anno_top() +
ggalign(data = function(d) ggalign_attr(d, "intersection_sizes")) +
ggplot2::geom_bar(aes(y = .data$value), stat = "identity") +
anno_right() +
ggalign(data = function(d) ggalign_attr(d, "set_sizes")) +
ggplot2::geom_bar(aes(x = .data$value),
  stat = "identity",
  orientation = "y"
)

```

ggwrap

Wrap Arbitrary Graphics to ggplot

Description

The `ggwrap()` function allows non-ggplot2 elements to be converted into a compliant representation for use with `align_plots()`. This is useful for adding any graphics that can be converted into a `grob` with the `as_grob()` method.

Usage

```
ggwrap(plot, ..., align = "panel", on_top = FALSE, clip = TRUE, vp = NULL)
```

Arguments

<code>plot</code>	Any graphic that can be converted into a <code>grob</code> using <code>as_grob()</code> .
<code>...</code>	Additional arguments passed to the <code>as_grob()</code> method.
<code>align</code>	A string specifying the area to place the plot: "full" for the full area, "plot" for the full plotting area (including the axis label), or "panel" for only the actual area where data is drawn.
<code>on_top</code>	A single boolean value indicates whether the graphic plot should be put front-most. Note: the graphic plot will always put above the background.
<code>clip</code>	A single boolean value indicating whether the grob should be clipped if they expand outside their designated area.
<code>vp</code>	A <code>viewport</code> object, you can use this to define the plot area.

Value

A `wrapped_plot` object that can be directly placed into `align_plots()`.

Examples

```

library(grid)
ggwrap(rectGrob(gp = gpar(fill = "goldenrod")), align = "full") +
  inset(rectGrob(gp = gpar(fill = "steelblue")), align = "panel") +
  inset(textGrob("Here are some text", gp = gpar(color = "black")),
        align = "panel"
  )
p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp)) +
  ggtitle("Plot 1")
align_plots(p1, ggwrap(
  ~ plot(mtcars$mpg, mtcars$disp),
  mar = c(0, 2, 0, 0), bg = NA
))

```

hclust2

*Generate Tree Structures with Hierarchical Clustering***Description**

Generate Tree Structures with Hierarchical Clustering

Usage

```

hclust2(
  matrix,
  distance = "euclidean",
  method = "complete",
  use_missing = "pairwise.complete.obs"
)

```

Arguments

matrix	A numeric matrix, or data frame.
distance	A string of distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Correlation coefficient can be also used, including "pearson", "spearman" or "kendall". In this way, 1 - cor will be used as the distance. In addition, you can also provide a dist object directly or a function return a dist object. Use NULL, if you don't want to calculate the distance.
method	A string of the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). You can also provide a function which accepts the calculated distance (or the input matrix if distance is NULL) and returns a hclust object. Alternative, you can supply an object which can be coerced to hclust .

`use_missing` An optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Only used when distance is a correlation coefficient string.

Value

A [hclust](#) object.

See Also

- [cor\(\)](#)
- [dist\(\)](#)
- [hclust\(\)](#)

Examples

```
hclust2(dist(USArrests), method = "ward.D")
```

heatmap_layout	Create a heatmap
----------------	------------------

Description

[Stable]

`heatmap_layout` is a specialized version of [quad_discrete\(\)](#), which simplifies the creation of heatmap plots by integrating essential elements for a standard heatmap layout, ensuring that the appropriate data mapping and visualization layers are automatically applied. `ggheatmap` is an alias for `heatmap_layout`.

Usage

```
heatmap_layout(
  data = NULL,
  mapping = aes(),
  ...,
  width = NA,
  height = NA,
  filling = waiver(),
  theme = NULL,
  active = NULL
)
```

Arguments

data	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout. By default, it will try to inherit from parent layout. fortify_matrix() will be used to convert data to a matrix.
mapping	Default list of aesthetic mappings to use for main plot in the layout. If not specified, must be supplied in each layer added to the main plot.
...	Additional arguments passed to fortify_matrix() .
width, height	The relative width/height of the main plot, can be a unit object.
filling	A single string of "raster" or "tile" to indicate the filling style. By default, waiver() is used, which means that if the input matrix has more than 20,000 cells ($nrow * ncol > 20000$), geom_raster() will be used for performance efficiency; for smaller matrices, geom_tile() will be used. To customize the filling style, set this to NULL. For backward compatibility, a single boolean value is acceptable: TRUE means waiver(), and FALSE means NULL. By default, the classic heatmap color scheme scale_fill_gradient2(low = "blue", high = "red") is utilized for continuous values. You can use the options "galign.heatmap_continuous_fill" or "galign.heatmap_discrete_fill" to modify the default heatmap body filling color scale. See scale_fill_continuous() or scale_fill_discrete() for details on option settings.
theme	A theme() object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, panel.border, and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.
active	A active() object that defines the context settings when added to a layout.

Value

A HeatmapLayout object.

ggplot2 specification

The data input will be converted to a matrix using [fortify_matrix\(\)](#), and the data in the underlying main plot will contain the following columns:

- .panel_x and .panel_y: the column and row panel groups.
- .x and .y: an integer index of x and y coordinates
- .discrete_x and .discrete_y: a factor of the data labels (only applicable when .row_names and .column_names exists).
- .row_names and .column_names: A character of the row and column names of the original matrix (only applicable when names exist).
- .row_index and .column_index: the row and column index of the original matrix.
- value: the actual matrix value.

Examples

```
ggheatmap(1:10)
ggheatmap(letters)
ggheatmap(matrix(rnorm(81), nrow = 9L))
```

inset

*Create a ggplot inset***Description**

Create a ggplot inset

Usage

```
inset(plot, ..., align = "panel", on_top = TRUE, clip = TRUE, vp = NULL)
```

Arguments

plot	Any graphic that can be converted into a grob using as_grob() .
...	Additional arguments passed to the as_grob() method.
align	A string specifying the area to place the plot: "full" for the full area, "plot" for the full plotting area (including the axis label), or "panel" for only the actual area where data is drawn.
on_top	A single boolean value indicates whether the graphic plot should be put front-most. Note: the graphic plot will always put above the background.
clip	A single boolean value indicating whether the grob should be clipped if they expand outside their designated area.
vp	A viewport object, you can use this to define the plot area.

Value

An inset object, which can be added to ggplot.

Examples

```
library(grid)
p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
  geom_boxplot(aes(gear, disp, group = gear))
p1 + inset(p2, vp = viewport(0.6, 0.6,
  just = c(0, 0), width = 0.4, height = 0.4
))
```

is_layout	<i>Reports whether x is layout object</i>
-----------	---

Description

Reports whether x is layout object

Usage

```
is_layout(x)

is_quad_layout(x)

is_stack_layout(x)

is_stack_cross(x)

is_circle_layout(x)

is_heatmap_layout(x)

is_ggheatmap(x)
```

Arguments

x	An object to test.
---	--------------------

Value

A single boolean value.

Examples

```
is_layout(ggheatmap(1:10))

# for quad_layout()
is_quad_layout(quad_alignb(1:10))
is_quad_layout(quad_alighh(1:10))
is_quad_layout(quad_alignv(1:10))
is_quad_layout(quad_free(mtcars))

# for stack_layout()
is_stack_layout(stack_discrete("h", 1:10))
is_stack_layout(stack_continuous("h", 1:10))

# for heatmap_layout()
is_heatmap_layout(ggheatmap(1:10))
is_ggheatmap(ggheatmap(1:10))
```

layer_order	Change the layer adding order
-------------	-------------------------------

Description

This function allows you to change the order in which layers are added to a ggplot.

Usage

```
layer_order(layer, order = 0)
```

Arguments

layer	A layer geometry object to be added.
order	An integer indicating the position at which the layer should be added. If ≤ 0 , the layer will be added at the beginning. If greater than the number of plot layers, it will be added at the end.

Value

A layer_order object.

Examples

```
ggplot(faithfuld, aes(waiting, eruptions)) +  
  geom_raster(aes(fill = density)) +  
  geom_point(color = "red", size = 1)  
ggplot(faithfuld, aes(waiting, eruptions)) +  
  geom_raster(aes(fill = density)) +  
  layer_order(geom_point(color = "red", size = 1))
```

layout-operator	Layout operator
-----------------	-----------------

Description

[Experimental]

- **+**: Adds elements to the active plot in the active layout.
- **&**: Applies elements to all plots in the layout.
- **-**: Adds elements to multiple plots in the layout.

Arguments

e1	A quad_layout() / ggheatmap() or stack_layout() object.
e2	An object to be added to the plot.

Details

The + operator is straightforward and should be used as needed.

In order to reduce code repetition ggalign provides two operators for adding ggplot elements (geoms, themes, facets, etc.) to multiple/all plots in `quad_layout()/ggheatmap()` or `stack_layout()` object: - and &.

Value

A modified Layout object.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 7)
ggheatmap(small_mat) +
  anno_top() +
  ggalign() +
  geom_point(aes(y = value))

# `` operator apply it to all plots
ggheatmap(small_mat) +
  anno_top() +
  align_dendro() &
  theme(panel.border = element_rect(
    colour = "red", fill = NA, linewidth = unit(2, "mm")
  ))

# If the active layout is the annotation stack, the `` operator will only
# add the elements to all plots in the active annotation stack:
ggheatmap(small_mat) +
  anno_left(size = 0.2) +
  align_dendro(aes(color = branch), k = 3L) +
  align_dendro(aes(color = branch), k = 3L) -
  # Modify the the color scales of all plots in the left annotation
  scale_color_brewer(palette = "Dark2")

# If the active layout is the `stack_layout()` itself, ``
# applies the elements to all plots in the layout except the nested
# `ggheatmap()`/`quad_layout()`.
stack_alignv(small_mat) +
  align_dendro() +
  ggtitle("I'm from the parent stack") +
  ggheatmap() +
  # remove any active context
  stack_active() +
  align_dendro() +
  ggtitle("I'm from the parent stack") -
  # Modify the the color scales of all plots in the stack layout except the
  # heatmap layout
  scale_color_brewer(palette = "Dark2") -
  # set the background of all plots in the stack layout except the heatmap
```

```
# layout
theme(plot.background = element_rect(fill = "red"))
```

layout_design

Define the grid to compose plots in

Description

To control how different plots are laid out, you need to add a layout design specification. If you are nesting grids, the layout is scoped to the current nesting level.

Usage

```
layout_design(
  ncol = NA_real_,
  nrow = NA_real_,
  byrow = NA,
  widths = NULL,
  heights = NULL,
  area = waiver(),
  guides = NA_character_
)
```

Arguments

ncol, nrow	The number of columns and rows in the grid. Defaults to NULL. If both are NULL, the layout dimensions are determined automatically using the same logic as facet_wrap() .
byrow	A logical value indicating whether plots should be filled in row-major order (TRUE) or column-major order (FALSE). Defaults to TRUE.
widths, heights	The relative widths and heights of each column and row in the grid. These values are recycled to match the grid dimensions. The special value NA is treated as a unit of 1null, unless a fixed-aspect plot is included — in that case, the affected dimension will expand or contract to maintain the aspect ratio of the plot. Defaults to NA.
area	A specification of the area layout. Can be defined either as a character string or as a combination of calls to area() . Defaults to NULL.
guides	A string with one or more of "t", "l", "b", "r", and "i" indicating which side of guide legends should be collected. Defaults to waiver() , which inherits from the parent layout. If there is no parent layout, or if NULL is provided, no guides will be collected.

Value

A layout_design object.

Examples

```
p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
  geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
  geom_bar(aes(gear)) +
  facet_wrap(~cyl)
align_plots(p1, p2, p3) +
  layout_design(nrow = 1L)
align_plots(p1, p2, p3) +
  layout_design(ncol = 1L)
```

 layout_tags

Control Plot Tagging in Layouts

Description

These arguments control how tags (labels) are assigned to plots within a layout, including nested layouts. Tags can be inherited from a parent layout, suppressed entirely, or generated automatically in various sequences. Formatting can be customized with separators, prefixes, and suffixes.

Usage

```
layout_tags(
  tags = NA_character_,
  sep = waiver(),
  prefix = waiver(),
  suffix = waiver()
)
```

Arguments

tags Tag templates for plots in the layout. If `waiver()` (default), tags are inherited from the parent layout. If there is no parent layout, no tags are applied. If `NULL`, tags are suppressed for this layout. In a nested layout, the parent layout's tag is applied to the the entire layout as a single unit. If not `NULL`, must be one of:

- A character vector specifying explicit tags for each plot, or
- A single character indicating an auto-generated sequence:
 - 'a': lowercase letters
 - 'A': uppercase letters
 - '1': numbers
 - 'i': lowercase Roman numerals
 - 'I': uppercase Roman numerals

	When a parent layout exists, each plot's tag is prefixed with the parent tag and separated by sep.
sep	Separator between the parent tag (without its own prefix and suffix) and the current tag.
prefix	String prepended to the tag.
suffix	String appended to the tag.

Details

The appearance of tags is controlled by the `plot.tag`, `plot.tag.position`, and `plot.tag.location` theme elements. Tag styling is first retrieved from the plot's theme; if not found there, the layout's theme is used.

Examples

```
p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
  geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
  geom_bar(aes(gear)) +
  facet_wrap(~cyl)

# Add tags to plots, by default the plot in nested layout will get own tag
align_plots(p1, align_plots(p2, p3), ncol = 1) + layout_tags("A")

# Treat a nested layout as a single plot by disabling its internal tags
align_plots(p1, align_plots(p2, p3) + layout_tags(NULL), ncol = 1) +
  layout_tags("A")

# Apply multilevel tagging – outer layout uses letters, inner layout uses
# numbers
align_plots(
  p1,
  align_plots(p2, p3) + layout_tags(1),
  ncol = 1
) +
  layout_tags("A")

# Use a custom tag sequence, possibly mixed with standard sequences
align_plots(
  p1,
  align_plots(p2, p3) + layout_tags(1),
  ncol = 1
) +
  layout_tags(c("&", "%"))
```

layout_theme

Modify theme of the layout

Description

Themes are a powerful way to customize the non-data components of your plots: i.e. titles, labels, fonts, background, gridlines, and legends. Themes can be used to give plots a consistent customized look. Modify a single plot's theme using `theme()`; see [theme_update\(\)](#) if you want modify the active theme, to affect all subsequent plots. Use the themes available in [complete themes](#) if you would like to use a complete theme such as `theme_bw()`, `theme_minimal()`, and more. Theme elements are documented together according to inheritance, read more about theme inheritance below.

Usage

```
layout_theme(  
  ...,  
  line,  
  rect,  
  text,  
  title,  
  point,  
  polygon,  
  geom,  
  spacing,  
  margins,  
  aspect.ratio,  
  axis.title,  
  axis.title.x,  
  axis.title.x.top,  
  axis.title.x.bottom,  
  axis.title.y,  
  axis.title.y.left,  
  axis.title.y.right,  
  axis.text,  
  axis.text.x,  
  axis.text.x.top,  
  axis.text.x.bottom,  
  axis.text.y,  
  axis.text.y.left,  
  axis.text.y.right,  
  axis.text.theta,  
  axis.text.r,  
  axis.ticks,  
  axis.ticks.x,  
  axis.ticks.x.top,  
  axis.ticks.x.bottom,
```

```
axis.ticks.y,  
axis.ticks.y.left,  
axis.ticks.y.right,  
axis.ticks.theta,  
axis.ticks.r,  
axis.minor.ticks.x.top,  
axis.minor.ticks.x.bottom,  
axis.minor.ticks.y.left,  
axis.minor.ticks.y.right,  
axis.minor.ticks.theta,  
axis.minor.ticks.r,  
axis.ticks.length,  
axis.ticks.length.x,  
axis.ticks.length.x.top,  
axis.ticks.length.x.bottom,  
axis.ticks.length.y,  
axis.ticks.length.y.left,  
axis.ticks.length.y.right,  
axis.ticks.length.theta,  
axis.ticks.length.r,  
axis.minor.ticks.length,  
axis.minor.ticks.length.x,  
axis.minor.ticks.length.x.top,  
axis.minor.ticks.length.x.bottom,  
axis.minor.ticks.length.y,  
axis.minor.ticks.length.y.left,  
axis.minor.ticks.length.y.right,  
axis.minor.ticks.length.theta,  
axis.minor.ticks.length.r,  
axis.line,  
axis.line.x,  
axis.line.x.top,  
axis.line.x.bottom,  
axis.line.y,  
axis.line.y.left,  
axis.line.y.right,  
axis.line.theta,  
axis.line.r,  
legend.background,  
legend.margin,  
legend.spacing,  
legend.spacing.x,  
legend.spacing.y,  
legend.key,  
legend.key.size,  
legend.key.height,  
legend.key.width,  
legend.key.spacing,
```

```
legend.key.spacing.x,  
legend.key.spacing.y,  
legend.key.justification,  
legend.frame,  
legend.ticks,  
legend.ticks.length,  
legend.axis.line,  
legend.text,  
legend.text.position,  
legend.title,  
legend.title.position,  
legend.position,  
legend.position.inside,  
legend.direction,  
legend.byrow,  
legend.justification,  
legend.justification.top,  
legend.justification.bottom,  
legend.justification.left,  
legend.justification.right,  
legend.justification.inside,  
legend.location,  
legend.box,  
legend.box.just,  
legend.box.margin,  
legend.box.background,  
legend.box.spacing,  
panel.background,  
panel.border,  
panel.spacing,  
panel.spacing.x,  
panel.spacing.y,  
panel.grid,  
panel.grid.major,  
panel.grid.minor,  
panel.grid.major.x,  
panel.grid.major.y,  
panel.grid.minor.x,  
panel.grid.minor.y,  
panel.ontop,  
panel.widths,  
panel.heights,  
plot.background,  
plot.title,  
plot.title.position,  
plot.subtitle,  
plot.caption,  
plot.caption.position,
```

```

plot.tag,
plot.tag.position,
plot.tag.location,
plot.margin,
strip.background,
strip.background.x,
strip.background.y,
strip.clip,
strip.placement,
strip.text,
strip.text.x,
strip.text.x.bottom,
strip.text.x.top,
strip.text.y,
strip.text.y.left,
strip.text.y.right,
strip.switch.pad.grid,
strip.switch.pad.wrap,
complete = FALSE,
validate = TRUE
)

```

Arguments

...	A theme() object or additional element specifications not part of base ggplot2. In general, these should also be defined in the <code>element</code> tree argument. Splicing a list is also supported.
line	all line elements (element_line())
rect	all rectangular elements (element_rect())
text	all text elements (element_text())
title	all title elements: plot, axes, legends (element_text() ; inherits from text)
point	all point elements (element_point())
polygon	all polygon elements (element_polygon())
geom	defaults for geoms (element_geom())
spacing	all spacings (unit())
margins	all margins (margin())
aspect.ratio	aspect ratio of the panel
axis.title, axis.title.x.bottom, axis.title.y.left, axis.title.y.right	axis.title.x, axis.title.y, axis.title.x.top, axis.title.x.bottom, axis.title.y.left, axis.title.y.right labels of axes (element_text()). Specify all axes' labels (axis.title), labels by plane (using axis.title.x or axis.title.y), or individually for each axis (using axis.title.x.bottom, axis.title.x.top, axis.title.y.left, axis.title.y.right). axis.title.*.* inherits from axis.title.* which inherits from axis.title, which in turn inherits from text

`axis.text,` `axis.text.x,` `axis.text.y,` `axis.text.x.top,`
`axis.text.x.bottom,` `axis.text.y.left,` `axis.text.y.right,`
`axis.text.theta,` `axis.text.r`
tick labels along axes (`element_text()`). Specify all axis tick labels (`axis.text`), tick labels by plane (using `axis.text.x` or `axis.text.y`), or individually for each axis (using `axis.text.x.bottom`, `axis.text.x.top`, `axis.text.y.left`, `axis.text.y.right`). `axis.text.*` inherits from `axis.text`, which in turn inherits from `text`

`axis.ticks,` `axis.ticks.x,` `axis.ticks.x.top,` `axis.ticks.x.bottom,`
`axis.ticks.y,` `axis.ticks.y.left,` `axis.ticks.y.right,`
`axis.ticks.theta,` `axis.ticks.r`
tick marks along axes (`element_line()`). Specify all tick marks (`axis.ticks`), ticks by plane (using `axis.ticks.x` or `axis.ticks.y`), or individually for each axis (using `axis.ticks.x.bottom`, `axis.ticks.x.top`, `axis.ticks.y.left`, `axis.ticks.y.right`). `axis.ticks.*` inherits from `axis.ticks`, which in turn inherits from `line`

`axis.minor.ticks.x.top,` `axis.minor.ticks.x.bottom,`
`axis.minor.ticks.y.left,` `axis.minor.ticks.y.right,`
`axis.minor.ticks.theta,` `axis.minor.ticks.r`
minor tick marks along axes (`element_line()`). `axis.minor.ticks.*` inherit from the corresponding major ticks `axis.ticks.*`.

`axis.ticks.length,` `axis.ticks.length.x,` `axis.ticks.length.x.top,`
`axis.ticks.length.x.bottom,` `axis.ticks.length.y,`
`axis.ticks.length.y.left,` `axis.ticks.length.y.right,`
`axis.ticks.length.theta,` `axis.ticks.length.r`
length of tick marks (unit). `axis.ticks.length` inherits from `spacing`.

`axis.minor.ticks.length,` `axis.minor.ticks.length.x,`
`axis.minor.ticks.length.x.top,` `axis.minor.ticks.length.x.bottom,`
`axis.minor.ticks.length.y,` `axis.minor.ticks.length.y.left,`
`axis.minor.ticks.length.y.right,` `axis.minor.ticks.length.theta,`
`axis.minor.ticks.length.r`
length of minor tick marks (unit), or relative to `axis.ticks.length` when provided with `rel()`.

`axis.line,` `axis.line.x,` `axis.line.x.top,` `axis.line.x.bottom,`
`axis.line.y,` `axis.line.y.left,` `axis.line.y.right,` `axis.line.theta,`
`axis.line.r`
lines along axes (`element_line()`). Specify lines along all axes (`axis.line`), lines for each plane (using `axis.line.x` or `axis.line.y`), or individually for each axis (using `axis.line.x.bottom`, `axis.line.x.top`, `axis.line.y.left`, `axis.line.y.right`). `axis.line.*` inherits from `axis.line`, which in turn inherits from `line`

`legend.background`
background of legend (`element_rect()`; inherits from `rect`)

`legend.margin` the margin around each legend (`margin()`; inherits from `margins`.

`legend.spacing,` `legend.spacing.x,` `legend.spacing.y`
the spacing between legends (unit). `legend.spacing.x` & `legend.spacing.y` inherit from `legend.spacing` or can be specified separately. `legend.spacing` inherits from `spacing`.

`legend.key` background underneath legend keys (`element_rect()`; inherits from `rect`)
`legend.key.size`, `legend.key.height`, `legend.key.width`
size of legend keys (unit); key background height & width inherit from `legend.key.size` or can be specified separately. In turn `legend.key.size` inherits from `spacing`.
`legend.key.spacing`, `legend.key.spacing.x`, `legend.key.spacing.y`
spacing between legend keys given as a unit. Spacing in the horizontal (x) and vertical (y) direction inherit from `legend.key.spacing` or can be specified separately. `legend.key.spacing` inherits from `spacing`.
`legend.key.justification`
Justification for positioning legend keys when more space is available than needed for display. The default, `NULL`, stretches keys into the available space. Can be a location like "center" or "top", or a two-element numeric vector.
`legend.frame` frame drawn around the bar (`element_rect()`).
`legend.ticks` tick marks shown along bars or axes (`element_line()`)
`legend.ticks.length`
length of tick marks in legend (`unit()`); inherits from `legend.key.size`.
`legend.axis.line`
lines along axes in legends (`element_line()`)
`legend.text` legend item labels (`element_text()`; inherits from `text`)
`legend.text.position`
placement of legend text relative to legend keys or bars ("top", "right", "bottom" or "left"). The legend text placement might be incompatible with the legend's direction for some guides.
`legend.title` title of legend (`element_text()`; inherits from `title`)
`legend.title.position`
placement of legend title relative to the main legend ("top", "right", "bottom" or "left").
`legend.position`
the default position of legends ("none", "left", "right", "bottom", "top", "inside")
`legend.position.inside`
A numeric vector of length two setting the placement of legends that have the "inside" position.
`legend.direction`
layout of items in legends ("horizontal" or "vertical")
`legend.byrow` whether the legend-matrix is filled by columns (`FALSE`, the default) or by rows (`TRUE`).
`legend.justification`
anchor point for positioning legend inside plot ("center" or two-element numeric vector) or the justification according to the plot area when positioned outside the plot
`legend.justification.top`, `legend.justification.bottom`,
`legend.justification.left`, `legend.justification.right`,
`legend.justification.inside`
Same as `legend.justification` but specified per `legend.position` option.

<code>legend.location</code>	Relative placement of legends outside the plot as a string. Can be "panel" (default) to align legends to the panels or "plot" to align legends to the plot as a whole.
<code>legend.box</code>	arrangement of multiple legends ("horizontal" or "vertical")
<code>legend.box.just</code>	justification of each legend within the overall bounding box, when there are multiple legends ("top", "bottom", "left", "right", "center" or "centre")
<code>legend.box.margin</code>	margins around the full legend area, as specified using <code>margin()</code> ; inherits from <code>margins</code> .
<code>legend.box.background</code>	background of legend area (<code>element_rect()</code> ; inherits from <code>rect</code>)
<code>legend.box.spacing</code>	The spacing between the plotting area and the legend box (unit); inherits from <code>spacing</code> .
<code>panel.background</code>	background of plotting area, drawn underneath plot (<code>element_rect()</code> ; inherits from <code>rect</code>)
<code>panel.border</code>	border around plotting area, drawn on top of plot so that it covers tick marks and grid lines. This should be used with <code>fill = NA</code> (<code>element_rect()</code> ; inherits from <code>rect</code>)
<code>panel.spacing</code> , <code>panel.spacing.x</code> , <code>panel.spacing.y</code>	spacing between facet panels (unit). <code>panel.spacing.x</code> & <code>panel.spacing.y</code> inherit from <code>panel.spacing</code> or can be specified separately. <code>panel.spacing</code> inherits from <code>spacing</code> .
<code>panel.grid</code> , <code>panel.grid.major</code> , <code>panel.grid.minor</code> , <code>panel.grid.major.x</code> , <code>panel.grid.major.y</code> , <code>panel.grid.minor.x</code> , <code>panel.grid.minor.y</code>	grid lines (<code>element_line()</code>). Specify major grid lines, or minor grid lines separately (using <code>panel.grid.major</code> or <code>panel.grid.minor</code>) or individually for each axis (using <code>panel.grid.major.x</code> , <code>panel.grid.minor.x</code> , <code>panel.grid.major.y</code> , <code>panel.grid.minor.y</code>). Y axis grid lines are horizontal and x axis grid lines are vertical. <code>panel.grid.*</code> inherits from <code>panel.grid.*</code> which inherits from <code>panel.grid</code> , which in turn inherits from <code>line</code>
<code>panel.ontop</code>	option to place the panel (background, gridlines) over the data layers (logical). Usually used with a transparent or blank <code>panel.background</code> .
<code>panel.widths</code> , <code>panel.heights</code>	Sizes for panels (units). Can be a single unit to set the total size for the panel area, or a unit vector to set the size of individual panels.
<code>plot.background</code>	background of the entire plot (<code>element_rect()</code> ; inherits from <code>rect</code>)
<code>plot.title</code>	plot title (text appearance) (<code>element_text()</code> ; inherits from <code>title</code>) left-aligned by default
<code>plot.title.position</code> , <code>plot.caption.position</code>	Alignment of the plot title/subtitle and caption. The setting for <code>plot.title.position</code> applies to both the title and the subtitle. A value of "panel" (the default) means

	that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).
plot.subtitle	plot subtitle (text appearance) (element_text() ; inherits from title) left-aligned by default
plot.caption	caption below the plot (text appearance) (element_text() ; inherits from title) right-aligned by default
plot.tag	upper-left label to identify a plot (text appearance) (element_text() ; inherits from title) left-aligned by default
plot.tag.position	The position of the tag as a string ("topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright") or a coordinate. If a coordinate, can be a numeric vector of length 2 to set the x,y-coordinate relative to the whole plot. The coordinate option is unavailable for <code>plot.tag.location = "margin"</code> .
plot.tag.location	The placement of the tag as a string, one of "panel", "plot" or "margin". Respectively, these will place the tag inside the panel space, anywhere in the plot as a whole, or in the margin around the panel space.
plot.margin	margin around entire plot (unit with the sizes of the top, right, bottom, and left margins); inherits from margin.
strip.background, strip.background.x, strip.background.y	background of facet labels (element_rect() ; inherits from rect). Horizontal facet background (<code>strip.background.x</code>) & vertical facet background (<code>strip.background.y</code>) inherit from <code>strip.background</code> or can be specified separately
strip.clip	should strip background edges and strip labels be clipped to the extend of the strip background? Options are "on" to clip, "off" to disable clipping or "inherit" (default) to take the clipping setting from the parent viewport.
strip.placement	placement of strip with respect to axes, either "inside" or "outside". Only important when axes and strips are on the same side of the plot.
strip.text, strip.text.x, strip.text.y, strip.text.x.top, strip.text.x.bottom, strip.text.y.left, strip.text.y.right	facet labels (element_text() ; inherits from text). Horizontal facet labels (<code>strip.text.x</code>) & vertical facet labels (<code>strip.text.y</code>) inherit from <code>strip.text</code> or can be specified separately. Facet strips have dedicated position-dependent theme elements (<code>strip.text.x.top</code> , <code>strip.text.x.bottom</code> , <code>strip.text.y.left</code> , <code>strip.text.y.right</code>) that inherit from <code>strip.text.x</code> and <code>strip.text.y</code> , respectively. As a consequence, some theme stylings need to be applied to the position-dependent elements rather than to the parent elements
strip.switch.pad.grid, strip.switch.pad.wrap	space between strips and axes when strips are switched (unit); inherits from spacing.
complete	set this to TRUE if this is a complete theme, such as the one returned by theme_grey() . Complete themes behave differently when added to a ggplot object. Also, when setting <code>complete = TRUE</code> all elements will be set to inherit from blank elements.
validate	TRUE to run <code>check_element()</code> , FALSE to bypass checks.

Details

A `theme()` object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, `panel.border`, and background. By default, the theme will inherit from the parent layout.

- `guides`, `panel.border`, and background will always be used even for the nested `alignpatches` object.
- title, subtitle, caption, and margins will be added for the top-level `alignpatches` object only.

Theme inheritance

Theme elements inherit properties from other theme elements hierarchically. For example, `axis.title.x.bottom` inherits from `axis.title.x` which inherits from `axis.title`, which in turn inherits from `text`. All text elements inherit directly or indirectly from `text`; all lines inherit from `line`, and all rectangular objects inherit from `rect`. This means that you can modify the appearance of multiple elements by setting a single high-level component.

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

See Also

`add_gg()` and `%+replace%`, `element_blank()`, `element_line()`, `element_rect()`, and `element_text()` for details of the specific theme elements.

The [modifying theme components](#) and [theme elements sections](#) of the online ggplot2 book.

Examples

```
p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
  geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
  geom_bar(aes(gear)) +
  facet_wrap(~cyl)
align_plots(
  p1 + theme(plot.background = element_blank()),
  p2 + theme(plot.background = element_blank()),
  p3 + theme(plot.background = element_blank())
) +
  layout_theme(plot.background = element_rect(fill = "red"))
```

layout_title

Annotate the whole layout

Description

Annotate the whole layout

Usage

```
layout_title(title = waiver(), subtitle = waiver(), caption = waiver())
```

Arguments

<code>title</code>	The text for the title.
<code>subtitle</code>	The text for the subtitle for the plot which will be displayed below the title.
<code>caption</code>	The text for the caption which will be displayed in the bottom-right of the plot by default.

Value

A `layout_title` object.

Examples

```
p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
  geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
  geom_bar(aes(gear)) +
  facet_wrap(~cyl)
align_plots(p1, p2, p3) +
  layout_title(title = "I'm title")
```

link_draw

Define the links to connect a pair of observations

Description

This function allows users to define links between a pair of observations, facilitating the visualization of connections between related data points.

Usage

```
link_draw(.draw, ...)
```

Arguments

<code>.draw</code>	A function used to draw the links. The function must return a <code>grob()</code> object. If the function does not return a valid grob, no drawing will occur. The input data for the function must contain two arguments: a data frame for the left hand coordinates and a data frame for the right hand observation coordinates.
--------------------	--

... [<dyn-dots>](#) A list of formulas, where each side of the formula should be an integer or character index of the original data, or a `range_link()` object defining the linked observations. Use `NULL` to indicate no link on that side. You can also combine these by wrapping them into a single `list()`. If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with `I()` to use the ordering from the layout. You can also use `waiver()` to inherit values from the other group.

See Also

- [link_line\(\)](#)
- [.link_draw\(\)](#)

link_line

Link the paired observations with a line

Description

Link the paired observations with a line

Usage

```
link_line(..., .element = NULL)
```

Arguments

... [<dyn-dots>](#) A list of formulas, where each side of the formula should be an integer or character index of the original data, or a `range_link()` object defining the linked observations. Use `NULL` to indicate no link on that side. You can also combine these by wrapping them into a single `list()`. If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with `I()` to use the ordering from the layout. You can also use `waiver()` to inherit values from the other group.

.element A `element_line()` object. Vectorized fields will be recycled to match the total number of groups, or you can wrap the element with `I()` to recycle to match the drawing groups. The drawing groups typically correspond to the product of the number of observations from both sides, as each pair of observations will be linked with a single line.

link_tetragon	<i>Link the paired observations with a quadrilateral</i>
---------------	--

Description

Link the paired observations with a quadrilateral

Usage

```
link_tetragon(..., .element = NULL)
```

Arguments

...	<dyn-dots> A list of formulas, where each side of the formula should be an integer or character index of the original data, or a <code>range_link()</code> object defining the linked observations. Use <code>NULL</code> to indicate no link on that side. You can also combine these by wrapping them into a single <code>list()</code> . If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with <code>I()</code> to use the ordering from the layout. You can also use <code>waiver()</code> to inherit values from the other group.
.element	A <code>element_polygon()</code> object. Vectorized fields will be recycled to match the total number of groups, or you can wrap the element with <code>I()</code> to recycle to match the drawing groups. The drawing groups are usually the same as the defined groups, but they will differ when the defined group of observations is separated and cannot be linked with a single quadrilateral. In such cases, the number of drawing groups will be larger than the number of defined groups.

magickGrob	<i>Rasterize a grob object with magick</i>
------------	--

Description

Rasterize a grob object with magick

Usage

```
magickGrob(grob, ...)

## S3 method for class 'grob'
magickGrob(
  grob,
  magick = NULL,
  ...,
  res = NULL,
  interpolate = FALSE,
```

```

    name = NULL,
    vp = NULL
)

## S3 method for class 'magickGrob'
magickGrob(
  grob,
  magick = waiver(),
  ...,
  res = waiver(),
  interpolate = waiver(),
  name = waiver(),
  vp = waiver()
)
```

Arguments

grob	A grob() . Use as_grob() to convert any objects into a grob.
...	These dots are for future extensions and must be empty.
magick	A function (purrr-style formula is accepted) that takes an image_read() object as input and returns an object compatible with as.raster() . You can use any of the image_*() functions from the magick package to process the raster image.
res	An integer sets the desired resolution in pixels.
interpolate	A logical value indicating whether to linearly interpolate the image (the alternative is to use nearest-neighbour interpolation, which gives a more blocky result).
name	A character identifier.
vp	A Grid viewport object (or NULL).

Value

A magickGrob object.

mark_draw

Define the links to connect the marked observations

Description

This function allows users to define links between marked observations and plot panel (e.g., for creating visual connections for related data), which could help explain the observations.

Usage

```
mark_draw(.draw, ...)
```

Arguments

- `.draw` A function used to draw the links. The function must return a `grob()` object. If the function does not return a valid grob, nothing will be drawn. The input data for the function must contain two arguments: a data frame for the panel side coordinates and a data frame for the marked observation coordinates.
- `...` **<dyn-dots>** A list of formulas, where each side of the formula should be an integer or character index of the original data, or a `range_link()` object defining the linked observations. Use `NULL` to indicate no link on that side. You can also combine these by wrapping them into a single `list()`. If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with `I()` to use the ordering from the layout. You can also use `waiver()` to inherit values from the other group.

See Also

- `mark_line()`
- `mark_tetragon()`
- `mark_triangle()`
- `.mark_draw()`

mark_line

Link the observations and the panel with a line

Description

Link the observations and the panel with a line

Usage

```
mark_line(..., .element = NULL)
```

Arguments

- `...` **<dyn-dots>** A list of formulas, where each side of the formula should be an integer or character index of the original data, or a `range_link()` object defining the linked observations. Use `NULL` to indicate no link on that side. You can also combine these by wrapping them into a single `list()`. If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with `I()` to use the ordering from the layout. You can also use `waiver()` to inherit values from the other group.
- `.element` A `element_line()` object. Vectorized fields will be recycled to match the total number of groups, or you can wrap the element with `I()` to recycle to match the drawing groups. The drawing groups typically correspond to the number of observations, as each observation will be linked with the plot panel.

mark_tetragon	<i>Link the observations and the panel with a quadrilateral</i>
---------------	---

Description

Link the observations and the panel with a quadrilateral

Usage

```
mark_tetragon(..., .element = NULL)
```

Arguments

...	<dyn-dots> A list of formulas, where each side of the formula should be an integer or character index of the original data, or a <code>range_link()</code> object defining the linked observations. Use <code>NULL</code> to indicate no link on that side. You can also combine these by wrapping them into a single <code>list()</code> . If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with <code>I()</code> to use the ordering from the layout. You can also use <code>waiver()</code> to inherit values from the other group.
.element	A <code>element_polygon()</code> object. Vectorized fields will be recycled to match the total number of groups, or you can wrap the element with <code>I()</code> to recycle to match the drawing groups. The drawing groups are usually the same as the defined groups, but they will differ when the defined group of observations is separated and cannot be linked with a single quadrilateral. In such cases, the number of drawing groups will be larger than the number of defined groups.

mark_triangle	<i>Link the observations and the panel with a triangle</i>
---------------	--

Description

Link the observations and the panel with a triangle

Usage

```
mark_triangle(..., orientation = "plot", .element = NULL)
```

Arguments

...	<dyn-dots> A list of formulas, where each side of the formula should be an integer or character index of the original data, or a <code>range_link()</code> object defining the linked observations. Use <code>NULL</code> to indicate no link on that side. You can also combine these by wrapping them into a single <code>list()</code> . If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with <code>I()</code> to use the ordering from the layout. You can also use <code>waiver()</code> to inherit values from the other group.
-----	--

orientation	A single string, either "plot" or "observation", indicating the base of the triangle.
.element	An <code>element_polygon()</code> object. Vectorized fields will be recycled to match the total number of groups, or you can wrap the element with <code>I()</code> to recycle to match the drawing groups. • When orientation is "plot", the drawing groups typically correspond to the number of observations. • When orientation is "observation", the drawing groups usually match the defined groups, but will differ if the defined group of observations is separated and cannot be linked with a single triangle. In this case, the number of drawing groups will be larger than the number of defined groups.

memo_order	<i>Sort matrix for better visualization</i>
------------	---

Description

Helper function used to order the Oncoplot samples. Typically, you would use this in combination with `align_order2()`, e.g., `align_order2(memo_order)`.

Usage

```
memo_order(x)
```

Arguments

x A matrix, where NA values will be treated as empty.

Value

A vector of ordering weights.

new_tune	<i>Change the shape of the input object</i>
----------	---

Description

- new_tune: Creates a new object by wrapping it in a scalar list with the specified attributes and class.
- tune_data: Retrieves the original input data.

Usage

```
new_tune(x, ..., class = character())
```

```
tune_data(x)
```

Arguments

x	An R object.
...	Additional attributes passed to <code>structure()</code> .
class	A character vector specifying the class name to be added.

no_expansion	<i>Remove scale expansion</i>
--------------	-------------------------------

Description

Remove scale expansion

Usage

```
no_expansion(borders = "tlbr")
```

Arguments

borders	Which border should be removed? A string containing one or more of "t", "l", "b", "r", "x", and "y".
---------	--

Value

An object which can be added to ggplot.

order2	<i>Ordering Permutation</i>
--------	-----------------------------

Description

order2 returns a permutation which rearranges its first argument into ascending order.

Usage

```
order2(x)

## S3 method for class 'hclust'
order2(x)

## S3 method for class 'dendrogram'
order2(x)

## S3 method for class 'ser_permutation_vector'
order2(x)
```

```
## S3 method for class 'ser_permutation'
order2(x)

## S3 method for class 'phylo'
order2(x)

## S3 method for class 'memo_weights'
order2(x)
```

Arguments

x Any objects can be extracting ordering.

Value

An integer vector unless any of the inputs has 2^{31} or more elements, when it is a double vector.

Examples

```
order2(hclust2(matrix(rnorm(100L), nrow = 10L)))
```

pair_links

Helper function to create pairs of observation groups

Description

[ggmark\(\)](#) and [cross_link\(\)](#) allow users to add links between observations. These functions help define the linked observations. The selected pairs will either be linked together, or each group in the pair will be linked separately to the same plot area.

- pair_links: Helper function to create pairs of observation groups.
- range_link: Helper function to create a range of observations.

Usage

```
pair_links(..., .handle_missing = "error", .reorder = NULL)

range_link(point1, point2)
```

Arguments

... [<dyn-dots>](#) A list of formulas, where each side of the formula should be an integer or character index of the original data, or a [range_link\(\)](#) object defining the linked observations. Use NULL to indicate no link on that side. You can also combine these by wrapping them into a single [list\(\)](#). If only the left-hand side of the formula exists, you can input it directly. For integer indices, wrap them with [I\(\)](#) to use the ordering from the layout. You can also use [waiver\(\)](#) to inherit values from the other group.

<code>.handle_missing</code>	A string of "error" or "remove" indicates the action for handling missing observations.
<code>.reorder</code>	A string of "hand1" or "hand2" indicating whether to reorder the input links to follow the specified layout ordering.
<code>point1, point2</code>	A single integer or character index, defining the lower and higher bounds of the range. For integer indices, wrap them with <code>I()</code> to indicate the ordered index by the layout.

Examples

```
x <- pair_links(
  # group on the left hand only
  c("a", "b"),
  # normally, integer index will be interpreted as the index of the
  # original data
  1:2,
  # wrapped with `I()` indicate the integer index is ordering of the
  # layout
  I(1:2),
  range_link(1, 6),
  range_link("a", "b"),
  # group on the right hand only
  ~ 1:2,
  ~ c("a", "b"),
  ~ range_link(1, 6),
  # group on the both side
  range_link(1, 6) ~ c("a", "b"),
  # waiver() indicates the right hand is the same of the left hand
  range_link(1, 6) ~ waiver(),
  # the same for the left hand
  waiver() ~ 1:2,
  ~NULL # an empty link
)
x

# we can modify it as usual list
x[[1]] <- NULL # remove the first link
x$a <- ~LETTERS
x

# modify with a list
x[1:2] <- list(~ c("a", "b"), ~ range_link("a", "b"))
x
```

Description

This function extends `ggplot2`'s title functionality, allowing you to add titles to each border of the plot: top, left, bottom, and right.

Usage

```
patch_title(  
  top = waiver(),  
  left = waiver(),  
  bottom = waiver(),  
  right = waiver()  
)
```

Arguments

top, left, bottom, right

A string specifying the title to be added to the top, left, bottom, and right border of the plot.

Details

The appearance and alignment of these patch titles can be customized using [theme\(\)](#):

- `plot.patch_title/plot.patch_title.*`: Controls the text appearance of patch titles. By default, `plot.patch_title` inherit from `plot.title`, and settings for each border will inherit from `plot.patch_title`, with the exception of the `angle` property, which is not inherited.
- `plot.patch_title.position/plot.patch_title.position.*`: Determines the alignment of the patch titles. By default, `plot.patch_title.position` inherit from `plot.title.position`, and settings for each border will inherit from `plot.patch_title`. The value "panel" aligns the patch titles with the plot panels. Setting this to "plot" aligns the patch title with the entire plot (excluding margins and plot tags).

Value

A [labels](#) object to be added to `ggplot`.

Examples

```
ggplot(mtcars) +  
  geom_point(aes(mpg, disp)) +  
  patch_title(  
    top = "I'm top patch title",  
    left = "I'm left patch title",  
    bottom = "I'm bottom patch title",  
    right = "I'm right patch title"  
  )
```

plot_ideogram	<i>Add an aligned cytoband ideogram plot</i>
---------------	--

Description

Creates a cytoband ideogram-typically representing chromosome banding patterns-and aligns it within a genomic layout.

Cytoband features (gieStain) are mapped to fill colors following standard cytogenetic conventions (e.g., gpos, gneg, acen, stalk). Optionally, chromosome names can be displayed as labels.

Usage

```
plot_ideogram(
  mapping = aes(),
  ...,
  seqnames = NULL,
  size = NULL,
  active = NULL
)
```

Arguments

- | | |
|----------|---|
| mapping | Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot. |
| ... | Arguments passed on to ggplot2::geom_text |
| stat | The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: <ul style="list-style-type: none"> • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation. |
| position | A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following: <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation. |

	<code>parse</code> If TRUE, the labels will be parsed into expressions and displayed as described in <code>?plotmath</code>. <code>size.unit</code> How the size aesthetic is interpreted: as millimetres ("mm", default), points ("pt"), centimetres ("cm"), inches ("in"), or picas ("pc"). <code>na.rm</code> If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed. <code>show.legend</code> logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted. <code>inherit.aes</code> If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>annotation_borders()</code>. <code>check_overlap</code> If TRUE, text that overlaps previous text in the same layer will not be plotted. <code>check_overlap</code> happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling <code>geom_text()</code>. Note that this argument is not supported by <code>geom_label()</code>.
seqnames	A single logical or numeric value controlling chromosome label display. Defaults to TRUE. • Logical (TRUE/FALSE): – TRUE: display labels at the default offset: * 1 above the ideogram (vertical layout) * -1 below the ideogram (horizontal layout) – FALSE: do not display labels. • Numeric: Specifies the vertical position of labels relative to the ideogram's y-axis: – Positive: above the ideogram (offset from the upper border) – Negative: below the ideogram (offset from the lower border) – 0: centered. Note: The cytoband vertical range spans from 0 to 1.
size	The relative size of the plot, can be specified as a <code>unit()</code>. Note that for <code>circle_layout()</code>, all size values will be interpreted as relative sizes, as this layout type adjusts based on the available space in the circular arrangement.
active	A <code>active()</code> object that defines the context settings when added to a layout.

Description**[Stable]**

- `quad_active`: Sets the active context to the `quad_layout()/ggheatmap()` itself.
- `quad_anno`: Sets the active context to the specified annotation stack based on the `position` argument.
- `anno_top`: A special case of `quad_anno` with `position = "top"`.
- `anno_left`: A special case of `quad_anno` with `position = "left"`.
- `anno_bottom`: A special case of `quad_anno` with `position = "bottom"`.
- `anno_right`: A special case of `quad_anno` with `position = "right"`.

Usage

```
quad_active(width = NULL, height = NULL)
```

```
quad_anno(  
  position,  
  size = NULL,  
  free_guides = waiver(),  
  initialize = NULL,  
  what = waiver()  
)
```

```
anno_top(  
  size = NULL,  
  free_guides = waiver(),  
  initialize = NULL,  
  what = waiver()  
)
```

```
anno_left(  
  size = NULL,  
  free_guides = waiver(),  
  initialize = NULL,  
  what = waiver()  
)
```

```
anno_bottom(  
  size = NULL,  
  free_guides = waiver(),  
  initialize = NULL,  
  what = waiver()  
)
```

```
anno_right(  
  size = NULL,  
  free_guides = waiver(),
```

```

    initialize = NULL,
    what = waiver()
  )

```

Arguments

width, height	The relative width/height of the main plot, can be a unit object.
position	A string of "top", "left", "bottom", or "right" indicates which annotation stack should be activated.
size	A numeric value or an unit object to set the total height/width of the annotation stack. • If position is "top" or "bottom", size sets the total height of the annotation. • If position is "left" or "right", size sets the total width of the annotation.
free_guides	Override the guides collection behavior specified in the quad_layout()/ggheatmap() for the annotation stack.
initialize	A boolean indicating whether the annotation stack should be initialized if it is not already. By default, the annotation stack layout will attempt to initialize when the data is compatible. If set to TRUE, and the data in quad_layout()/ggheatmap() is incompatible with the annotation stack, no data will be used in the stack.
what	What should get activated in the annotation stack? A single number or string of the plot elements in the layout. If NULL, will remove any active context.

Details

By default, `quad_anno()` attempts to initialize the annotation stack layout using data from [quad_layout\(\)/ggheatmap\(\)](#). However, in situations where you want to use different data for the annotation stack, you can set `initialize = FALSE` and then provide a custom `stack_layout()`.

Value

An object that can be added to [quad_layout\(\)/ggheatmap\(\)](#).

See Also

[quad_switch\(\)](#)

quad_layout*Arrange plots in the quad-side of a main plot*

Description

[Stable]

This function arranges plots around the quad-sides of a main plot, aligning both horizontal and vertical axes, and can handle either discrete or continuous variables.

- If `xlim` is provided, a continuous variable will be required and aligned in the vertical direction. Otherwise, a discrete variable will be required and aligned.
- If `ylim` is provided, a continuous variable will be required and aligned in the horizontal direction. Otherwise, a discrete variable will be required and aligned.

The `quad_discrete` is a special case where both `xlim` and `ylim` are not provided.

The `quad_continuous` is a special case where both `xlim` and `ylim` are provided.

For historical reasons, the following aliases are available:

- `quad_alignh`: Align discrete variables in the horizontal direction and continuous variables in vertical direction.
- `quad_alignv`: Align discrete variables in the vertical direction and continuous variables in horizontal direction.
- `quad_alignb` is an alias for `quad_discrete`.
- `quad_free` is an alias for `quad_continuous`.

Usage

```
quad_layout(  
  data = waiver(),  
  mapping = aes(),  
  xlim = waiver(),  
  ylim = waiver(),  
  ...,  
  theme = NULL,  
  active = NULL,  
  width = NA,  
  height = NA  
)  
  
quad_alignh(..., ylim = waiver())  
  
quad_alignv(..., xlim = waiver())  
  
quad_discrete(  
  data = waiver(),
```

```

mapping = aes(),
...,
theme = NULL,
active = NULL,
width = NA,
height = NA
)

quad_continuous(
  data = waiver(),
  mapping = aes(),
  xlim = NULL,
  ylim = NULL,
  ...,
  theme = NULL,
  active = NULL,
  width = NA,
  height = NA
)

```

Arguments

data	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout. By default, this will attempt to inherit from the parent layout. If both <code>xlim</code> and <code>ylim</code> are provided, a data frame is required, and <code>fortify_data_frame()</code> will be used to convert the data to a data frame. When inherited by an annotation stack, no transposition will be applied. Otherwise, a matrix is required, and <code>fortify_matrix()</code> will be used to convert the data to a matrix. When inherited by the column annotation stack, the data will be transposed.
mapping	Default list of aesthetic mappings to use for main plot in the layout. If not specified, must be supplied in each layer added to the main plot.
xlim, ylim	A <code>continuous_limits()</code> object specifying the left/lower limit and the right/upper limit of the scale. Used to align the continuous axis.
...	Additional arguments passed to <code>fortify_data_frame()</code> or <code>fortify_matrix()</code> .
theme	A <code>theme()</code> object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, <code>panel.border</code> , and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.
active	A <code>active()</code> object that defines the context settings when added to a layout.
width, height	The relative width/height of the main plot, can be a <code>unit</code> object.

Value

A `QuadLayout` object.

ggplot2 specification

If either `xlim` or `ylim` is not provided, the data input will be converted to a matrix using `fortify_matrix()`, and the data in the underlying main plot will contain the following columns:

- `.panel_x` and `.panel_y`: the column and row panel groups.
- `.x` and `.y`: an integer index of x and y coordinates
- `.discrete_x` and `.discrete_y`: a factor of the data labels (only applicable when `.row_names` and `.column_names` exists).
- `.row_names` and `.column_names`: A character of the row and column names of the original matrix (only applicable when names exist).
- `.row_index` and `.column_index`: the row and column index of the original matrix.
- `value`: the actual matrix value.

Otherwise, the data input will be used for the main plot.

quad_scope	<i>Modify operated Context in quad_layout()</i>
------------	---

Description

[Experimental]

The `quad_scope()` function controls how plot elements (e.g., themes, scales, or other ggplot objects) are applied within a `ggheatmap()` or `quad_layout()` context. It allows you to direct modifications to specific annotation positions or the main plot without altering the currently active layout or nesting state.

Usage

```
quad_scope(object = NULL, position = waiver())
```

Arguments

object	An object which can be added to the ggplot.
position	A string or character vector specifying one or more positions ("t", "l", "b", "r", and "i") indicating where x should be applied. Use 'i' to refer to the quad body (i.e., the main plot). If NULL, the active annotation context is cleared, behaving as if no annotation is active. See the Details section for more information.

Details

Default behavior when adding objects wrapped with `quad_scope()`:

- **When no annotation stack is active:** Modifications are applied normally without needing `quad_scope()`.
- **When an annotation stack is active:** `quad_scope()` ensures the object is also applied to:
 - The active annotation stack
 - The main plot

When position is manually specified:

- If `NULL`, it behaves as if no annotation is active
- If a string, the object is applied only to the specified positions (to include the main plot, explicitly add `"i"` to position)

Value

The original object with an added attribute that sets the specified context.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 7)

# By wrapping object with `quad_scope()`, the `+` operator will apply the
# object not only to the active plot in the annotation stack, but also to
# the main plot unless specified by `main` argument otherwise.
ggheatmap(small_mat) +
  # initialize the left annotation
  anno_left(size = 0.2) +
  align_dendro() +
  # apply the object not only to the active plot in the annotation stack,
  # but also to the main plot
  quad_scope(theme(plot.background = element_rect(fill = "red"))))

# When the `position` argument is manually set, the
# we must explicitly include `"i"` in `position` to apply it to the main plot
ggheatmap(small_mat) +
  anno_left(size = 0.2) +
  align_dendro(aes(color = branch), k = 3L) +
  anno_top(size = 0.2) +
  align_dendro(aes(color = branch), k = 3L) +
  anno_bottom(size = 0.2) +
  align_dendro(aes(color = branch), k = 3L) -
  # Modify the background of all plots in the left and top annotation
  quad_scope(theme(plot.background = element_rect(fill = "red")), "tl")
```

quad_switch

*Determine the Active Context of Quad-Layout***Description****[Stable]**

quad_switch() integrates [quad_active\(\)](#) and [quad_anno\(\)](#) into one function for ease of use. This function allows you to quickly change the active context of the [quad_layout\(\)](#) and its annotations.

hmanno is an alias for quad_switch, with additional arguments for backward compatibility

Usage

```
quad_switch(
  position = NULL,
  size = NULL,
  width = NULL,
  height = NULL,
  free_guides = waiver(),
  initialize = NULL,
  what = waiver()
)

hmanno(
  position = NULL,
  size = NULL,
  width = NULL,
  height = NULL,
  free_guides = waiver(),
  initialize = NULL,
  what = waiver()
)
```

Arguments

position	A string of "top", "left", "bottom", or "right" indicates which annotation stack should be activated. If NULL, it sets the active context to the quad_layout()/ggheatmap() itself.
size	A numeric value or an unit object to set the total height/width of the annotation stack. • If position is "top" or "bottom", size sets the total height of the annotation. • If position is "left" or "right", size sets the total width of the annotation.
width, height	The relative width/height of the main plot, can be a unit object.

free_guides	Override the guides collection behavior specified in the quad_layout()/ggheatmap() for the annotation stack.
initialize	A boolean indicating whether the annotation stack should be initialized if it is not already. By default, the annotation stack layout will attempt to initialize when the data is compatible. If set to TRUE, and the data in quad_layout()/ggheatmap() is incompatible with the annotation stack, no data will be used in the stack.
what	What should get activated in the annotation stack? A single number or string of the plot elements in the layout. If NULL, will remove any active context.

Value

An object that can be added to [quad_layout\(\)/ggheatmap\(\)](#).

See Also

[quad_active\(\)/quad_anno\(\)](#)

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
  anno_top() +
  align_dendro()
```

raster_magick

Rasterize the ggplot layers

Description

The function rasterizes input graphical objects (e.g., grob, layer, ggplot) and optionally processes the resulting raster using magick, a powerful image manipulation library. This allows for advanced graphical transformations directly within the plotting pipeline.

Usage

```
raster_magick(x, ...)
```

Arguments

x	An object to rasterize, can be a grob() , layer() , ggplot() , or a list of such objects.
...	Additional arguments passed on to magickGrob() .

Value

An object with the same class of the input.

See Also[magickGrob\(\)](#)**Examples**

```
# Currently, `magick` package require R >= 4.1.0
if (requireNamespace("magick")) {
  # data generated code was copied from `ComplexHeatmap`
  set.seed(123)
  small_mat <- matrix(rnorm(56), nrow = 7)
  rownames(small_mat) <- paste0("row", seq_len(nrow(small_mat)))
  colnames(small_mat) <- paste0("column", seq_len(ncol(small_mat)))
  ggheatmap(small_mat, aes(.x, .y), filling = NULL) +
    raster_magick(geom_tile(aes(fill = value)), res = 20)

  ggheatmap(small_mat, aes(.x, .y), filling = NULL) +
    # Use `magick::filter_types()` to check available `filter` arguments
    raster_magick(
      geom_tile(aes(fill = value)),
      magick = function(image) {
        magick::image_resize(image,
          geometry = "50%x", filter = "Lanczos"
        )
      }
    )
}
```

read_example

*Read Example Data***Description**

This function reads example data from the file. If no file is specified, it returns a list of available example files.

Usage

```
read_example(file = NULL)
```

Arguments

file	A string representing the name of the example file to be read. If NULL, the function will return a list of available example file names.
------	--

Value

If file is NULL, returns a character vector of available example file names. Otherwise, returns the contents of the specified example file, read as an R object.

Examples

```
read_example()
```

```
scale_gshape_manual
```

Scale for gshape aesthetic

Description**[Questioning]**

`geom_gshape` depends on the new aesthetics `gshape` (shape with grid functions), which should always be provided with `scale_gshape_manual()`, in which, we can provide a list of grobs or functions that define how each value should be drawn. Any `ggplot2` aesthetics can be used as the arguments.

Usage

```
scale_gshape_manual(..., values, breaks = waiver(), na.value = NA)
```

Arguments

```
...
```

Arguments passed on to `ggplot2::discrete_scale`

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.

minor_breaks One of:

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

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

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

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 <code>rlang::lambda</code> function notation.
<code>na.translate</code>	Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify <code>na.translate = FALSE</code> .
<code>drop</code>	Should unused factor levels be omitted from the scale? The default, <code>TRUE</code> , uses the levels that appear in the data; <code>FALSE</code> includes the levels in the factor. Please note that to display every level in a legend, the layer should use <code>show.legend = TRUE</code> .
<code>guide</code>	A function used to create a guide or its name. See <code>guides()</code> for more information.
<code>call</code>	The call used to construct the scale for reporting messages.
<code>super</code>	The super class to use for the constructed scale
<code>values</code>	A list of grobs or functions (including purrr-like lambda syntax) that define how each cell's grob (graphical object) should be drawn.
<code>breaks</code>	One of: • <code>NULL</code> for no breaks • <code>waiver()</code> 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
<code>na.value</code>	The aesthetic value to use for missing (NA) values

Life cycle

We're unsure whether this function is truly necessary, which is why it is marked as questioning. So far, we've found that `geom_subrect()` and `geom_subtile()` handle most use cases effectively.

Examples

```
library(grid)
ggplot(data.frame(value = letters[seq_len(5)], y = seq_len(5))) +
  geom_gshape(aes(x = 1, y = y, gshape = value, fill = value)) +
  scale_gshape_manual(values = list(
    a = function(x, y, width, height, fill) {
      rectGrob(x, y,
        width = width, height = height,
        gp = gpar(fill = fill),
        default.units = "native"
      )
    },
    b = function(x, y, width, height, fill) {
      rectGrob(x, y,
        width = width, height = height,
        gp = gpar(fill = fill),
        default.units = "native"
      )
    }
  ))
```

```

    },
    c = function(x, y, width, height, fill) {
      rectGrob(x, y,
        width = width, height = height,
        gp = gpar(fill = fill),
        default.units = "native"
      )
    },
    d = function(x, y, width, height, shape) {
      gList(
        pointsGrob(x, y, pch = shape),
        # To ensure the rectangle color is shown in the legends, you
        # must explicitly provide a color argument and include it in
        # the `gpar()` of the graphical object
        rectGrob(x, y, width, height,
          gp = gpar(col = "black", fill = NA)
        )
      )
    },
    e = function(xmin, xmax, ymin, ymax) {
      segmentsGrob(
        xmin, ymin,
        xmax, ymax,
        gp = gpar(lwd = 2)
      )
    }
  )) +
  scale_fill_brewer(palette = "Dark2") +
  theme_void()

```

scale_z_continuous *z scales*

Description

z scales

Usage

```
scale_z_continuous(name = waiver(), ..., range = c(0.1, 1), guide = "none")
```

```
scale_z_binned(name = waiver(), ..., range = c(0.1, 1), guide = "none")
```

```
scale_z_discrete(...)
```

```
scale_z_ordinal(name = waiver(), ..., range = c(0.1, 1), guide = "none")
```

Arguments

name	The name of the scale. Used as the axis or legend title. If <code>waiver()</code> , the default, the name of the scale is taken from the first mapping used for that aesthetic. If <code>NULL</code> , the legend title will be omitted.
...	Other arguments passed on to <code>continuous_scale()</code> , <code>binned_scale()</code> , or <code>discrete_scale()</code> as appropriate, to control name, limits, breaks, labels and so forth.
range	Output range of z values. Must larger than 0.
guide	A function used to create a guide or its name. See <code>guides()</code> for more information.

See Also

`geom_tile3d()/geom_rect3d()`

Examples

```
set.seed(7)
mat <- matrix(runif(100), 10)
rownames(mat) <- LETTERS[1:10]
colnames(mat) <- letters[1:10]
ggheatmap(mat,
  filling = FALSE,
  theme = theme(
    legend.box.spacing = unit(10, "mm"),
    plot.margin = margin(t = 15, unit = "mm")
  )
) +
  geom_tile3d(aes(fill = value, z = value, width = 0.8, height = 0.8)) +
  scale_z_continuous(range = c(0.2, 1)) +
  coord_cartesian(clip = "off")
```

scheme_align

Align Specifications in the Layout

Description**[Experimental]**

The `scheme_align()` function defines the align Specifications for plots.

Usage

```
scheme_align(guides = NA, free_spaces = NA, free_labs = NA)
```

Arguments

guides	A string with one or more of "t", "l", "b", "r", and "i" indicating which side of guide legends should be collected. Defaults to <code>waiver()</code> , which inherits from the parent layout. If no parent layout, all guides will be collected. If NULL, no guides will be collected.
free_spaces	A string with one or more of "t", "l", "b", and "r" indicating which border spaces should be removed. Defaults to <code>waiver()</code> , which inherits from the parent layout. If no parent, the default is NULL, meaning no spaces are removed. Usually you want to apply this with the whole layout, instead of individual plots.
free_labs	A string with one or more of "t", "l", "b", and "r" indicating which axis titles should be free from alignment. Defaults to <code>waiver()</code> , which inherits from the parent layout. If no parent layout, no axis titles will be aligned. If NULL, all axis titles will be aligned.

Value

A `scheme_align` object.

Examples

```
set.seed(123)
mat <- matrix(rnorm(72), nrow = 8)
# used in the layout, define the default action for all plots in the layout
ggheatmap(mat) -
  scheme_align(guides = NULL) +
  anno_right() +
  align_dendro(aes(color = branch), k = 3)

# You can also add it for a single plot
ggheatmap(mat) -
  # for all plots in the layout, we default won't collect any guide legends
  scheme_align(guides = NULL) +
  # for the heatmap body, we collect guide legends in the right
  # note, the guide legends will be collected to the right side of the
  # layout which will overlap the legends in the right annotation
  scheme_align(guides = "r") +
  anno_right() +
  align_dendro(aes(color = branch), k = 3)

# to avoid overlapping, we can also collect the guide legends in the
# right annotation
ggheatmap(mat) -
  scheme_align(guides = NULL) +
  scheme_align(guides = "r") +
  anno_right() +
  align_dendro(aes(color = branch), k = 3) +
  scheme_align(guides = "r")
```

scheme_data	<i>Plot data Specifications</i>
-------------	---------------------------------

Description

[Experimental]

Transforms the plot data. Many functions in this package require a specific data format to align observations, `scheme_data()` helps reformat data frames as needed.

Usage

```
scheme_data(data = NULL, inherit = FALSE)
```

Arguments

data	A function to transform the plot data before rendering. Acceptable values include: • <code>NULL</code>: No action taken. • <code>waiver()</code>: Inherits from the parent layout. • A function or purrr-style formula: Used to transform the plot data, which should accept a data frame and return a data frame. You can apply this after the parent layout <code>scheme_data</code> function, using the <code>inherit</code> argument. Use this hook to modify the data for all geoms after the layout is created (for matrix data, it has been melted to a long format data frame) but before rendering by <code>ggplot2</code>. The returned data must be a data frame for <code>ggplot</code>.
inherit	A single boolean value indicates whether to apply the parent <code>scheme_data</code> first and then apply the specified <code>scheme_data</code> for the plot. Defaults to <code>FALSE</code>.

Details

Defaults will attempt to inherit from the parent layout if the actual data is inherited from the parent layout, with one exception: `align_dendro()`, which will not inherit the `scheme_data` by default.

scheme_theme	<i>Plot default theme</i>
--------------	---------------------------

Description

[Experimental]

`scheme_theme()` serves as the default theme and will always be overridden by any `theme()` settings applied directly to the plot. The default theme (`scheme_theme()`) is applied first, followed by any specific `theme()` settings, even if `theme()` is added before `scheme_theme()`.

Usage

```
scheme_theme(  
  ...,  
  line,  
  rect,  
  text,  
  title,  
  point,  
  polygon,  
  geom,  
  spacing,  
  margins,  
  aspect.ratio,  
  axis.title,  
  axis.title.x,  
  axis.title.x.top,  
  axis.title.x.bottom,  
  axis.title.y,  
  axis.title.y.left,  
  axis.title.y.right,  
  axis.text,  
  axis.text.x,  
  axis.text.x.top,  
  axis.text.x.bottom,  
  axis.text.y,  
  axis.text.y.left,  
  axis.text.y.right,  
  axis.text.theta,  
  axis.text.r,  
  axis.ticks,  
  axis.ticks.x,  
  axis.ticks.x.top,  
  axis.ticks.x.bottom,  
  axis.ticks.y,  
  axis.ticks.y.left,  
  axis.ticks.y.right,  
  axis.ticks.theta,  
  axis.ticks.r,  
  axis.minor.ticks.x.top,  
  axis.minor.ticks.x.bottom,  
  axis.minor.ticks.y.left,  
  axis.minor.ticks.y.right,  
  axis.minor.ticks.theta,  
  axis.minor.ticks.r,  
  axis.ticks.length,  
  axis.ticks.length.x,  
  axis.ticks.length.x.top,  
  axis.ticks.length.x.bottom,
```

```
axis.ticks.length.y,  
axis.ticks.length.y.left,  
axis.ticks.length.y.right,  
axis.ticks.length.theta,  
axis.ticks.length.r,  
axis.minor.ticks.length,  
axis.minor.ticks.length.x,  
axis.minor.ticks.length.x.top,  
axis.minor.ticks.length.x.bottom,  
axis.minor.ticks.length.y,  
axis.minor.ticks.length.y.left,  
axis.minor.ticks.length.y.right,  
axis.minor.ticks.length.theta,  
axis.minor.ticks.length.r,  
axis.line,  
axis.line.x,  
axis.line.x.top,  
axis.line.x.bottom,  
axis.line.y,  
axis.line.y.left,  
axis.line.y.right,  
axis.line.theta,  
axis.line.r,  
legend.background,  
legend.margin,  
legend.spacing,  
legend.spacing.x,  
legend.spacing.y,  
legend.key,  
legend.key.size,  
legend.key.height,  
legend.key.width,  
legend.key.spacing,  
legend.key.spacing.x,  
legend.key.spacing.y,  
legend.key.justification,  
legend.frame,  
legend.ticks,  
legend.ticks.length,  
legend.axis.line,  
legend.text,  
legend.text.position,  
legend.title,  
legend.title.position,  
legend.position,  
legend.position.inside,  
legend.direction,  
legend.byrow,
```

```
legend.justification,  
legend.justification.top,  
legend.justification.bottom,  
legend.justification.left,  
legend.justification.right,  
legend.justification.inside,  
legend.location,  
legend.box,  
legend.box.just,  
legend.box.margin,  
legend.box.background,  
legend.box.spacing,  
panel.background,  
panel.border,  
panel.spacing,  
panel.spacing.x,  
panel.spacing.y,  
panel.grid,  
panel.grid.major,  
panel.grid.minor,  
panel.grid.major.x,  
panel.grid.major.y,  
panel.grid.minor.x,  
panel.grid.minor.y,  
panel.ontop,  
panel.widths,  
panel.heights,  
plot.background,  
plot.title,  
plot.title.position,  
plot.subtitle,  
plot.caption,  
plot.caption.position,  
plot.tag,  
plot.tag.position,  
plot.tag.location,  
plot.margin,  
strip.background,  
strip.background.x,  
strip.background.y,  
strip.clip,  
strip.placement,  
strip.text,  
strip.text.x,  
strip.text.x.bottom,  
strip.text.x.top,  
strip.text.y,  
strip.text.y.left,
```

```

strip.text.y.right,
strip.switch.pad.grid,
strip.switch.pad.wrap,
complete = FALSE,
validate = TRUE
)

```

Arguments

... A `theme()` object or additional element specifications not part of base `ggplot2`. In general, these should also be defined in the `element_tree` argument. [Splicing](#) a list is also supported.

line all line elements ([element_line\(\)](#))

rect all rectangular elements ([element_rect\(\)](#))

text all text elements ([element_text\(\)](#))

title all title elements: plot, axes, legends ([element_text\(\)](#); inherits from `text`)

point all point elements ([element_point\(\)](#))

polygon all polygon elements ([element_polygon\(\)](#))

geom defaults for geoms ([element_geom\(\)](#))

spacing all spacings ([unit\(\)](#))

margins all margins ([margin\(\)](#))

aspect.ratio aspect ratio of the panel

axis.title, axis.title.x, axis.title.y, axis.title.x.top,
axis.title.x.bottom, axis.title.y.left, axis.title.y.right
labels of axes ([element_text\(\)](#)). Specify all axes' labels (`axis.title`), labels by plane (using `axis.title.x` or `axis.title.y`), or individually for each axis (using `axis.title.x.bottom`, `axis.title.x.top`, `axis.title.y.left`, `axis.title.y.right`). `axis.title.*` inherits from `axis.title` which inherits from `axis.title`, which in turn inherits from `text`

axis.text, axis.text.x, axis.text.y, axis.text.x.top,
axis.text.x.bottom, axis.text.y.left, axis.text.y.right,
axis.text.theta, axis.text.r
tick labels along axes ([element_text\(\)](#)). Specify all axis tick labels (`axis.text`), tick labels by plane (using `axis.text.x` or `axis.text.y`), or individually for each axis (using `axis.text.x.bottom`, `axis.text.x.top`, `axis.text.y.left`, `axis.text.y.right`). `axis.text.*` inherits from `axis.text` which inherits from `axis.text`, which in turn inherits from `text`

axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom,
axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right,
axis.ticks.theta, axis.ticks.r
tick marks along axes ([element_line\(\)](#)). Specify all tick marks (`axis.ticks`), ticks by plane (using `axis.ticks.x` or `axis.ticks.y`), or individually for each axis (using `axis.ticks.x.bottom`, `axis.ticks.x.top`, `axis.ticks.y.left`, `axis.ticks.y.right`). `axis.ticks.*` inherits from `axis.ticks` which inherits from `axis.ticks`, which in turn inherits from `line`

`axis.minor.ticks.x.top,` `axis.minor.ticks.x.bottom,`
`axis.minor.ticks.y.left,` `axis.minor.ticks.y.right,`
`axis.minor.ticks.theta, axis.minor.ticks.r`
 minor tick marks along axes (`element_line()`). `axis.minor.ticks.*.*` inherit from the corresponding major ticks `axis.ticks.*.*`.

`axis.ticks.length,` `axis.ticks.length.x,` `axis.ticks.length.x.top,`
`axis.ticks.length.x.bottom,` `axis.ticks.length.y,`
`axis.ticks.length.y.left,` `axis.ticks.length.y.right,`
`axis.ticks.length.theta, axis.ticks.length.r`
 length of tick marks (unit). `axis.ticks.length` inherits from `spacing`.

`axis.minor.ticks.length,` `axis.minor.ticks.length.x,`
`axis.minor.ticks.length.x.top,` `axis.minor.ticks.length.x.bottom,`
`axis.minor.ticks.length.y,` `axis.minor.ticks.length.y.left,`
`axis.minor.ticks.length.y.right,` `axis.minor.ticks.length.theta,`
`axis.minor.ticks.length.r`
 length of minor tick marks (unit), or relative to `axis.ticks.length` when provided with `rel()`.

`axis.line,` `axis.line.x,` `axis.line.x.top,` `axis.line.x.bottom,`
`axis.line.y,` `axis.line.y.left,` `axis.line.y.right,` `axis.line.theta,`
`axis.line.r`
 lines along axes (`element_line()`). Specify lines along all axes (`axis.line`), lines for each plane (using `axis.line.x` or `axis.line.y`), or individually for each axis (using `axis.line.x.bottom`, `axis.line.x.top`, `axis.line.y.left`, `axis.line.y.right`). `axis.line.*.*` inherits from `axis.line.*` which inherits from `axis.line`, which in turn inherits from `line`.

`legend.background`
 background of legend (`element_rect()`; inherits from `rect`)

`legend.margin` the margin around each legend (`margin()`; inherits from `margins`).

`legend.spacing,` `legend.spacing.x,` `legend.spacing.y`
 the spacing between legends (unit). `legend.spacing.x` & `legend.spacing.y` inherit from `legend.spacing` or can be specified separately. `legend.spacing` inherits from `spacing`.

`legend.key` background underneath legend keys (`element_rect()`; inherits from `rect`)

`legend.key.size,` `legend.key.height,` `legend.key.width`
 size of legend keys (unit); key background height & width inherit from `legend.key.size` or can be specified separately. In turn `legend.key.size` inherits from `spacing`.

`legend.key.spacing,` `legend.key.spacing.x,` `legend.key.spacing.y`
 spacing between legend keys given as a unit. Spacing in the horizontal (x) and vertical (y) direction inherit from `legend.key.spacing` or can be specified separately. `legend.key.spacing` inherits from `spacing`.

`legend.key.justification`
 Justification for positioning legend keys when more space is available than needed for display. The default, `NULL`, stretches keys into the available space. Can be a location like "center" or "top", or a two-element numeric vector.

`legend.frame` frame drawn around the bar (`element_rect()`).

`legend.ticks` tick marks shown along bars or axes (`element_line()`)

legend.ticks.length	length of tick marks in legend (<code>unit()</code>); inherits from <code>legend.key.size</code> .
legend.axis.line	lines along axes in legends (<code>element_line()</code>)
legend.text	legend item labels (<code>element_text()</code>); inherits from <code>text</code>)
legend.text.position	placement of legend text relative to legend keys or bars ("top", "right", "bottom" or "left"). The legend text placement might be incompatible with the legend's direction for some guides.
legend.title	title of legend (<code>element_text()</code>); inherits from <code>title</code>)
legend.title.position	placement of legend title relative to the main legend ("top", "right", "bottom" or "left").
legend.position	the default position of legends ("none", "left", "right", "bottom", "top", "inside")
legend.position.inside	A numeric vector of length two setting the placement of legends that have the "inside" position.
legend.direction	layout of items in legends ("horizontal" or "vertical")
legend.byrow	whether the legend-matrix is filled by columns (FALSE, the default) or by rows (TRUE).
legend.justification	anchor point for positioning legend inside plot ("center" or two-element numeric vector) or the justification according to the plot area when positioned outside the plot
legend.justification.top, legend.justification.left, legend.justification.inside	legend.justification.bottom, legend.justification.right,
	Same as <code>legend.justification</code> but specified per <code>legend.position</code> option.
legend.location	Relative placement of legends outside the plot as a string. Can be "panel" (default) to align legends to the panels or "plot" to align legends to the plot as a whole.
legend.box	arrangement of multiple legends ("horizontal" or "vertical")
legend.box.just	justification of each legend within the overall bounding box, when there are multiple legends ("top", "bottom", "left", "right", "center" or "centre")
legend.box.margin	margins around the full legend area, as specified using <code>margin()</code> ; inherits from <code>margins</code> .
legend.box.background	background of legend area (<code>element_rect()</code>); inherits from <code>rect</code>)
legend.box.spacing	The spacing between the plotting area and the legend box (unit); inherits from <code>spacing</code> .

<code>panel.background</code>	background of plotting area, drawn underneath plot (<code>element_rect()</code> ; inherits from <code>rect</code>)
<code>panel.border</code>	border around plotting area, drawn on top of plot so that it covers tick marks and grid lines. This should be used with <code>fill = NA</code> (<code>element_rect()</code> ; inherits from <code>rect</code>)
<code>panel.spacing</code> , <code>panel.spacing.x</code> , <code>panel.spacing.y</code>	spacing between facet panels (unit). <code>panel.spacing.x</code> & <code>panel.spacing.y</code> inherit from <code>panel.spacing</code> or can be specified separately. <code>panel.spacing</code> inherits from <code>spacing</code> .
<code>panel.grid</code> , <code>panel.grid.major</code> , <code>panel.grid.minor</code> , <code>panel.grid.major.x</code> , <code>panel.grid.major.y</code> , <code>panel.grid.minor.x</code> , <code>panel.grid.minor.y</code>	grid lines (<code>element_line()</code>). Specify major grid lines, or minor grid lines separately (using <code>panel.grid.major</code> or <code>panel.grid.minor</code>) or individually for each axis (using <code>panel.grid.major.x</code> , <code>panel.grid.minor.x</code> , <code>panel.grid.major.y</code> , <code>panel.grid.minor.y</code>). Y axis grid lines are horizontal and x axis grid lines are vertical. <code>panel.grid.*</code> inherits from <code>panel.grid.*</code> which inherits from <code>panel.grid</code> , which in turn inherits from <code>line</code>
<code>panel.ontop</code>	option to place the panel (background, gridlines) over the data layers (logical). Usually used with a transparent or blank <code>panel.background</code> .
<code>panel.widths</code> , <code>panel.heights</code>	Sizes for panels (units). Can be a single unit to set the total size for the panel area, or a unit vector to set the size of individual panels.
<code>plot.background</code>	background of the entire plot (<code>element_rect()</code> ; inherits from <code>rect</code>)
<code>plot.title</code>	plot title (text appearance) (<code>element_text()</code> ; inherits from <code>title</code>) left-aligned by default
<code>plot.title.position</code> , <code>plot.caption.position</code>	Alignment of the plot title/subtitle and caption. The setting for <code>plot.title.position</code> applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).
<code>plot.subtitle</code>	plot subtitle (text appearance) (<code>element_text()</code> ; inherits from <code>title</code>) left-aligned by default
<code>plot.caption</code>	caption below the plot (text appearance) (<code>element_text()</code> ; inherits from <code>title</code>) right-aligned by default
<code>plot.tag</code>	upper-left label to identify a plot (text appearance) (<code>element_text()</code> ; inherits from <code>title</code>) left-aligned by default
<code>plot.tag.position</code>	The position of the tag as a string ("topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright") or a coordinate. If a coordinate, can be a numeric vector of length 2 to set the x,y-coordinate relative to the whole plot. The coordinate option is unavailable for <code>plot.tag.location = "margin"</code> .

<code>plot.tag.location</code>	The placement of the tag as a string, one of "panel", "plot" or "margin". Respectively, these will place the tag inside the panel space, anywhere in the plot as a whole, or in the margin around the panel space.
<code>plot.margin</code>	margin around entire plot (unit with the sizes of the top, right, bottom, and left margins); inherits from margin.
<code>strip.background</code> , <code>strip.background.x</code> , <code>strip.background.y</code>	background of facet labels (element_rect() ; inherits from rect). Horizontal facet background (<code>strip.background.x</code>) & vertical facet background (<code>strip.background.y</code>) inherit from <code>strip.background</code> or can be specified separately
<code>strip.clip</code>	should strip background edges and strip labels be clipped to the extend of the strip background? Options are "on" to clip, "off" to disable clipping or "inherit" (default) to take the clipping setting from the parent viewport.
<code>strip.placement</code>	placement of strip with respect to axes, either "inside" or "outside". Only important when axes and strips are on the same side of the plot.
<code>strip.text</code> , <code>strip.text.x</code> , <code>strip.text.y</code> , <code>strip.text.x.top</code> , <code>strip.text.x.bottom</code> , <code>strip.text.y.left</code> , <code>strip.text.y.right</code>	facet labels (element_text() ; inherits from text). Horizontal facet labels (<code>strip.text.x</code>) & vertical facet labels (<code>strip.text.y</code>) inherit from <code>strip.text</code> or can be specified separately. Facet strips have dedicated position-dependent theme elements (<code>strip.text.x.top</code> , <code>strip.text.x.bottom</code> , <code>strip.text.y.left</code> , <code>strip.text.y.right</code>) that inherit from <code>strip.text.x</code> and <code>strip.text.y</code> , respectively. As a consequence, some theme stylings need to be applied to the position-dependent elements rather than to the parent elements
<code>strip.switch.pad.grid</code> , <code>strip.switch.pad.wrap</code>	space between strips and axes when strips are switched (unit); inherits from spacing.
<code>complete</code>	set this to TRUE if this is a complete theme, such as the one returned by theme_grey() . Complete themes behave differently when added to a ggplot object. Also, when setting <code>complete = TRUE</code> all elements will be set to inherit from blank elements.
<code>validate</code>	TRUE to run <code>check_element()</code> , FALSE to bypass checks.

Theme inheritance

Theme elements inherit properties from other theme elements hierarchically. For example, `axis.title.x.bottom` inherits from `axis.title.x` which inherits from `axis.title`, which in turn inherits from `text`. All text elements inherit directly or indirectly from `text`; all lines inherit from `line`, and all rectangular objects inherit from `rect`. This means that you can modify the appearance of multiple elements by setting a single high-level component.

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

See Also

[add_gg\(\)](#) and `%+replace%`, [element_blank\(\)](#), [element_line\(\)](#), [element_rect\(\)](#), and [element_text\(\)](#) for details of the specific theme elements.

The [modifying theme components](#) and [theme elements sections](#) of the online ggplot2 book.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 8)
ggheatmap(small_mat) +
  scheme_theme(plot.background = element_rect(fill = "red"))

# `scheme_theme()` serves as the default theme and will always be
# overridden by any `theme()` settings applied directly to the plot
ggheatmap(small_mat) +
  theme(plot.background = element_rect(fill = "blue")) +
  scheme_theme(plot.background = element_rect(fill = "red"))
```

stack_cross

Arrange plots crosswise horizontally or vertically

Description

[Experimental]

The `stack_cross` function is derived from `stack_discrete()` and allows for different layout ordering indices within a single layout.

Two aliases are provided for convenience:

- `stack_crossv`: A special case of `stack_cross` that sets `direction = "v"` for vertical alignment.
- `stack_crossh`: A special case of `stack_cross` that sets `direction = "h"` for horizontal alignment.

Usage

```
stack_cross(direction, data = NULL, ..., theme = NULL, sizes = NA)
```

```
stack_crossv(data = NULL, ...)
```

```
stack_crossh(data = NULL, ...)
```

Arguments

<code>direction</code>	A string indicating the direction of the stack layout, either "h"(horizontal) or "v"(vertical).
<code>data</code>	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout, <code>fortify_matrix()</code> will be used to convert the data to a matrix.
<code>...</code>	Additional arguments passed to <code>fortify_matrix()</code> .

theme	A theme() object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, panel.border, and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.
sizes	A numeric value or a unit object. When used for the quad_layout() annotation, it must be of length 1. When used in the stack_layout() with a nested quad_layout() , it should be of length 3, specifying the relative heights (for direction = "h") or widths (for direction = "v") to be applied to the layout.

See Also

[ggcross\(\)](#)

stack_genomic	<i>Create a stack Layout for Genomic Data</i>
---------------	---

Description

stack_genomic() constructs a stack layout specifically for genomic data. It is a specialized variant of stack_continuous() that applies default axis limits and coerces the first column of each plot's data to use chromosome (seqname) identifiers-matching those in the layout data-as factor levels.

Usage

```
stack_genomic(direction, data = NULL, ..., theme = NULL, sizes = NA)

stack_genomicv(data = NULL, ...)

stack_genomich(data = NULL, ...)
```

Arguments

direction	A string indicating the direction of the stack layout, either "h"(horizontal) or "v"(vertical).
data	The input data, which can be: • A character string ("hg19" or "hg38") to load a predefined cytoband reference. • A data.frame with at least three columns: chromosome, start, and end positions. • A genomic object convertible via fortify_data_frame().
...	Additional arguments passed to specific methods or fortify_data_frame() .
theme	A theme() object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, panel.border, and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.

sizes A numeric value or a `unit` object. When used for the `quad_layout()` annotation, it must be of length 1. When used in the `stack_layout()` with a nested `quad_layout()`, it should be of length 3, specifying the relative heights (for `direction = "h"`) or widths (for `direction = "v"`) to be applied to the layout.

Value

A `stack_layout` object representing the genomic layout.

<code>stack_layout</code>	<i>Arrange plots horizontally or vertically</i>
---------------------------	---

Description

[Stable]

If `limits` is provided, a continuous variable will be required and aligned in the direction specified (`stack_continuous`). Otherwise, a discrete variable will be required and aligned (`stack_discrete`).

Several aliases are provided for convenience:

- `stack_vertical`: A special case of `stack_layout` that sets `direction = "v"`.
- `stack_horizontal`: A special case of `stack_layout` that sets `direction = "h"`.
- `stack_discretev`: A special case of `stack_discrete` that sets `direction = "v"`.
- `stack_discreteh`: A special case of `stack_discrete` that sets `direction = "h"`.
- `stack_continuousv()`: A special case of `stack_free` that sets `direction = "v"`.
- `stack_continuoush()`: A special case of `stack_free` that sets `direction = "h"`.

For historical reasons, the following aliases are available:

- `stack_align` is an alias for `stack_discrete`.
- `stack_alignv` is an alias for `stack_discretev`.
- `stack_alighh` is an alias for `stack_discreteh`.
- `stack_free` is an alias for `stack_continuous`.
- `stack_freev` is an alias for `stack_continuousv`.
- `stack_freeh` is an alias for `stack_continuoush`.

Usage

```
stack_layout(
  direction,
  data = NULL,
  ...,
  theme = NULL,
  sizes = NA,
  limits = waiver())
```

```

)

stack_horizontal(data = NULL, ..., limits = waiver())

stack_vertical(data = NULL, ..., limits = waiver())

stack_discrete(direction, data = NULL, ..., theme = NULL, sizes = NA)

stack_discretev(data = NULL, ...)

stack_discreteh(data = NULL, ...)

stack_continuous(
  direction,
  data = NULL,
  ...,
  limits = NULL,
  theme = NULL,
  sizes = NA
)

stack_continuousv(data = NULL, ...)

stack_continuoush(data = NULL, ...)

```

Arguments

direction	A string indicating the direction of the stack layout, either "h"(horizontal) or "v"(vertical).
data	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout: • If <code>limits</code> is not provided, <code>fortify_matrix()</code> will be used to get a matrix. • If <code>limits</code> is specified, <code>fortify_data_frame()</code> will be used to get a data frame.
...	Additional arguments passed to <code>fortify_data_frame()</code> or <code>fortify_matrix()</code> .
theme	A <code>theme()</code> object used to customize various elements of the layout, including guides, title, subtitle, caption, margins, <code>panel.border</code> , and background. By default, the theme will inherit from the parent layout. It also controls the panel spacing for all plots in the layout.
sizes	A numeric value or a <code>unit</code> object. When used for the <code>quad_layout()</code> annotation, it must be of length 1. When used in the <code>stack_layout()</code> with a nested <code>quad_layout()</code> , it should be of length 3, specifying the relative heights (for <code>direction = "h"</code>) or widths (for <code>direction = "v"</code>) to be applied to the layout.
limits	A <code>continuous_limits()</code> object specifying the left/lower limit and the right/upper limit of the scale. Used to align the continuous axis.

Value

A StackLayout object.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 7L)

stack_horizontal(small_mat) + align_dendro()

# this is the same with:
stack_discrete("h", small_mat) + align_dendro()

stack_discreteh(small_mat) + align_dendro()

# For vertical layout:
stack_vertical(small_mat) + align_dendro()
```

stack_switch	<i>Determine the active context of stack layout</i>
--------------	---

Description**[Stable]**

stack_active is an alias for stack_switch(), which sets what = NULL by default.

Usage

```
stack_switch(sizes = NULL, what = waiver(), ...)

stack_active(sizes = NULL, ...)
```

Arguments

sizes	A numeric value or a unit object. When used for the quad_layout() annotation, it must be of length 1. When used in the stack_layout() with a nested quad_layout() , it should be of length 3, specifying the relative heights (for direction = "h") or widths (for direction = "v") to be applied to the layout.
what	What should get activated for the stack layout? A single number or string of the plot elements in the layout. If NULL, will remove any active context, this is useful when the active context is a quad_layout() object, where any align_*() will be added to the quad_layout() . By removing the active context, we can add align_*() into the stack_layout() .
...	These dots are for future extensions and must be empty.

Value

A `stack_switch` object which can be added to `stack_layout()`.

Examples

```
stack_discrete("h", matrix(1:9, nrow = 3L)) +
  ggheatmap() +
  # ggheatmap will set the active context, directing following addition
  # into the heatmap plot area. To remove the heatmap active context,
  # we can use `stack_active()` which will direct subsequent addition into
  # the stack
  stack_active() +
  # here we add a dendrogram to the stack.
  align_dendro()
```

theme_no_axes	<i>Remove axis elements</i>
---------------	-----------------------------

Description

Remove axis elements

Usage

```
theme_no_axes(
  axes = "xy",
  text = TRUE,
  ticks = TRUE,
  title = TRUE,
  line = FALSE
)
```

Arguments

axes	Which axes elements should be removed? A string containing one or more of "t", "l", "b", "r", "x", and "y".
text	If TRUE, will remove the axis labels.
ticks	If TRUE, will remove the axis ticks.
title	If TRUE, will remove the axis title.
line	If TRUE, will remove the axis line.

Value

A `theme()` object.

Examples

```
p <- ggplot() +
  geom_point(aes(x = wt, y = qsec), data = mtcars)
p + theme_no_axes()
p + theme_no_axes("b")
p + theme_no_axes("l")
```

tune

Change the shape of the input object

Description

Change the shape of the input object

Usage

```
tune(data, shape = NULL)
```

Arguments

data	An R object.
shape	Usually NULL or a string, specifying the new shape for the object. Refer to the detailed method for allowed values.

Details

In most cases, `fortify_matrix()` or `fortify_data_frame()` provide full support for transforming objects. However, some objects may require two completely different approaches to be fortified. The `tune` function acts as a helper to create a new class tailored for these objects.

tune **method collections**

- `tune.list()`
- `tune.MAF()`
- `tune.matrix()`

tune.list	<i>Convert the shape of a list for fortify method</i>
-----------	---

Description

Convert the shape of a list for fortify method

Usage

```
## S3 method for class 'list'  
tune(data, shape = NULL)
```

Arguments

data	A list
shape	Not used currently.

See Also

[fortify_matrix.list_upset\(\)](#)

Other tune: [tune.MAF\(\)](#), [tune.matrix\(\)](#)

tune.MAF	<i>Convert the shape of a MAF for fortify method</i>
----------	--

Description

Convert the shape of a MAF for fortify method

Usage

```
## S3 method for class 'MAF'  
tune(data, shape = NULL)
```

Arguments

data	A MAF object.
shape	Not used currently.

See Also

[fortify_matrix.MAF_pathways\(\)](#)

Other tune: [tune.list\(\)](#), [tune.matrix\(\)](#)

tune.matrix	<i>Convert the shape of a matrix for fortify method</i>
-------------	---

Description

Convert the shape of a matrix for fortify method

Usage

```
## S3 method for class 'matrix'  
tune(data, shape)
```

Arguments

data	A matrix.
shape	A string of "upset" or "oncoplot".

See Also

- `fortify_matrix.matrix()`
- `fortify_matrix.matrix_upset()`
- `fortify_matrix.matrix_oncoplot()`

Other tune: `tune.MAF()`, `tune.list()`

Index

- * **as_grob**
 - as_grob.formula, 20
 - as_grob.ggalignment::alignpatches, 21
 - as_grob.ggplot, 22
 - as_grob.gList, 23
 - as_grob.grob, 23
 - as_grob.Heatmap, 24
 - as_grob.patch, 25
 - as_grob.patchwork, 26
 - as_grob.pheatmap, 27
 - as_grob.recordedplot, 28
 - as_grob.trellis, 29
- * **fortify_data_frame**
 - fortify_data_frame.character, 44
 - fortify_data_frame.default, 45
 - fortify_data_frame.dendrogram, 46
 - fortify_data_frame.GRanges, 48
 - fortify_data_frame.matrix, 49
 - fortify_data_frame.phylo, 50
- * **fortify_matrix**
 - fortify_matrix.default, 53
 - fortify_matrix.GISTIC, 53
 - fortify_matrix.list_upset, 55
 - fortify_matrix.MAF, 56
 - fortify_matrix.matrix, 58
 - fortify_matrix.matrix_oncoplot, 59
 - fortify_matrix.matrix_upset, 60
- * **tune**
 - tune.list, 161
 - tune.MAF, 161
 - tune.matrix, 162
- .link_draw, 5
- .link_draw(), 119
- .mark_draw, 6
- .mark_draw(), 122
- %+replace%, 117, 153
- active, 7
- active(), 12, 14–17, 19, 38, 39, 86, 90, 91, 93, 95, 97, 101, 130, 134
- add_gg(), 117, 153
- aes(), 67, 69, 73, 75, 78, 82
- align_dendro, 10
- align_dendro(), 46, 47, 50, 51
- align_group, 12
- align_hclust, 13
- align_kmeans, 14
- align_order, 15
- align_order(), 95
- align_order2, 16
- align_order2(), 95, 124
- align_phylo, 18
- align_plots, 61
- align_plots(alignpatches), 7
- align_plots(), 22, 98
- alignpatches, 7, 61
- alpha, 71, 74, 77, 80, 84
- anno_bottom(quad_active), 130
- anno_left(quad_active), 130
- anno_right(quad_active), 130
- anno_top(quad_active), 130
- annotation_borders(), 68, 71, 74, 77, 80, 84, 130
- area, 19
- area(), 8, 106
- as.mask, 62
- as.matrix(), 52, 53
- as.path, 62
- as.raster(), 73, 121
- as_grob(), 98, 102, 121
- as_grob.formula, 20, 22–30
- as_grob.formula(), 21, 25–30
- as_grob.function(as_grob.formula), 20
- as_grob.ggalignment::alignpatches, 21, 21, 22–30
- as_grob.ggalignment::alignpatches(), 21, 25–30
- as_grob.ggplot, 21, 22, 22, 23–30
- as_grob.ggplot(), 21, 24, 26–30

- as_grob.gList, [21](#), [22](#), [23](#), [24–30](#)
- as_grob.gList(), [21](#), [24](#), [26–30](#)
- as_grob.grob, [21–23](#), [23](#), [25–30](#)
- as_grob.grob(), [21](#), [24](#), [26–30](#)
- as_grob.Heatmap, [21–24](#), [24](#), [26–30](#)
- as_grob.Heatmap(), [21](#), [25–30](#)
- as_grob.HeatmapAnnotation
(as_grob.Heatmap), [24](#)
- as_grob.HeatmapList(as_grob.Heatmap),
[24](#)
- as_grob.patch, [21–25](#), [25](#), [27–30](#)
- as_grob.patch(), [21](#), [25–30](#)
- as_grob.patch_ggplot, [21–30](#)
- as_grob.patch_ggplot(), [21](#), [24](#), [26–30](#)
- as_grob.patchwork, [21–26](#), [26](#), [28–30](#)
- as_grob.patchwork(), [21](#), [25–30](#)
- as_grob.pheatmap, [21–27](#), [27](#), [29](#), [30](#)
- as_grob.pheatmap(), [21](#), [25–30](#)
- as_grob.recordedplot, [21–28](#), [28](#), [30](#)
- as_grob.recordedplot(), [21](#), [25–30](#)
- as_grob.trellis, [21–29](#), [29](#)
- as_grob.trellis(), [21](#), [25–30](#)

- binned_scale(), [143](#)

- circle_continuous(circle_layout), [32](#)
- circle_discrete(circle_layout), [32](#)
- circle_genomic, [31](#)
- circle_layout, [32](#)
- circle_layout(), [11](#), [19](#), [35](#), [38](#), [39](#), [86](#), [90](#),
[91](#), [93](#), [130](#)
- circle_switch, [34](#)
- colour, [71](#), [77](#), [80](#), [84](#)
- complete themes, [109](#)
- continuous_limits, [35](#)
- continuous_limits(), [33](#), [134](#), [157](#)
- continuous_scale(), [143](#)
- coord_cartesian(), [36](#)
- coord_circle, [36](#)
- coord_circle(), [31](#), [33](#), [34](#), [42](#)
- coord_radial(), [31](#), [33](#), [34](#), [36](#), [37](#), [42](#)
- cor(), [100](#)
- cross_link, [37](#)
- cross_link(), [126](#)
- cross_mark, [38](#)
- cross_none, [39](#)
- cutree, [47](#)
- cutree(), [11](#), [14](#)

- dendrogram, [11](#), [14](#), [17](#), [46](#), [50](#)
- discrete_scale(), [143](#)
- dist, [11](#), [13](#), [99](#)
- dist(), [100](#)
- draw(), [24](#)
- draw_key_gshape, [40](#)
- dyn-dots, [6](#), [8](#), [10](#), [16–18](#), [38–40](#), [86](#), [89](#), [91](#),
[93](#), [119](#), [120](#), [122](#), [123](#), [126](#)

- element, [41](#), [42](#)
- element_blank(), [117](#), [153](#)
- element_geom(), [112](#), [149](#)
- element_line(), [112–115](#), [117](#), [119](#), [122](#),
[149–153](#)
- element_point(), [112](#), [149](#)
- element_polygon(), [112](#), [120](#), [123](#), [124](#), [149](#)
- element_rect(), [112–117](#), [149–153](#)
- element_rep(element_vec), [41](#)
- element_rep_len(element_vec), [41](#)
- element_text(), [112–117](#), [149](#), [151–153](#)
- element_vec, [41](#)
- element_vec_fields(element_vec), [41](#)
- element_vec_recycle(element_vec), [41](#)
- element_vec_rep(element_vec), [41](#)
- element_vec_rep_each(element_vec), [41](#)
- element_vec_slice(element_vec), [41](#)

- facet_grid, [47](#), [51](#)
- facet_sector, [42](#)
- facet_wrap(), [8](#), [84](#), [106](#)
- fill, [71](#), [74](#), [77](#), [80](#), [81](#), [84](#)
- fortify(), [45](#), [67](#), [69](#), [73](#), [75](#), [79](#), [82](#)
- fortify_data_frame, [43](#)
- fortify_data_frame(), [31](#), [33](#), [45](#), [48](#), [50](#),
[52](#), [86](#), [88](#), [89](#), [91–93](#), [134](#), [155](#), [157](#),
[160](#)
- fortify_data_frame.character, [44](#), [45](#), [48](#),
[50](#), [52](#)
- fortify_data_frame.character(), [44](#)
- fortify_data_frame.complex
(fortify_data_frame.character),
[44](#)
- fortify_data_frame.default, [45](#), [45](#), [48](#),
[50](#), [52](#)
- fortify_data_frame.default(), [44](#)
- fortify_data_frame.DelayedMatrix
(fortify_data_frame.matrix), [49](#)
- fortify_data_frame.dendrogram, [45](#), [46](#),
[48](#), [50](#), [52](#)

- fortify_data_frame.dendrogram(), [12](#), [44](#)
- fortify_data_frame.factor
(fortify_data_frame.character),
[44](#)
- fortify_data_frame.GRanges, [45](#), [48](#), [48](#),
[50](#), [52](#)
- fortify_data_frame.GRanges(), [44](#)
- fortify_data_frame.hclust
(fortify_data_frame.dendrogram),
[46](#)
- fortify_data_frame.logical
(fortify_data_frame.character),
[44](#)
- fortify_data_frame.Matrix
(fortify_data_frame.matrix), [49](#)
- fortify_data_frame.matrix, [45](#), [48](#), [49](#), [52](#)
- fortify_data_frame.matrix(), [44](#)
- fortify_data_frame.numeric
(fortify_data_frame.character),
[44](#)
- fortify_data_frame.phylo, [45](#), [48](#), [50](#), [50](#)
- fortify_data_frame.phylo(), [44](#)
- fortify_matrix, [52](#)
- fortify_matrix(), [33](#), [38–40](#), [53–55](#), [58](#), [60](#),
[61](#), [88](#), [89](#), [95](#), [97](#), [101](#), [134](#), [135](#),
[154](#), [157](#), [160](#)
- fortify_matrix.default, [53](#), [54](#), [55](#), [58](#), [60](#),
[61](#)
- fortify_matrix.default(), [52](#)
- fortify_matrix.GISTIC, [53](#), [53](#), [55](#), [58](#), [60](#),
[61](#)
- fortify_matrix.GISTIC(), [52](#)
- fortify_matrix.list
(fortify_matrix.list_upset), [55](#)
- fortify_matrix.list_upset, [53](#), [54](#), [55](#), [58](#),
[60](#), [61](#), [97](#)
- fortify_matrix.list_upset(), [52](#), [161](#)
- fortify_matrix.MAF, [53–55](#), [56](#), [58](#), [60](#), [61](#)
- fortify_matrix.MAF(), [52](#), [88](#)
- fortify_matrix.MAF_pathways
(fortify_matrix.MAF), [56](#)
- fortify_matrix.MAF_pathways(), [161](#)
- fortify_matrix.matrix, [53–55](#), [58](#), [58](#), [60](#),
[61](#)
- fortify_matrix.matrix(), [52](#), [162](#)
- fortify_matrix.matrix_oncoplot, [53–55](#),
[58](#), [59](#), [61](#)
- fortify_matrix.matrix_oncoplot(), [52](#),
[58](#), [162](#)
- fortify_matrix.matrix_upset, [53–55](#), [58](#),
[60](#), [60](#), [97](#)
- fortify_matrix.matrix_upset(), [52](#), [58](#),
[162](#)
- free_align, [61](#)
- free_border (free_align), [61](#)
- free_guide (free_align), [61](#)
- free_lab (free_align), [61](#)
- free_space (free_align), [61](#)
- free_vp (free_align), [61](#)
- genomic_density, [64](#)
- genomic_dist, [65](#)
- geom_draw, [66](#)
- geom_gshape, [69](#)
- geom_gshape(), [40](#)
- geom_line(), [97](#)
- geom_magick, [72](#)
- geom_pie, [75](#)
- geom_point(), [97](#)
- geom_raster(), [101](#)
- geom_rect(), [97](#)
- geom_rect3d, [78](#)
- geom_rect3d(), [143](#)
- geom_segment, [12](#)
- geom_segment(), [10](#), [18](#)
- geom_subrect, [81](#)
- geom_subrect(), [71](#), [141](#)
- geom_subtile (geom_subrect), [81](#)
- geom_subtile(), [71](#), [141](#)
- geom_tile(), [101](#)
- geom_tile3d (geom_rect3d), [78](#)
- geom_tile3d(), [143](#)
- ggalign, [12](#), [85](#)
- ggalign_attr, [88](#)
- ggalign_attr(), [89](#)
- ggalign_data_set, [89](#)
- ggalign_lvls (ggalign_attr), [88](#)
- ggalign_lvls(), [49](#), [57](#), [89](#)
- ggalign_stat, [89](#)
- ggalignGrob, [87](#)
- ggcross, [90](#)
- ggcross(), [155](#)
- ggfree, [91](#)
- ggheatmap (heatmap_layout), [100](#)
- ggheatmap(), [86](#), [88](#), [90](#), [95](#), [104](#), [105](#), [131](#),
[132](#), [137](#), [138](#)
- ggmark, [92](#)

- ggmark(), [126](#)
- ggoncoplot, [94](#)
- ggplot, [22](#), [61](#)
- ggplot(), [67](#), [69](#), [73](#), [75](#), [78](#), [82](#), [138](#)
- ggplot2::discrete_scale, [140](#)
- ggplot2::geom_text, [129](#)
- ggside (quad_layout), [133](#)
- ggupset, [96](#)
- ggwrap, [98](#)
- GISTIC, [54](#)
- gpar, [62](#)
- grid::gList, [68](#)
- grid::grid.grabExpr, [30](#)
- grid::gTree, [68](#)
- grid::viewport, [62](#)
- grob, [20–30](#), [67](#), [87](#), [98](#), [102](#)
- grob(), [5](#), [6](#), [87](#), [118](#), [121](#), [122](#), [138](#)
- group, [71](#), [74](#), [77](#), [80](#), [81](#), [84](#)
- gTree, [23](#)
- guides(), [141](#), [143](#)
- hclust, [11](#), [13](#), [14](#), [17](#), [46](#), [50](#), [99](#), [100](#)
- hclust(), [100](#)
- hclust2, [99](#)
- hclust2(), [14](#)
- Heatmap(), [25](#)
- heatmap_layout, [100](#)
- HeatmapAnnotation(), [25](#)
- hmanno (quad_switch), [137](#)
- I(), [6](#), [16](#), [119](#), [120](#), [122–124](#), [126](#), [127](#)
- image_read(), [73](#), [121](#)
- inset, [102](#)
- is_circle_layout (is_layout), [103](#)
- is_ggheatmap (is_layout), [103](#)
- is_heatmap_layout (is_layout), [103](#)
- is_layout, [103](#)
- is_quad_layout (is_layout), [103](#)
- is_stack_cross (is_layout), [103](#)
- is_stack_layout (is_layout), [103](#)
- key glyphs, [68](#), [70](#), [76](#), [80](#), [83](#)
- labels, [128](#)
- lambda, [140](#), [141](#)
- layer position, [67](#), [70](#), [73](#), [76](#), [79](#), [83](#), [129](#)
- layer stat, [67](#), [70](#), [73](#), [76](#), [79](#), [83](#), [129](#)
- layer(), [67](#), [68](#), [70](#), [76](#), [79](#), [80](#), [83](#), [138](#)
- layer_order, [104](#)
- layout-operator, [104](#)
- layout_design, [106](#)
- layout_design(), [9](#)
- layout_tags, [107](#)
- layout_tags(), [9](#)
- layout_theme, [109](#)
- layout_theme(), [9](#)
- layout_title, [117](#)
- layout_title(), [9](#)
- linetype, [71](#), [77](#), [80](#), [81](#), [84](#)
- linewidth, [71](#), [77](#), [80](#), [81](#), [84](#)
- link_draw, [118](#)
- link_draw(), [5](#), [6](#), [38](#)
- link_line, [119](#)
- link_line(), [38](#), [119](#)
- link_tetragon, [120](#)
- MAF, [56](#), [161](#)
- magickGrob, [120](#)
- magickGrob(), [138](#), [139](#)
- margin(), [112](#), [113](#), [115](#), [149–151](#)
- mark_draw, [6](#), [121](#)
- mark_draw(), [6](#), [39](#), [92](#)
- mark_line, [122](#)
- mark_line(), [39](#), [92](#), [122](#)
- mark_tetragon, [123](#)
- mark_tetragon(), [39](#), [92](#), [122](#)
- mark_triangle, [123](#)
- mark_triangle(), [122](#)
- memo_order, [124](#)
- new_tune, [124](#)
- no_expansion, [125](#)
- NROW(), [39](#), [86](#), [91](#), [93](#)
- order2, [125](#)
- order2(), [17](#)
- pair_links, [126](#)
- par(), [20](#)
- patch, [26](#)
- patch(), [8](#)
- patch_title, [127](#)
- patch_titles (patch_title), [127](#)
- patchGrob, [25](#)
- patchwork, [27](#)
- patchworkGrob, [26](#)
- pheatmap(), [28](#)
- phylo, [18](#)

- plot(), [21](#)
- plot_ideogram, [129](#)
- quad_active, [130](#)
- quad_active(), [137](#), [138](#)
- quad_alignb (quad_layout), [133](#)
- quad_alignh (quad_layout), [133](#)
- quad_alignv (quad_layout), [133](#)
- quad_anno (quad_active), [130](#)
- quad_anno(), [7](#), [137](#), [138](#)
- quad_continuous (quad_layout), [133](#)
- quad_discrete (quad_layout), [133](#)
- quad_discrete(), [96](#), [100](#)
- quad_free (quad_layout), [133](#)
- quad_layout, [133](#)
- quad_layout(), [86](#), [88](#), [90](#), [104](#), [105](#), [131](#),
[132](#), [137](#), [138](#), [155–158](#)
- quad_scope, [135](#)
- quad_switch, [137](#)
- quad_switch(), [132](#)
- range_link (pair_links), [126](#)
- raster_magick, [138](#)
- read_example, [139](#)
- recordPlot(), [29](#)
- recycled, [42](#)
- rel(), [31](#), [33](#), [42](#)
- rep(), [41](#)
- rep_len(), [41](#)
- scale_fill_continuous(), [101](#)
- scale_fill_discrete(), [101](#)
- scale_gshape_manual, [140](#)
- scale_gshape_manual(), [69](#), [140](#)
- scale_z_binned (scale_z_continuous), [142](#)
- scale_z_continuous, [142](#)
- scale_z_continuous(), [80](#)
- scale_z_date (scale_z_continuous), [142](#)
- scale_z_datetime (scale_z_continuous),
[142](#)
- scale_z_discrete (scale_z_continuous),
[142](#)
- scale_z_ordinal (scale_z_continuous),
[142](#)
- scheme_align, [143](#)
- scheme_data, [145](#)
- scheme_data(), [38](#), [39](#), [86](#), [91](#), [93](#)
- scheme_theme, [145](#)
- shape, [71](#)
- size, [71](#), [74](#)
- Splicing, [112](#), [149](#)
- stack_active (stack_switch), [158](#)
- stack_align (stack_layout), [156](#)
- stack_alignh (stack_layout), [156](#)
- stack_alignv (stack_layout), [156](#)
- stack_continuous (stack_layout), [156](#)
- stack_continuoush (stack_layout), [156](#)
- stack_continuousv (stack_layout), [156](#)
- stack_cross, [154](#)
- stack_cross(), [90](#)
- stack_crossh (stack_cross), [154](#)
- stack_crossv (stack_cross), [154](#)
- stack_discrete (stack_layout), [156](#)
- stack_discrete(), [154](#)
- stack_discreteh (stack_layout), [156](#)
- stack_discretev (stack_layout), [156](#)
- stack_free (stack_layout), [156](#)
- stack_freeh (stack_layout), [156](#)
- stack_freev (stack_layout), [156](#)
- stack_genomic, [155](#)
- stack_genomich (stack_genomic), [155](#)
- stack_genomicv (stack_genomic), [155](#)
- stack_horizontal (stack_layout), [156](#)
- stack_layout, [156](#)
- stack_layout(), [88](#), [90](#), [104](#), [105](#), [155–159](#)
- stack_switch, [158](#)
- stack_switch(), [7](#)
- stack_vertical (stack_layout), [156](#)
- stats::kmeans, [15](#)
- structure(), [125](#)
- theme(), [9](#), [12](#), [19](#), [31](#), [33](#), [86](#), [90](#), [95](#), [97](#), [101](#),
[112](#), [117](#), [128](#), [134](#), [149](#), [155](#), [157](#),
[159](#)
- theme_grey(), [116](#), [153](#)
- theme_no_axes, [159](#)
- theme_update(), [109](#)
- trellis, [30](#)
- tune, [160](#)
- tune.list, [161](#), [161](#), [162](#)
- tune.list(), [55](#), [160](#)
- tune.MAF, [161](#), [161](#), [162](#)
- tune.MAF(), [56](#), [160](#)
- tune.matrix, [161](#), [162](#)
- tune.matrix(), [59–61](#), [160](#)
- tune_data (new_tune), [124](#)
- unit, [95](#), [97](#), [101](#), [132](#), [134](#), [137](#), [155–158](#)

`unit()`, [11](#), [19](#), [38](#), [39](#), [86](#), [90](#), [91](#), [93](#), [112](#),
[114](#), [130](#), [149](#), [151](#)

`vars()`, [42](#)

`vec_names()`, [39](#), [86](#), [91](#), [93](#)

`vec_recycle()`, [41](#)

`vec_rep()`, [41](#)

`vec_rep_each()`, [41](#)

`vec_size()`, [39](#), [86](#), [91](#), [93](#)

`vec_slice()`, [41](#)

`viewport`, [61](#), [98](#), [102](#)

`waiver()`, [6](#), [8](#), [86](#), [91](#), [93](#), [106](#), [119](#), [120](#), [122](#),
[123](#), [126](#), [144](#), [145](#)

`x`, [71](#), [74](#), [77](#), [80](#), [84](#)

`xmax`, [80](#), [84](#)

`xmin`, [80](#), [84](#)

`y`, [71](#), [74](#), [77](#), [80](#), [84](#)

`ymax`, [80](#), [84](#)

`ymin`, [80](#), [84](#)