Package ‘ggalign’

October 12, 2024

Title A 'ggplot2' Extension for Consistent Axis Alignment

Version 0.0.4

Description A 'ggplot2' extension offers various tools for organizing and arranging plots. It is designed to consistently align a specific axis across multiple 'ggplot' objects, making it especially useful for plots requiring data order manipulation. A typical use case includes organizing combinations like a dendrogram and a heatmap.

License MIT + file LICENSE

URL https://github.com/Yunuuuu/ggalign,

https://yunuuuu.github.io/ggalign/

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

Depends ggplot2 (>= 3.3.0)

Imports cli, data.table, grid, gtable, methods, rlang (>= 1.1.0), stats, utils, vctrs

Suggests bookdown, ggrastr, gridGraphics, knitr, patchwork, ragg, rmarkdown, scales, seriation, testthat (>= 3.0.0), vdiffr

ByteCompile true

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Yun Peng [aut, cre, cph] (<https://orcid.org/0000-0003-2801-3332>)

Maintainer Yun Peng <yunyunp96@163.com>

Repository CRAN

Date/Publication 2024-10-12 13:10:02 UTC

Contents

align_dendro *Reorder or Group layout based on hierarchical clustering*

Description

Reorder or Group layout based on hierarchical clustering

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,
  plot_dendrogram = TRUE,
  plot_cut_height = NULL,
  root = NULL,
  center = FALSE,
  type = "rectangle",
  size = NULL,
  free_guides = waiver(),
  free_spaces = waiver(),
  plot_data = waiver(),
  theme = waiver(),
  free_labs = waiver(),
  data = NULL,
  set_context = NULL,
  order = NULL,
  name = NULL
)
```

Arguments

`mapping`	Additional default list of aesthetic mappings to use for 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 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 distance 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, indicates whether we should reorder the dendrogram based on the means. Default: 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.

plot_dendrogram
: A boolean value indicates whether plot the dendrogram tree.

plot_cut_height
: A boolean value indicates whether plot the cut height.

root
: A length one string or numeric indicates the root branch.

center
: A boolean value. if TRUE, nodes are plotted centered with respect to the leaves in the branch. Otherwise (default), plot them in the middle of all direct child nodes.

type
: A string indicates the plot type, "rectangle" or "triangle".

size
: Plot size, can be an unit object.

free_guides
: Override the guides argument specified in the layout for a plot. Options include:

- waiver(): inherits behavior from the layout.
- NULL: no guide legends will be collected for the plot.
- A string containing one or more of "t", "l", "b", and "r" indicates which side of guide legends should be collected for the plot..

free_spaces
: A string with one or more of "t", "l", "b", and "r" indicating which border spaces should be removed. Defaults to waiver(), which inherits from the parent layout. If no parent, the default is NULL, meaning no spaces are removed.

plot_data
: A function to transform plot data before rendering. Defaults to waiver(), which inherits from the parent layout. If no parent layout, the default is NULL, meaning the data won't be modified.

Used to modify the data after layout has been created, which should be a data frame, but before the data is handled of to the ggplot2 for rendering. Use this hook if the you needs change the default data for all geoms.

`theme` Default plot theme: One of:

- `waiver()`: will inherit from the parent layout.
- `NULL`: Use the default theme.
- `theme()`: will be added with the parent layout theme.

`Note:` The axis title and labels parallel to the layout axis will always be removed by default. For vertical stack layouts, this refers to the `x-axis`, and for horizontal stack layouts, this refers to the `y-axis`. If you want to display the axis title or labels, you should manually add theme() elements for the parallel axis title or labels.

`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 `waiver()`, which inherits from the parent layout. If no parent layout, no axis titles will be aligned. If `NULL`, all axis titles will be aligned.

`data` A matrix, data frame, or a simple vector. If an atomic vector is provided, it will be converted into a one-column matrix. When `data` = `NULL`, the internal `layout` data will be used by default. Additionally, `data` can be a function (including purrr-like lambdas), which will be applied to the `layout` data.

It is important to note that we consider the `rows` as the observations. It means the `NROW(data)` must return the same number with the specific `layout` axis (meaning the x-axis for vertical stack layout, or y-axis for horizontal stack layout).

- `heatmap_layout()`: for column annotation, the `layout` data will be transposed before using (If data is a `function`, it will be applied with the transposed matrix). This is necessary because column annotation uses heatmap columns as observations, but we need rows.
- `stack_layout()`: the `layout` data will be used as it is since we place all plots along a single axis.

`set_context` A single boolean value indicates whether to set the active context to current plot. If `TRUE`, all subsequent ggplot elements will be added into this plot.

`order` An single integer for the plot area order.

`name` A string of the plot name. Used to switch the active context in `hmanno()` or `stack_active()`.

Value

A new `Align` object.

ggplot2 specification

`align_dendro` initializes a ggplot `data` and `mapping`.

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

The default ggplot data is the `node` coordinates, in addition, a geom_segment layer with a data of the tree segments `edge` coordinates will be added.

`node` and tree segments `edge` coordinates contains following columns:

- `index`: the original index in the tree for the current node

- label: node label text
- x and y: x-axis and y-axis coordinates for current node or the start node of the current edge.
- xend and yend: the x-axis and y-axis coordinates of the terminal node for current edge.
- branch: which branch current node or edge is. You can use this column to color different groups.
- panel: which panel current node is, if we split the plot into panel using facet_grid, this column will show which panel current node or edge is from. Note: some nodes may fall outside panel (between two panel), so there are possible NA values in this column.
- .panel: Similar with panel column, but always give the correct branch for usage of the ggplot facet.
- panel1 and panel2: The panel1 and panel2 variables have the same functionality as panel, but they are specifically for the edge data and correspond to both nodes of each edge.
- leaf: A logical value indicates whether current node is a leaf.

See Also

- dendrogram_data()
- hclust2()

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno("top") +
    align_dendro()
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno("top") +
    align_dendro(k = 3L)
```

align_gg *Create ggplot object with a customized data*

Description

ggalign is just an alias of align_gg.

Usage

```
align_gg(
  mapping = aes(),
  size = NULL,
  free_guides = waiver(),
  free_spaces = waiver(),
  plot_data = waiver(),
  theme = waiver(),
  free_labs = waiver(),
  data = NULL,
  limits = TRUE,
  facet = TRUE,
  set_context = TRUE,
  order = NULL,
  name = NULL
)

ggalign(
  mapping = aes(),
  size = NULL,
  free_guides = waiver(),
  free_spaces = waiver(),
  plot_data = waiver(),
  theme = waiver(),
  free_labs = waiver(),
  data = NULL,
  limits = TRUE,
  facet = TRUE,
  set_context = TRUE,
  order = NULL,
  name = NULL
)
```

Arguments

`mapping` Additional default list of aesthetic mappings to use for plot.

`size` Plot size, can be an unit object.

`free_guides` Override the `guides` argument specified in the layout for a plot. Options include:

- `waiver()`: inherits behavior from the layout.
- `NULL`: no guide legends will be collected for the plot.
- A string containing one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicates which side of guide legends should be collected for the plot..

`free_spaces` A string with one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicating which border spaces should be removed. Defaults to `waiver()`, which inherits from the parent layout. If no parent, the default is `NULL`, meaning no spaces are removed.

`plot_data` A function to transform plot data before rendering. Defaults to `waiver()`, which inherits from the parent layout. If no parent layout, the default is `NULL`, meaning the data won't be modified.

Used to modify the data after layout has been created, which should be a data frame, but before the data is handled of to the ggplot2 for rendering. Use this hook if the you needs change the default data for all `geoms`.

`theme` Default plot theme: One of:

- `waiver()`: will inherit from the parent layout.
- `NULL`: Use the default theme.
- `theme()`: will be added with the parent layout theme.

Note: The axis title and labels parallel to the layout axis will always be removed by default. For vertical stack layouts, this refers to the x-axis, and for horizontal stack layouts, this refers to the y-axis. If you want to display the axis title or labels, you should manually add theme() elements for the parallel axis title or labels.

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 waiver(), which inherits from the parent layout. If no parent layout, no axis titles will be aligned. If NULL, all axis titles will be aligned.

data A matrix, data frame, or a simple vector. If an atomic vector is provided, it will be converted into a one-column matrix. When data = NULL, the internal layout data will be used by default. Additionally, data can be a function (including purrr-like lambdas), which will be applied to the layout data.

It is important to note that we consider the rows as the observations. It means the NROW(data) must return the same number with the specific layout axis (meaning the x-axis for vertical stack layout, or y-axis for horizontal stack layout).

- heatmap_layout(): for column annotation, the layout data will be transposed before using (If data is a function, it will be applied with the transposed matrix). This is necessary because column annotation uses heatmap columns as observations, but we need rows.
- stack_layout(): the layout data will be used as it is since we place all plots along a single axis.

limits A boolean value indicates whether to set the layout limtis for the plot.

facet A boolean value indicates whether to set the layout facet for the plot. If this is FALSE, limits will always be FALSE too.

set_context A single boolean value indicates whether to set the active context to current plot. If TRUE, all subsequent ggplot elements will be added into this plot.

order An single integer for the plot area order.

name A string of the plot name. Used to switch the active context in hmanno() or stack_active().

Value

A AlignGG object.

ggplot2 specification

align_gg initializes a ggplot data and mapping.

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

For ggplot usage, matrix (including a simple vector) data is converted into a long-format data frame. The data in the underlying ggplot object will contain following columns:

- .panel: the panel for current layout axis. It means x-axis for vertical stack layout, y-axis for horizontal stack layout.

- `.x` or `.y`: the `x` or `y` coordinates
- `.row_names` and `.row_index`: A factor of the row names and an integer of row index of the original matrix or data frame.
- `.column_names` and `.column_index`: the column names and column index of the original matrix (only applicable if `data` is a `matrix`).
- `value`: the actual matrix value (only applicable if `data` is a `matrix`).

In the case where the input data is already a data frame, three additional columns (`.row_names`, `.row_index`, and `.panel`) are added to the data frame.

If the data is inherit from `heatmap_layout()`, an additional column will be added.

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

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno("top") +
    ggalign() +
    geom_point(aes(y = value))
```

`align_group` *Group and align layout based on categorical or factor levels.*

Description

Group and align layout based on categorical or factor levels.

Usage

```
align_group(group, set_context = FALSE, name = NULL)
```

Arguments

`group` A character define the groups, this will split the axis into different panel.

`set_context` A single boolean value indicates whether to set the active context to current plot. If `TRUE`, all subsequent ggplot elements will be added into this plot.

`name` A string of the plot name. Used to switch the active context in `hmanno()` or `stack_active()`.

Value

A new `Align` object.

Examples

```
small_mat <- matrix(rnorm(81), nrow = 9)
ggheatmap(small_mat) +
    hmanno("top") +
    align_group(sample(letters[1:4], ncol(small_mat), replace = TRUE))
```

`align_kmeans` *Split layout by k-means clustering groups.*

Description

Split layout by k-means clustering groups.

Usage

```
align_kmeans(centers, ..., data = NULL, set_context = FALSE, name = NULL)
```

Arguments

`centers` either the number of clusters, say k, or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in `x` is chosen as the initial centres.

`...` Arguments passed on to `stats::kmeans`

`iter.max` the maximum number of iterations allowed.

`nstart` if `centers` is a number, how many random sets should be chosen?

`algorithm` character: may be abbreviated. Note that `"Lloyd"` and `"Forgy"` are alternative names for one algorithm.

`trace` 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 matrix, data frame, or a simple vector. If an atomic vector is provided, it will be converted into a one-column matrix. When `data = NULL`, the internal `layout` data will be used by default. Additionally, `data` can be a function (including purrr-like lambdas), which will be applied to the `layout` data.

It is important to note that we consider the `rows` as the observations. It means the `NROW(data)` must return the same number with the specific `layout` axis (meaning the x-axis for vertical stack layout, or y-axis for horizontal stack layout).

- `heatmap_layout()`: for column annotation, the `layout` data will be transposed before using (If data is a `function`, it will be applied with the transposed matrix). This is necessary because column annotation uses heatmap columns as observations, but we need rows.
- `stack_layout()`: the `layout` data will be used as it is since we place all plots along a single axis.

`set_context` A single boolean value indicates whether to set the active context to current plot. If `TRUE`, all subsequent ggplot elements will be added into this plot.

`name` A string of the plot name. Used to switch the active context in `hmanno()` or `stack_active()`.

Value

A new `Align` object.

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno("t") +
    align_kmeans(3L)
```

`align_order` *Order layout observations based on weights*

Description

Order layout observations based on weights

Usage

```
align_order(
  wts = rowMeans,
  ...,
  reverse = FALSE,
  strict = TRUE,
  data = NULL,
  set_context = FALSE,
  name = NULL
)
```

Arguments

`wts`	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 `I()`.
`...`	<dyn-dots> Additional arguments passed to function provided in `wts` 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 a simple vector. If an atomic vector is provided, it will be converted into a one-column matrix. When `data = NULL`, the internal `layout` data will be used by default. Additionally, `data` can be a function (including purrr-like lambdas), which will be applied to the `layout` data. It is important to note that we consider the `rows` as the observations. It means the `NROW(data)` must return the same number with the specific `layout` axis (meaning the x-axis for vertical stack layout, or y-axis for horizontal stack layout).

- `heatmap_layout()`: for column annotation, the `layout` data will be transposed before using (If data is a `function`, it will be applied with the transposed matrix). This is necessary because column annotation uses heatmap columns as observations, but we need rows.
- `stack_layout()`: the `layout` data will be used as it is since we place all plots along a single axis.

`set_context` A single boolean value indicates whether to set the active context to current plot. If `TRUE`, all subsequent ggplot elements will be added into this plot.

`name` A string of the plot name. Used to switch the active context in `hmanno()` or `stack_active()`.

Value

A new `Align` object.

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno(”l”) +
    align_order(I(”rowMeans”))
```

`align_panel` *Create ggplot object with layout panel data*

Description

This is similar with `ggalign()` function, but it will always use the layout panel data. `ggpanel` is just an alias of `align_panel`.

Usage

```
align_panel(
  mapping = aes(),
  size = NULL,
  free_guides = waiver(),
  free_spaces = waiver(),
  plot_data = waiver(),
  theme = waiver(),
  free_labs = waiver(),
  limits = TRUE,
  facet = TRUE,
  set_context = TRUE,
  order = NULL,
  name = NULL
)

ggpanel(
  mapping = aes(),
  size = NULL,
  free_guides = waiver(),
  free_spaces = waiver(),
  plot_data = waiver(),
  theme = waiver(),
  free_labs = waiver(),
  limits = TRUE,
  facet = TRUE,
  set_context = TRUE,
  order = NULL,
  name = NULL
)
```

Arguments

`mapping`
Additional default list of aesthetic mappings to use for plot.

`size`
Plot size, can be an unit object.

`free_guides`
Override the `guides` argument specified in the layout for a plot. Options include:

- `waiver()`: inherits behavior from the layout.
- `NULL`: no guide legends will be collected for the plot.
- A string containing one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicates which side of guide legends should be collected for the plot..

`free_spaces`
A string with one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicating which border spaces should be removed. Defaults to `waiver()`, which inherits from the parent layout. If no parent, the default is `NULL`, meaning no spaces are removed.

`plot_data`
A function to transform plot data before rendering. Defaults to `waiver()`, which inherits from the parent layout. If no parent layout, the default is `NULL`, meaning the data won't be modified.

Used to modify the data after layout has been created, which should be a data frame, but before the data is handled of to the ggplot2 for rendering. Use this hook if the you needs change the default data for all `geoms`.

`theme`
Default plot theme: One of:

- `waiver()`: will inherit from the parent layout.
- `NULL`: Use the default theme.
- `theme()`: will be added with the parent layout theme.

`Note:` The axis title and labels parallel to the layout axis will always be removed by default. For vertical stack layouts, this refers to the `x-axis`, and for horizontal stack layouts, this refers to the `y-axis`. If you want to display the axis title or labels, you should manually add theme() elements for the parallel axis title or labels.

`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 `waiver()`, which inherits from the parent layout. If no parent layout, no axis titles will be aligned. If `NULL`, all axis titles will be aligned.

`limits` A boolean value indicates whether to set the layout limtis for the plot.

`facet` A boolean value indicates whether to set the layout facet for the plot. If this is `FALSE`, `limits` will always be `FALSE` too.

`set_context` A single boolean value indicates whether to set the active context to current plot. If `TRUE`, all subsequent ggplot elements will be added into this plot.

`order` An single integer for the plot area order.

`name` A string of the plot name. Used to switch the active context in `hmanno()` or `stack_active()`.

Value

A `AlignPanel` object.

ggplot2 specification

`align_panel` initializes a ggplot `data` and `mapping`.

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

The data in the underlying `ggplot` object contains following columns:

- `.panel`: the panel for current layout axis. It means `x-axis` for vertical stack layout, `y-axis` for horizontal stack layout.
- `.index`: the index of the original layout data.
- `.x` or `.y`: the `x` or `y` coordinates

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno(”top”) +
    ggalign() +
    geom_point(aes(y = value))
```

`align_plots` *Arrange multiple plots into a grid*

Description

Arrange multiple plots into a grid

Usage

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

Arguments

`...` <dyn-dots> A list of plots, ususally the ggplot object. Use `NULL` to indicate an empty spacer.

`ncol, nrow` The dimensions of the grid to create - if both are `NULL` it will use the same logic as `facet_wrap()` to set the dimensions

`byrow` If `FALSE` the plots will be filled in in column-major order.

`widths, heights` The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid. The special value of `NA` will behave as `1null` unit unless a fixed aspect plot is inserted in which case it will allow the dimension to expand or contract to match the aspect ratio of the content.

`design` Specification of the location of areas in the layout. Can either be specified as a text string or by concatenating calls to `area()` together.

`guides` A string containing one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicates which side of guide legends should be collected. If `NULL`, no guide legends will be collected. If `waiver()`, it will inherit from the parent layout

`theme` A `theme()` used to render the `guides`, `title`, `subtitle`, `caption`, `margins`, `patch.title`, `panel.border`, and `background`. If `NULL` (default), will inherit from the parent `layout`.

Value

An `alignpatches` object.

See Also

- `layout_design()`
- `layout_title()`
- `layout_annotation()`

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
design <- "#BB
           AA#"
align_plots(B = p1, A = p2, design = design)

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

`align_reorder` *Reorders layout observations based on specific statistics.*

Description

Reorders layout observations based on specific statistics.

Usage

```
align_reorder(
  stat,
  ...,
  reverse = FALSE,
  strict = TRUE,
  data = NULL,
  set_context = FALSE,
  name = NULL
)
```

Arguments

`stat` A summary function which accepts a data and returns the statistic, which we'll call `order2()` 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 a simple vector. If an atomic vector is provided, it will be converted into a one-column matrix. When `data = NULL`, the internal `layout` data will be used by default. Additionally, `data` can be a function (including purrr-like lambdas), which will be applied to the `layout` data.

It is important to note that we consider the `rows` as the observations. It means the `NROW(data)` must return the same number with the specific `layout` axis (meaning the x-axis for vertical stack layout, or y-axis for horizontal stack layout).

- `heatmap_layout()`: for column annotation, the `layout` data will be transposed before using (If data is a `function`, it will be applied with the transposed matrix). This is necessary because column annotation uses heatmap columns as observations, but we need rows.
- `stack_layout()`: the `layout` data will be used as it is since we place all plots along a single axis.

`set_context` A single boolean value indicates whether to set the active context to current plot. If `TRUE`, all subsequent ggplot elements will be added into this plot.

`name` A string of the plot name. Used to switch the active context in `hmanno()` or `stack_active()`.

Details

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

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

Value

A new `Align` object.

See Also

`order2()`

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno(”l”) +
    align_reorder(hclust2)
```

`area` *Define the plotting areas in* `align_plots`

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 int the grid

Details

The grid that the areas are specified in reference to enumerate rows from top to bottom, and coloumns 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 `align_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)
```

dendrogram_data *Dengrogram x and y coordinates*

Description

Dengrogram x and y coordinates

Usage

```
dendrogram_data(
  tree,
  priority = "right",
  center = FALSE,
  type = "rectangle",
  leaf_pos = NULL,
  leaf_braches = NULL,
  reorder_branches = TRUE,
  branch_gap = NULL,
  root = NULL
)
```

Arguments

`tree`	A hclust or a dendrogram object.
`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 the leaves in the branch. Otherwise (default), plot them in the middle of all 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 tree.

leaf_braches — Branches of the leaf node. Must be the same length of the number of observations in tree. 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.

Value

A list of 2 data.frame. One for node coordinates, another for edge coordinates. node and tree segments edge coordinates contains following columns:

- index: the original index in the tree for the current node
- label: node label text
- x and y: x-axis and y-axis coordinates for current node or the start node of the current edge.
- xend and yend: the x-axis and y-axis coordinates of the terminal node for current edge.
- branch: which branch current node or edge is. You can use this column to color different groups.
- panel: which panel current node is, if we split the plot into panel using facet_grid, this column will show which panel current node or edge is from. Note: some nodes may fall outside panel (between two panels), so there are possible NA values in this column.
- .panel: Similar with panel column, but always give the correct branch for usage of the ggplot facet.
- panel1 and panel2: The panel1 and panel2 variables have the same functionality as panel, but they are specifically for the edge data and correspond to both nodes of each edge.
- leaf: A logical value indicates whether current node is a leaf.

Examples

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

free_align *Free from alignment*

Description

align_plots will try to align plot panels, and every elements of the plot, following functions romove 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`: If we want to compose plots without alignment of the panel borders (but still align the panels themselves), we can wrap the plot with `free_border`.
- `free_lab`: If we want to compose plots without alignment of the axis title, we can wrap the plot with `free_lab`.
- `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"`, and `"r"` 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 `grid::viewport`

- `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 `gpar`. 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 `as.path`.
- `mask` One of `"none"` (or `FALSE`) or `"inherit"` (or `TRUE`) or a grob (or a gTree) or the result of call to `as.mask`. 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 `free_align` class.

- `free_border`: A modified version of `plot` with a `free_border` class.
- `free_guide`: A modified version of `plot` with a `free_guide` class.
- `free_lab`: A modified version of `plot` with a `free_lab` class.
- `free_space`: A modified version of `plot` with a `free_space` class.
- `free_vp`: A modified version of `plot` with a `free_vp` class.

Examples

```
# directly copied from `patchwork`
# Sometimes you have a plot that defies good composition alginment, 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 be 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, "l"), p2, p2)
```

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_stat *Get the statistics from the layout*

Description

Get the statistics from the layout

Usage

```
ggalign_stat(x, ...)

## S3 method for class 'HeatmapLayout'
ggalign_stat(x, position, ...)

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

Arguments

`x` A `heatmap_layout()` or `stack_layout()` 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

`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 distance 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 *Arrange plots in a Heatmap*

Description

ggheatmap is an alias of heatmap_layout.

Usage

```
heatmap_layout(
  data,
  mapping = aes(),
  ...,
  filling = TRUE,
  .width = NA,
  .height = NA,
  guides = waiver(),
  theme = NULL,
  set_context = TRUE,
  order = NULL,
  name = NULL
)

ggheatmap(
  data,
  mapping = aes(),
  ...,
  filling = TRUE,
  .width = NA,
  .height = NA,
  guides = waiver(),
  theme = NULL,
  set_context = TRUE,
  order = NULL,
  name = NULL
)
```

Arguments

`data` A numeric or character vector, a data frame, and any other data which can be converted into a matrix. Simple vector will be converted into a one column matrix.

`mapping` Default list of aesthetic mappings to use for plot. In addition, we will always add mapping `aes(.data$.x, .data$.y)`.

`...` Additional arguments passed to geom_tile. Only used when `filling = TRUE`.

`filling` A boolean value indicating whether to fill the heatmap. If you wish to customize the filling style, set this to `FALSE`.

By default, the classic heatmap colour scheme `scale_fill_gradient2(low = "blue", high = "red")` is utilized for continuous values. You can use the option `"ggalign.heatmap_continuous_fill"` or `"ggalign.heatmap_discrete_fill"` to modify the default heatmap body fill color scale. See `scale_fill_continuous()` or `scale_fill_discrete()` for option setting details.

`.width, .height` Heatmap body width/height, can be a unit object.

`guides` A string containing one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicates which side of guide legends should be collected. If `NULL`, no guide legends will be collected. If `waiver()`, it will inherit from the parent layout

`theme` A `theme()` used to render the `guides`, `title`, `subtitle`, `caption`, `margins`, `patch.title`, `panel.border`, and `background`. If `NULL` (default), will inherit from the parent `layout`.

`set_context` A single boolean value indicates whether to set the active context to current plot. If `TRUE`, all subsequent ggplot elements will be added into this plot.

`order` An single integer for the plot area order.

`name` A string of the plot name. Used to switch the active context in `hmanno()` or `stack_active()`.

Value

A `HeatmapLayout` object.

ggplot2 specification

The data input in `ggheatmap` will be converted into the long formated data frame when drawing. The default mapping will use `aes(.data$.x, .data$.y)`, you can use `mapping` argument to control it. The data in the underlying `ggplot` object contains following columns:

- `.xpanel` and `.ypanel`: the column and row panel
- `.x` and `.y`: the x and y coordinates
- `.row_names` and `.column_names`: A factor 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))
```

hmanno *Determine the active context of heatmap layout*

Description

Determine the active context of heatmap layout

Usage

```
hmanno(
  position = NULL,
  size = NULL,
  guides = NA,
  free_guides = NA,
  free_spaces = NA,
  plot_data = NA,
  theme = NA,
  free_labs = NA,
  what = waiver(),
  width = NULL,
  height = NULL
)
```

Arguments

`position` Which heatmap annotation should get activated? Possible values are follows:

- A string of `"top"`, `"left"`, `"bottom"`, or `"right"`.
- `NULL`: means set the active context into the `heatmap` itself.

`size` An unit object to set the total size of the heatmap annotation. This will only be used if `position` is a string.

- If `position` is `"top"` or `"bottom"`, `size` set the total height of the annotation.
- If `position` is `"left"` or `"right"`, `size` set the total width of the annotation.

`guides` A string containing one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicates which side of guide legends should be collected. If `NULL`, no guide legends will be collected. If `waiver()`, it will inherit from the parent layout

- If `position` is `NULL`, this applies to the heamtap layout.
- If `position` is a string, this applies to the heatmap annotation stack layout.

`free_guides` Override the `guides` collection behavior specified in the heatmap layout. Options include:

- `waiver()`: inherits behavior from the layout.
- `NULL`: no guide legends will be collected for the plot.
- A string containing one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicates which side of guide legends should be collected for the plot.

`guides` argument controls the global guide collection behavior for all plots in the layout, while the `free_guides` argument overrides this for a single plot in the layout.

`free_spaces` A string with one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicating which border spaces should be removed. Defaults to `waiver()`, which inherits from the parent layout. If no parent, the default is `NULL`, meaning no spaces are removed.

`plot_data` A function to transform plot data before rendering. Defaults to `waiver()`, which inherits from the parent layout. If no parent layout, the default is `NULL`, meaning the data won't be modified.

Used to modify the data after layout has been created, which should be a data frame, but before the data is handled of to the ggplot2 for rendering. Use this hook if the you needs change the default data for all `geoms`.

`theme` Default theme for the plot in the layout. One of:

- `waiver()`: will inherit from the parent layout.
- `NULL`: Use the default theme.
- `theme()`: will be added with the parent layout theme.

`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 `waiver()`, which inherits from the parent layout. If no parent layout, no axis titles will be aligned. If `NULL`, all axis titles will be aligned.

`what` What should get activated for the anntoation stack? Only used when position is a string. Options include::

- A single number or string of the plot elements in the stack layout.
- `NULL`: remove any active context.

`width, height` Heatmap body width/height, can be a unit object.

Value

A `heatmap_active` object which can be added into heatmap_layout.

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno("top") +
    align_dendro()
```

`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 `patch()`.

`...` Additional arguments passed to the `patch()` 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 frontmost. 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

A `patch_inset` object, which can be added in ggplot.

See Also

- patch.grob / patch.gList
- patch.ggplot
- patch.patch_ggplot
- patch.patchwork
- patch.patch
- patch.trellis
- patch.formula / patch.function
- patch.recordedplot
- patch.Heatmap
- patch.HeatmapList
- patch.HeatmapAnnotation
- patch.pheatmap

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_ggheatmap *Reports whether* x *is a* heatmap_layout() *object*

Description

Reports whether x is a heatmap_layout() object

Usage

```
is_ggheatmap(x)
```

Arguments

x	An object to test

Value

A boolean value

Examples

```
is_ggheatmap(ggheatmap(1:10))
```

is_ggstack *Reports whether* x *is a* StackLayout *object*

Description

Reports whether x is a StackLayout object

Usage

```
is_ggstack(x)
```

Arguments

x	An object to test

Value

A boolean value

Examples

```
is_ggstack(ggstack(1:10))
```

---

layout-add *Add components to* Layout

Description

Add components to Layout

Usage

```
## S4 method for signature 'Layout,ANY'
e1 + e2
```

Arguments

e1	A heatmap_layout() or stack_layout() object.
e2	An object to be added to the plot, including gg elements or align object.

Value

A modified Layout object.

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno("t") +
    ggalign() +
    geom_point(aes(y = value))
```

layout-operator *Layout operator*

Description

Layout operator

Usage

```
## S4 method for signature 'Layout,ANY'
e1 & e2

## S4 method for signature 'Layout,ANY'
e1 - e2
```

Arguments

e1	A heatmap_layout() or stack_layout() object.
e2	An object to be added to the plot.

Details

In order to reduce code repetition ggalign provides two operators for adding ggplot elements (geoms, themes, facets, etc.) to multiple/all plots in heatmap_layout() or stack_layout() object.

Like patchwork, & add the element to all plots in the plot. If the element is a theme, this will also modify the layout theme.

Unlike patchwork, the - operator adds ggplot2 elements (geoms, themes, facets, etc.) rather than a ggplot plot. The key difference between & and - is in how they behave in heatmap_layout(). The - operator only applies the element to the current active context in heatmap_layout(). Using - might seem unintuitive if you think of the operator as "subtract", the underlying reason is that - is the only operator in the same precedence group as +.

Value

A modified Layout object.

Examples

```
mat <- matrix(rnorm(81), nrow = 9)
ggheatmap(mat) +
    hmanno("top") +
    align_dendro() &
    theme(panel.border = element_rect(
        colour = "red", fill = NA, linewidth = unit(2, "mm")
    ))
ggheatmap(mat) +
    hmanno("top") +
    align_dendro() -
    theme(panel.border = element_rect(
        colour = "red", fill = NA, linewidth = unit(2, "mm")
    ))
```

layout_annotation *Modify components of the layout*

Description

- modify the theme of the layout

Usage

```
layout_annotation(theme = waiver(), ...)
```

Arguments

theme	A theme() used to render the guides, title, subtitle, caption, margins, patch.title, panel.border, and background. If NULL (default), will inherit from the parent layout.
...	These dots are for future extensions and must be empty.

Details

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

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_annotation(
        theme = 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 = waiver(),
  nrow = waiver(),
  byrow = waiver(),
  widths = waiver(),
  heights = waiver(),
  design = waiver(),
  guides = NA
)
```

Arguments

ncol, nrow	The dimensions of the grid to create - if both are NULL it will use the same logic as facet_wrap() to set the dimensions
byrow	If FALSE the plots will be filled in in column-major order.
widths, heights	The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid. The special value of NA will behave as 1null unit unless a fixed aspect plot is inserted in which case it will allow the dimension to expand or contract to match the aspect ratio of the content.
design	Specification of the location of areas in the layout. Can either be specified as a text string or by concatenating calls to area() together.
guides	A string containing one or more of "t", "l", "b", and "r" indicates which side of guide legends should be collected. If NULL, no guide legends will be collected. If waiver(), it will inherit from the parent layout

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_title` *Annotate the whole layout*

Description

Annotate the whole layout

Usage

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

Arguments

`title`	The text for the title.
`subtitle`	The text for the subtitle for the plot which will be displayed below the title.
`caption`	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")
```

order2 *Ordering Permutation*

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)
```

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)))
```

patch.alignpatches *Convert Object into a Grob*

Description

The patch() function is used by wrap() and inset() to convert objects into a grob.

Usage

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

Arguments

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

Value

A grob object.

See Also

alignpatches

`patch.formula` *Convert Object into a Grob*

Description

The `patch()` function is used by `wrap()` and `inset()` to convert objects into a grob.

Usage

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

## S3 method for class '`function`'
patch(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 `grid.echo()` 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.

See Also

plot

patch.ggplot *Convert Object into a Grob*

Description

The patch() function is used by wrap() and inset() to convert objects into a grob.

Usage

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

Arguments

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

Value

A grob object.

See Also

ggplot

patch.grob *Convert Object into a Grob*

Description

The patch() function is used by wrap() and inset() to convert objects into a grob.

Usage

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

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

Arguments

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

Value

A grob object.

patch.Heatmap *Convert Object into a Grob*

Description

The `patch()` function is used by `wrap()` and `inset()` to convert objects into a grob.

Usage

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

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

## S3 method for class 'HeatmapAnnotation'
patch(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.

See Also

- Heatmap
- HeatmapAnnotation

patch.patch *Convert Object into a Grob*

Description

The `patch()` function is used by `wrap()` and `inset()` to convert objects into a grob.

Usage

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

Arguments

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

Value

A grob object.

See Also

patch

`patch.patchwork` *Convert Object into a Grob*

Description

The `patch()` function is used by `wrap()` and `inset()` to convert objects into a grob.

Usage

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

Arguments

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

Value

A grob object.

See Also

patchwork

patch.patch_ggplot *Convert Object into a Grob*

Description

The patch() function is used by wrap() and inset() to convert objects into a grob.

Usage

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

Arguments

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

Value

A grob object.

See Also

- patch_titles
- inset
- wrap

patch.pheatmap *Convert Object into a Grob*

Description

The patch() function is used by wrap() and inset() to convert objects into a grob.

Usage

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

Arguments

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

Value

A grob object.

See Also

pheatmap

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

Description

The `patch()` function is used by `wrap()` and `inset()` to convert objects into a grob.

Usage

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

Arguments

`x`	An object to be converted into a grob.
`...`	Not used currently.
`device`	A function that opens a graphics device for `grid.echo()` 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.

See Also

recordPlot

patch.trellis *Convert Object into a Grob*

Description

The `patch()` function is used by `wrap()` and `inset()` to convert objects into a grob.

Usage

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

Arguments

`x` An object to be converted into a grob.

`...` Arguments passed on to `grid::grid.grabExpr`

- `warn` 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.
- `wrap` 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.
- `wrap.grobs` A logical indicating whether, if we are wrapping elements (`wrap=TRUE`), we should wrap grobs (or just wrap viewports).
- `width,height` Size of the device used for temporary rendering.

`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.

See Also

trellis

patch_titles *Add patch titles to plot borders*

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_titles(
  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

You can also use labs() to specify the titles (use arguments `"top"`, `"left"`, `"bottom"`, and `"right"`) for the top, left, bottom, and right borders of the plot.

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_titles(
        top = "I'm top patch title",
        left = "I'm left patch title",
        bottom = "I'm bottom patch title",
        right = "I'm right patch title"
    )
```

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()
```

stack_active *Determine the active context of stack layout*

Description

Determine the active context of stack layout

Usage

```
stack_active(
  guides = NA,
  free_spaces = NA,
  plot_data = NA,
  theme = NA,
  free_labs = NA,
  what = NULL,
  sizes = NULL
)
```

Arguments

`guides` A string containing one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicates which side of guide legends should be collected. If `NULL`, no guide legends will be collected. If `waiver()`, it will inherit from the parent layout

`free_spaces` A string with one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicating which border spaces should be removed. Defaults to `waiver()`, which inherits from the parent layout. If no parent, the default is `NULL`, meaning no spaces are removed.

`plot_data` A function to transform plot data before rendering. Defaults to `waiver()`, which inherits from the parent layout. If no parent layout, the default is `NULL`, meaning the data won't be modified.

Used to modify the data after layout has been created, which should be a data frame, but before the data is handled of to the ggplot2 for rendering. Use this hook if the you needs change the default data for all `geoms`.

`theme` Default theme for the plot in the layout. One of:

- `waiver()`: will inherit from the parent layout.
- `NULL`: Use the default theme.
- `theme()`: will be added with the parent layout theme.

`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 `waiver()`, which inherits from the parent layout. If no parent layout, no axis titles will be aligned. If `NULL`, all axis titles will be aligned.

`what` What should get activated for the stack layout? Options include::

- A single number or string of the plot elements in the stack layout.
- `NULL`: remove any active context, this is useful when the active context is a `heatmap_layout()` object, where any `align_*()` will be added into the heatmap. By removing the active context, we can add `align_*()` into the `stack_layout()`.

`sizes` A numeric or unit object of length `3` indicates the relative widths (`direction = "horizontal"`) / heights (`direction = "vertical"`).

Value

A `stack_active` object which can be added into StackLayout.

Examples

```
ggstack(matrix(1:9, nrow = 3L)) +
    ggheatmap() +
    # ggheamtap 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()
```

stack_layout *Put plots horizontally or vertically*

Description

ggstack is an alias of stack_layout.

Usage

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

ggstack(
  data,
  direction = NULL,
  sizes = NA,
  ...,
  guides = waiver(),
  theme = NULL
)
```

Arguments

data	A numeric or character vector, a data frame, or a matrix.
direction	A string of "horizontal" or "vertical", indicates the direction of the stack layout.
sizes	A numeric or unit object of length 3 indicates the relative widths (direction = "horizontal") / heights (direction = "vertical").
...	<dyn-dots> A list of plots, ususally the ggplot object. Use NULL to indicate an empty spacer.
guides	A string containing one or more of "t", "l", "b", and "r" indicates which side of guide legends should be collected. If NULL, no guide legends will be collected. If waiver(), it will inherit from the parent layout
theme	A theme() used to render the guides, title, subtitle, caption, margins, patch.title, panel.border, and background. If NULL (default), will inherit from the parent layout.

Value

A StackLayout object.

Examples

```
ggstack(matrix(rnorm(100L), nrow = 10L)) + align_dendro()
```

`theme_ggalign` *Complete theme for layout plots*

Description

Default theme for `heatmap_layout()` or `stack_layout()` object.

Usage

```
theme_ggalign(...)
```

Arguments

`...` Arguments passed on to `ggplot2::theme_classic`

`base_size` base font size, given in pts.

`base_family` base font family

`base_line_size` base size for line elements

`base_rect_size` base size for rect elements

Details

You can use the option `"ggalign.default_theme"` to change the default theme.

Value

A theme object.

Examples

```
old <- options(ggalign.default_theme = theme_bw())
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    hmanno("top") +
    align_dendro(k = 3L)
options(ggalign.default_theme = old)
```

`wrap` *Wrap Arbitrary Graphics for Alignment*

Description

The `wrap()` 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 `patch()` method.

Usage

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

Arguments

`plot` Any graphic that can be converted into a grob using `patch()`.

`...` Additional arguments passed to the `patch()` 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 frontmost. 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

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

See Also

- patch.grob / patch.gList
- patch.ggplot
- patch.patch_ggplot
- patch.patchwork
- patch.patch
- patch.trellis
- patch.formula / patch.function
- patch.recordedplot
- patch.Heatmap
- patch.HeatmapList
- patch.HeatmapAnnotation
- patch.pheatmap

Examples

```
library(grid)
wrap(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, wrap(
    ~ plot(mtcars$mpg, mtcars$disp),
    mar = c(0, 2, 0, 0), bg = NA
))
```

Index