ggalign: Bridging the Grammar of Graphics and Complex layout

Yun Peng

2024-11-24

Table of contents

Preface

Welcome to `ggalign` documents

1 Introduction

`ggalign` extends ggplot2 by providing advanced tools for aligning and organizing multiple plots, particularly those that automatically reorder observations, such as dendrogram. It offers fine control over layout adjustment and plot annotations, enabling you to create complex, publication-quality visualizations while still using the familiar grammar of ggplot2.

1.1 Installation

You can install `ggalign` from `CRAN` using:

```r
install.packages("ggalign")
```

Alternatively, install the development version from r-universe with:

```r
install.packages("ggalign",
    repos = c("https://yunuuuu.r-universe.dev", "https://cloud.r-project.org")
)
```

or from GitHub with:

```r
# install.packages("remotes")
remotes::install_github("Yunuuuu/ggalign")
```

1.2 General design

The core feature of `ggalign` lies in its integration of the grammar of graphics principles into advanced visualization, achieved through its object-oriented `Layout` system. Two primary `Layout` classes are available:

- the `StackLayout` class: Put plots horizontally or vertically.
- the `QuadLayout` class: Arranges plots in the four quadrants (top, left, bottom, right) around a main plot. This layout is ideal for designs that require supplementary plots or annotations surrounding a central figure.

Both `Layout` classes support the alignment of observations. Observations refer to data points or samples, allowing for consistent alignment of corresponding data across multiple plots when using the same axis values. Depending on whether you want to align observations across multiple plots within the layout, the following variants are available:

For `StackLayout`:

- `stack_align()`: align the observations along the stack.
- `stack_free()`: don't align the observations.

For `QuadLayout`:

- `quad_free`/`ggside`: Never align observations.
- `quad_alignh`: Align observations in the horizontal direction.
- `quad_alignv`: Align observations in the vertical direction.
- `quad_alignb`: Align observations in both horizontal and vertical directions.

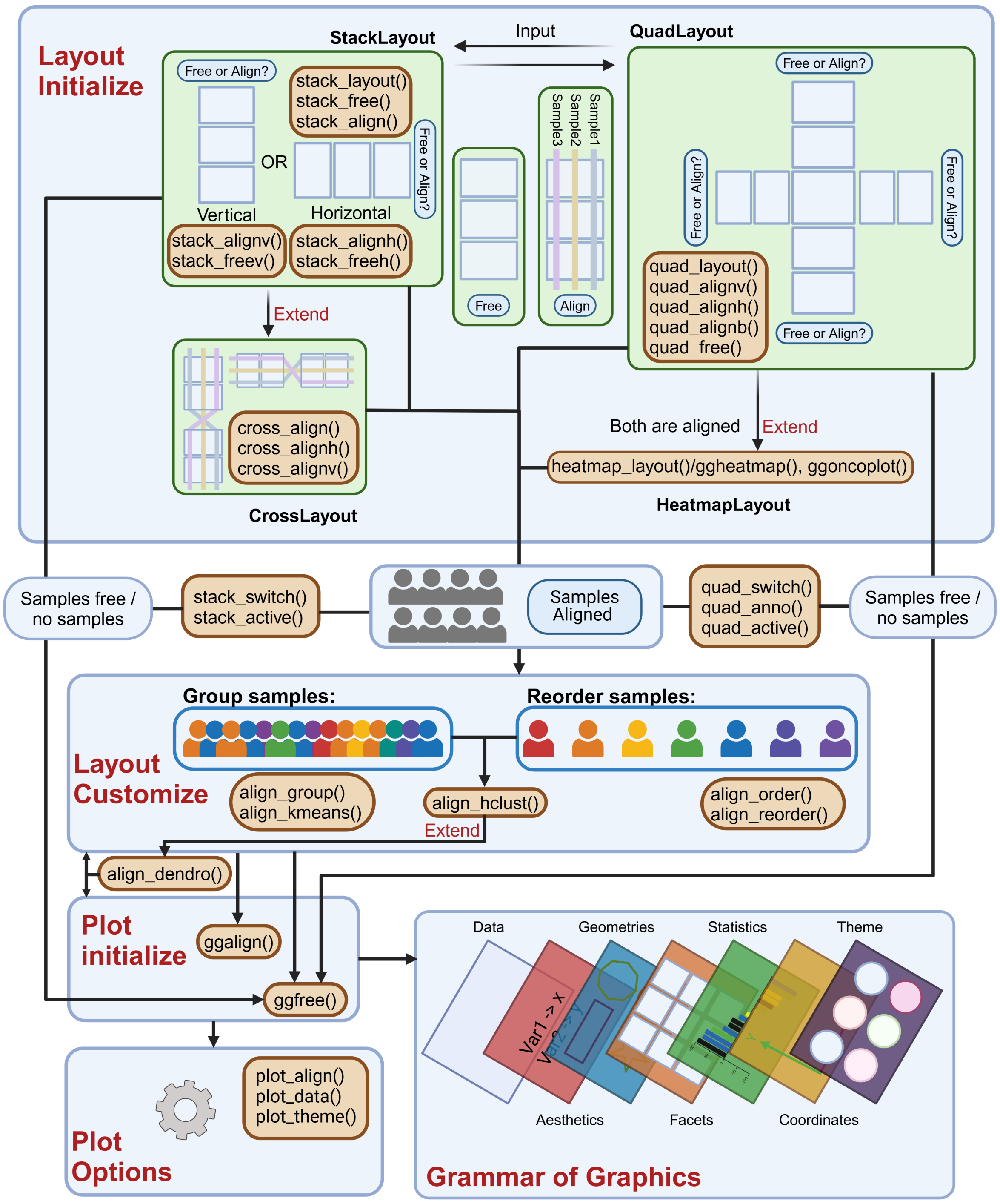


Figure 1.1: General design of ggalign

1.3 Getting Started

```r
library(ggalign)
```

```
Loading required package: ggplot2
```

The usage of `ggalign` is simple if you're familiar with `ggplot2` syntax, the typical workflow includes:

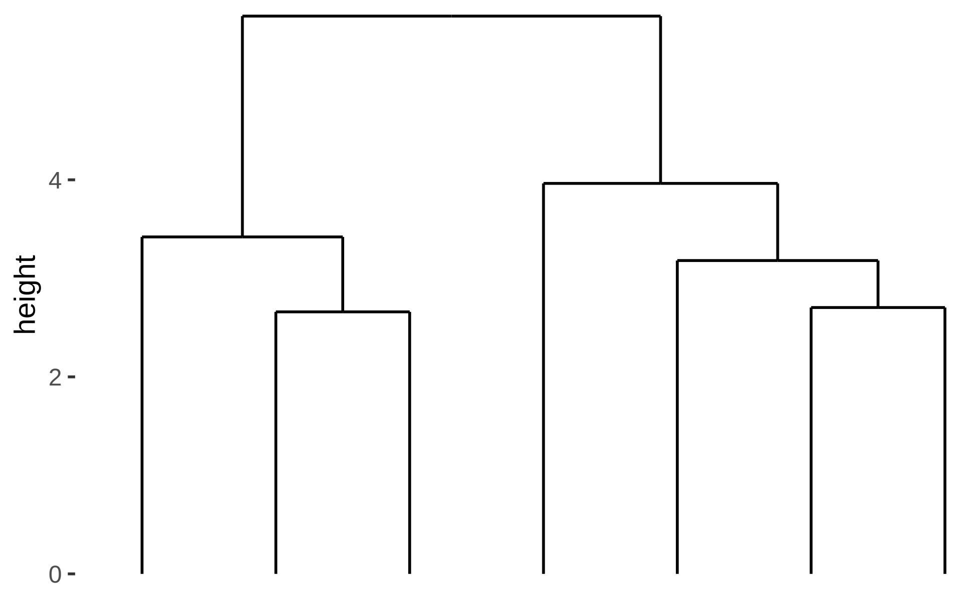
- Initialize the layout using `stack_layout()` (`ggstack()`/`cross_align()`) or `quad_layout()` (`ggside()`/`ggheatmap()`/`ggoncoplot()`).
- Customize the layout with:
  - `align_group()`: Group observations into panel with a group variable.
  - `align_kmeans()`: Group observations into panel by kmeans.
  - `align_order()`: Reorder layout observations based on statistical weights or by manually specifying the observation index.
  - `align_hclust()`/`align_dendro()`: Reorder or Group layout based on hierarchical clustering.
- Adding plots with `ggalign()` or `ggfree()`, and then layer additional ggplot2 elements such as geoms, stats, or scales.

```r
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)))
```

Every `*_layout()` function accepts default data, which will be inherited by all plots within the layout.

Here's a simple example:

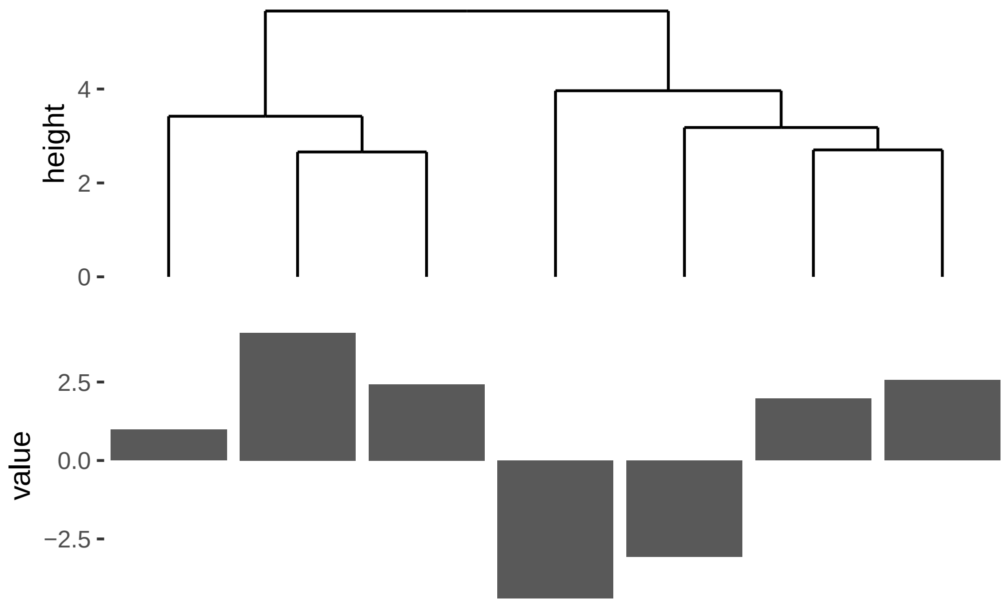
```r
stack_alignv(small_mat) +
    align_dendro() +
    theme(axis.text.y = element_text())
```



This produces a simple dendrogram. By default, `stack_alignv()` removes the axis text on the axis used for aligning observations. This is because it's often unclear which plot should display the axis text, as typically, we want it to appear in only one plot. However, you can easily use the `theme()` function to control where the axis text appears.

Internally, `align_dendro()` will reorder the observations based on the dendrogram, and other plots in the layout will follow this ordering.

```r
stack_alignv(small_mat) +
    align_dendro() +
    ggalign(data = rowSums) +
    geom_bar(aes(.names, value), stat = "identity") +
    theme(axis.text.y = element_text())
```



In this example:

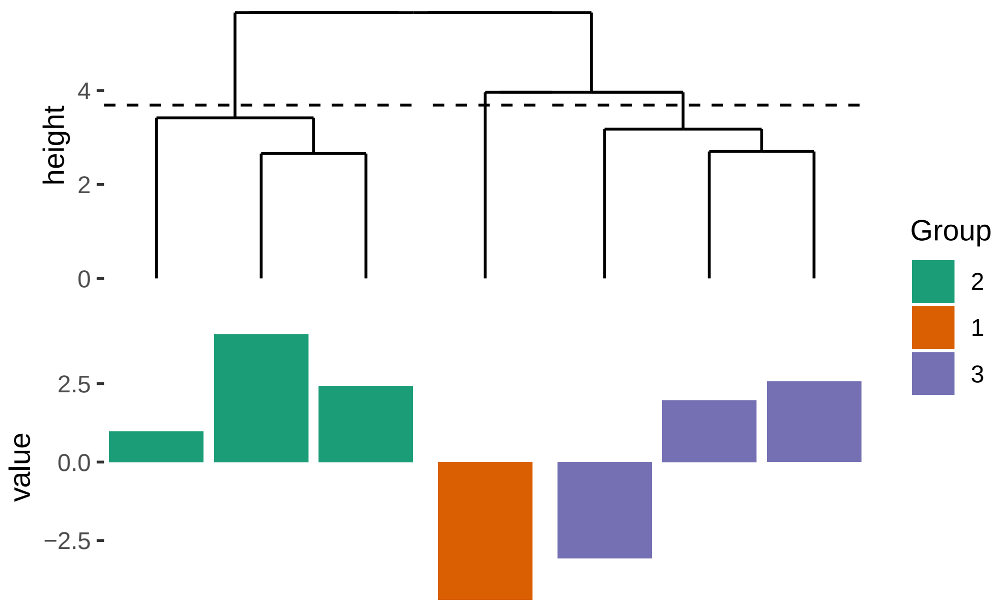
1. We initialize a vertical stack (`stack_alignv(small_mat)`).
2. Reorder the observations based on hierarchical clustering and add a dendrogram tree (`align_dendro()`).
3. Create a new plot in the layout, and use data based on the sum of the layout data (`ggalign(data = rowSums)`).
4. Add ggplot2 components to the plot.

The data in the underlying `ggplot` object of `ggalign` contains at least following columns (more details will be introduced in a later chapter):

- `.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).
- `.x` or `.y`: the `x` or `y` coordinates
- `.names` and `.index`: A factor of the names (only applicable when names exists) and an integer of index of the original data.
- `value`: the actual value (only applicable if `data` is a `matrix` or atomic vector).

`align_dendro()` can also split the observations into groups.

```r
stack_alignv(small_mat) +
    align_dendro(k = 3) +
    ggalign(data = rowSums) +
    geom_bar(aes(.names, value, fill = .panel), stat = "identity") +
    scale_fill_brewer(palette = "Dark2", name = "Group") +
    theme(axis.text.y = element_text())
```



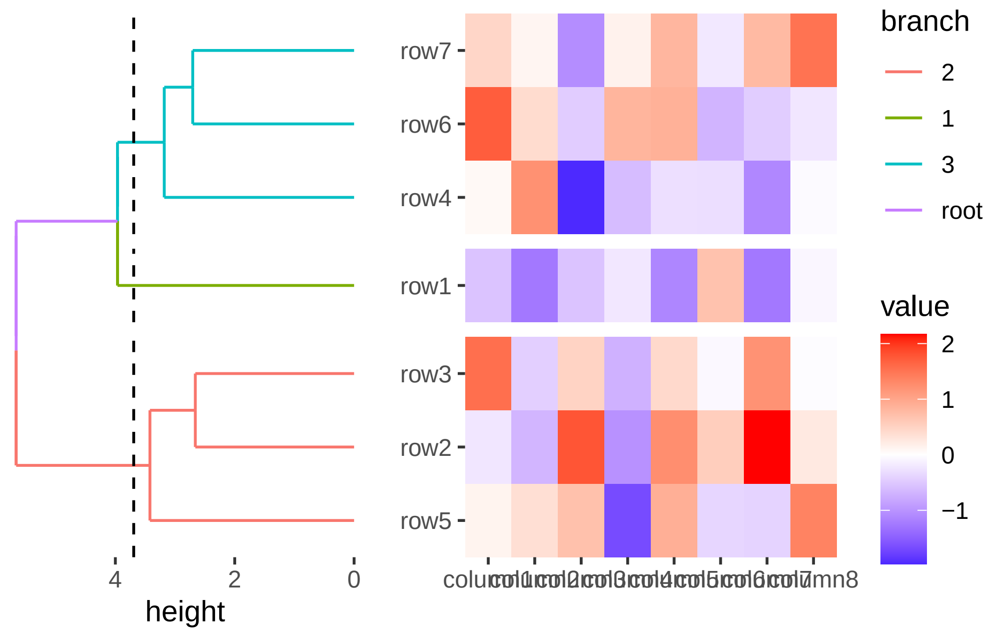
In this example:

1. We initialize a vertical stack (`stack_alignv(small_mat)`).
2. Reorder and group the observations based on hierarchical clustering, and add a dendrogram tree (`align_dendro(k = 3)`).
3. Create a new plot in the layout (`ggalign(data = rowSums)`), which will use data based on the sums of the layout data.
4. Add ggplot2 components to the plot.

One common visualization associated with the dendrogram is the heatmap. You can use `ggheatmap()` to initialize a heatmap layout. When grouping the observations using `align_dendro(k = 3)`, a special column named branch is added, which you can use to color the dendrogram tree.

```
ggheatmap(small_mat) +
    anno_left() +
    align_dendro(aes(color = branch), k = 3)+
    scale_fill_brewer(palette = "Dark2")
```

```
> heatmap built with `geom_tile()`
```



In this example:

1. We initialize a heatmap layout (`ggheatmap(small_mat)`).
2. we initialize an annotation in the left side of the heatmap body, and set it as the active context, in this way, all following addition will be directed to the left annotation. (`anno_left()`)
3. Reorder and group the observations based on hierarchical clustering, and add a dendrogram tree, coloring the tree by `branch` (`align_dendro(k = 3)`).

`ggheatmap()` will automatically add the names for the heatmap, so you don't need to manually adjust axis text visibility using `theme(axis.text.x = element_text())`/`theme(axis.text.y = element_text())`.

2 stack layout

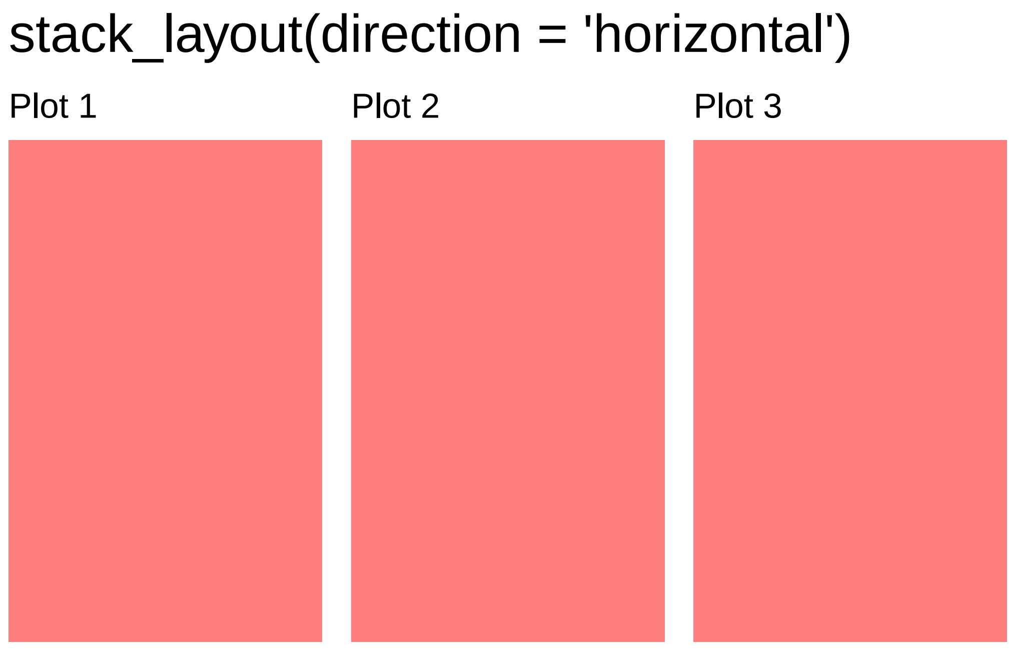
`stack_layout()` arranges plots either horizontally or vertically, and we can also use the alias `ggstack()`. Based on whether we want to align the observations, there are two types of stack layouts:

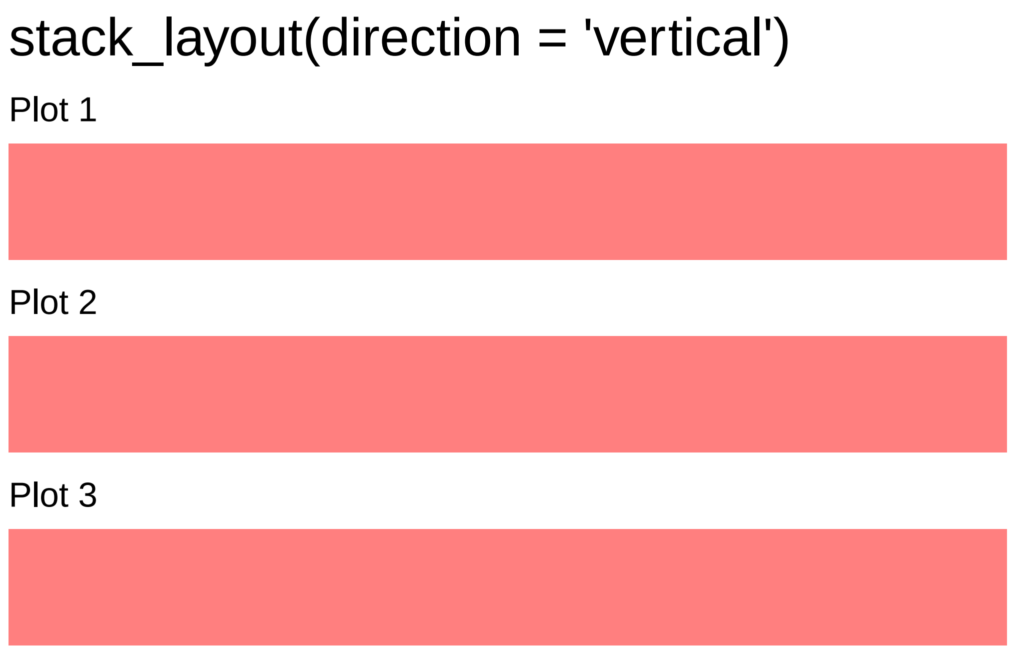
- `stack_align()`: align the observations along the stack.
- `stack_free()`: don't align the observations.

Several aliases are available for convenience:

- `stack_alignv`: Aligns the stack vertically (special case of `stack_align()`).
- `stack_alignh`: Aligns the stack horizontally (special case of `stack_align()`).
- `stack_freev`: A vertical version of `stack_free()`.
- `stack_freeh`: A horizontal version of `stack_free()`.

```
Loading required package: ggplot2
```





2.1 Input data

When aligning observations, we typically use a matrix, as it is easy to melt the matrix into a long formated data frame. In addition, we need a matrix to fit the observations concept, since we need transpose the data (rows to columns, columns to rows) when used in `quad_layout()`/`ggheatmap()`, which can align observations in both direction simutaneuously.

- For `stack_free()`, a data frame is required, and the input will be automatically converted using `fortify_data_frame()` if needed.
- For `stack_align()`, a matrix is required, and the input will be automatically converted using `fortify_matrix()` if needed.

By default, `fortify_data_frame()` will invoke the `ggplot2::fortify()` function for conversion. Note, for matrices, it will convert the matrix into a long-formatted data frame.

```r
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)))
```

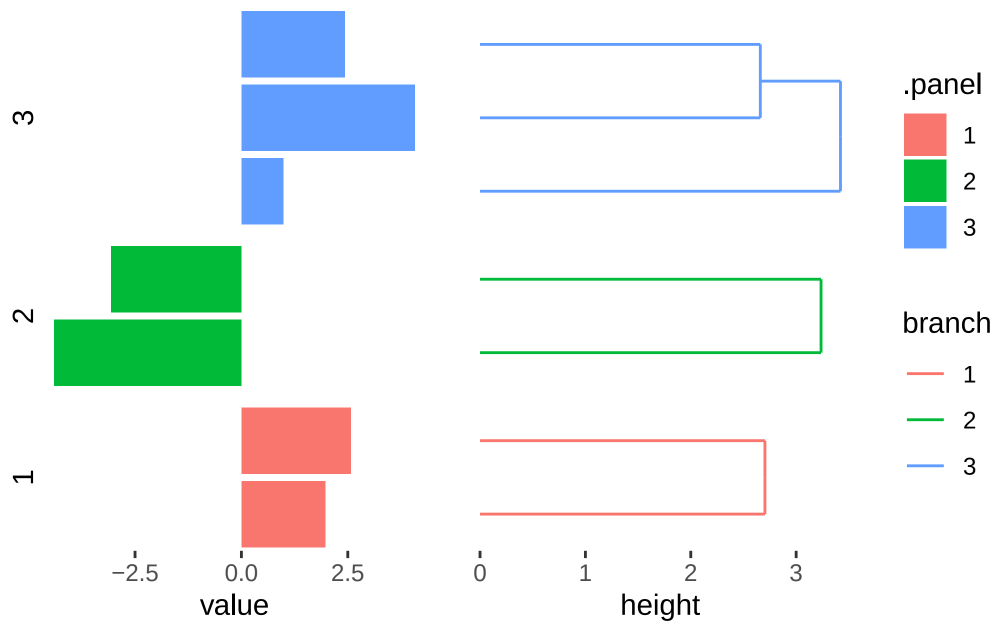
By default, `stack_align()`/`stack_free()` will set up the layout, but no plot will be drawn until you add a plot element:

```
stack_alignh(small_mat)
# the same for `stack_free()`
```

2.2 plot initialize

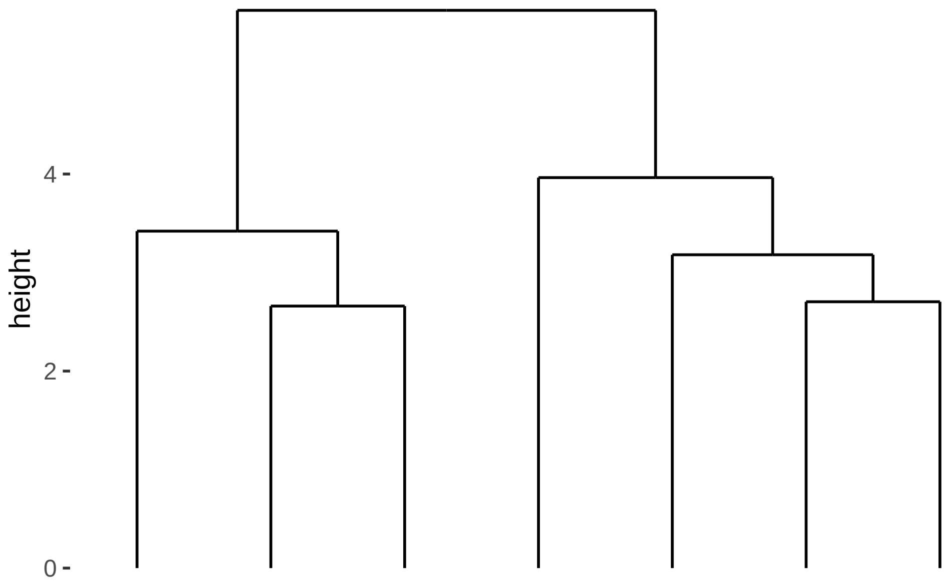
For `stack_align()`, plots can be added regardless of whether they need to align observations.

```
stack_alignh(small_mat) +
    align_kmeans(centers = 3L) +
    ggalign(data = rowSums) +
    geom_bar(aes(value, fill = .panel), orientation = "y", stat = "identity") +
    facet_grid(switch = "y") +
    theme(strip.text = element_text()) +
    align_dendro(aes(color = branch))
```



We can stack the plots vertically with `stack_alignv()`:

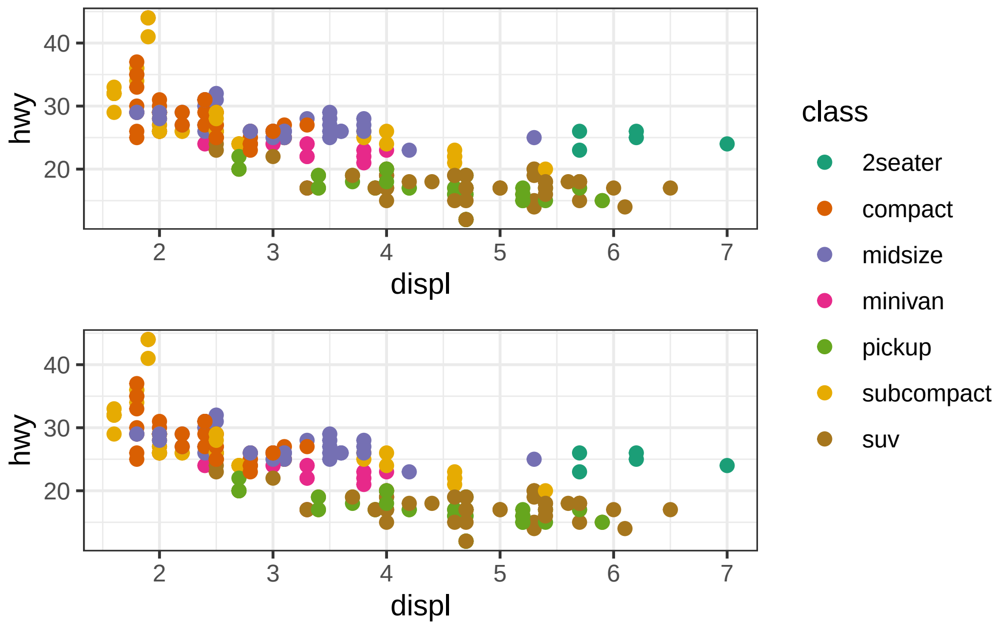
```r
stack_alignv(small_mat) + align_dendro()
```



Note that vertical stack take the `x-axis` as the observations, but horizontal stack take the `y-axis` as the observations.

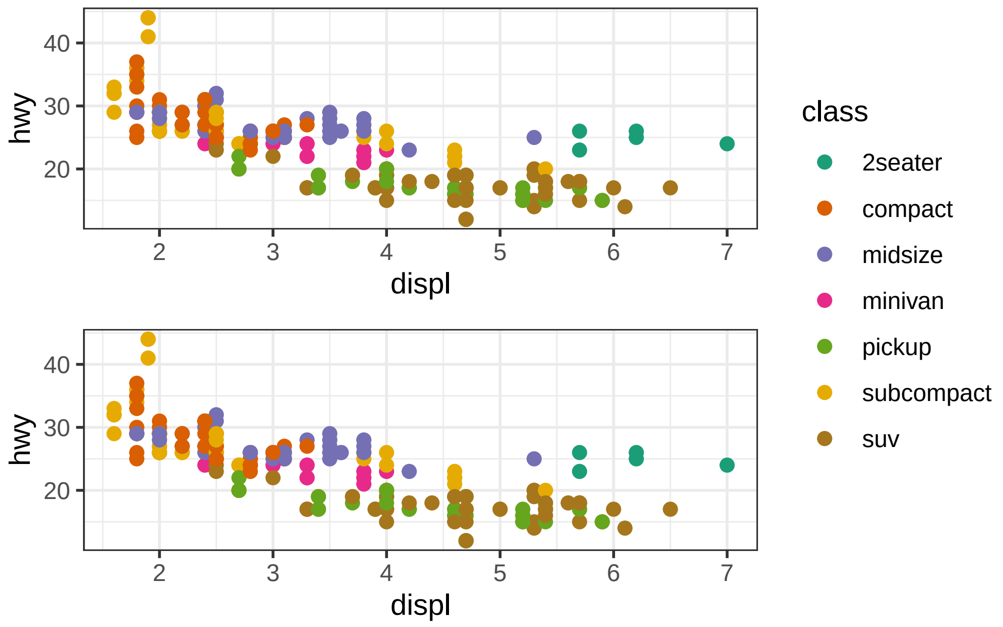
`stack_align()` can add plot without alignment of observations.

```
stack_alignv() +
    ggfree(aes(displ, hwy, colour = class), data = mpg) +
    geom_point(size = 2) +
    ggfree(aes(displ, hwy, colour = class), data = mpg) +
    geom_point(size = 2) &
    scale_color_brewer(palette = "Dark2") &
    theme_bw()
```



For `stack_free()`, only plots that are free from alignment (`ggfree()`) can be added along the axis.

```
stack_freev(mpg) +
    ggfree(aes(displ, hwy, colour = class)) +
    geom_point(size = 2) +
    ggfree(aes(displ, hwy, colour = class)) +
    geom_point(size = 2) &
    scale_color_brewer(palette = "Dark2") &
    theme_bw()
```



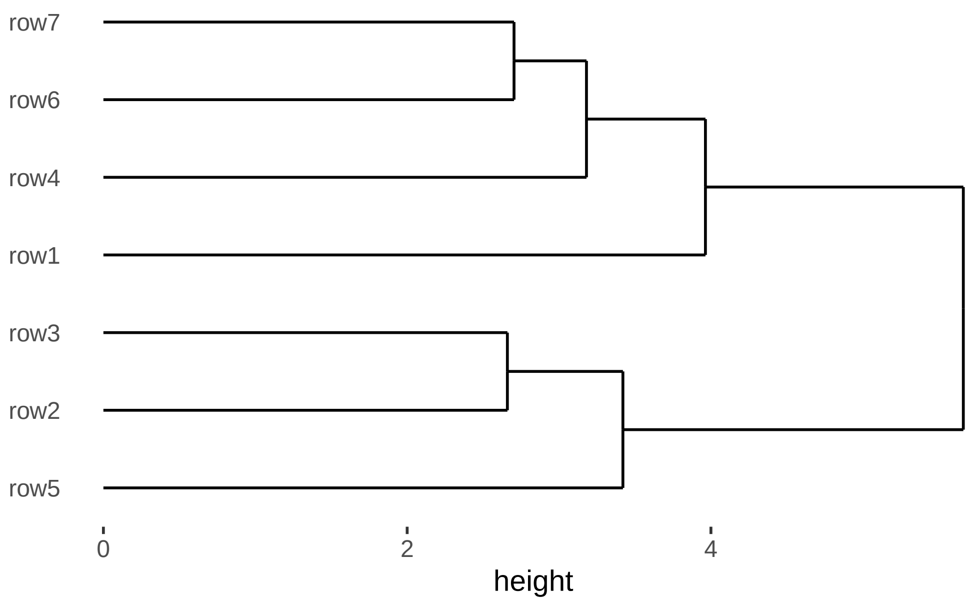
2.3 Layout Customization

When we use `stack_align()`, it aligns the observations across multiple plots along the specified direction:

- For `stack_alignh()`: Alignment occurs along the horizontal direction (y-axis).
- For `stack_alignv()`: Alignment occurs along the vertical direction (x-axis).

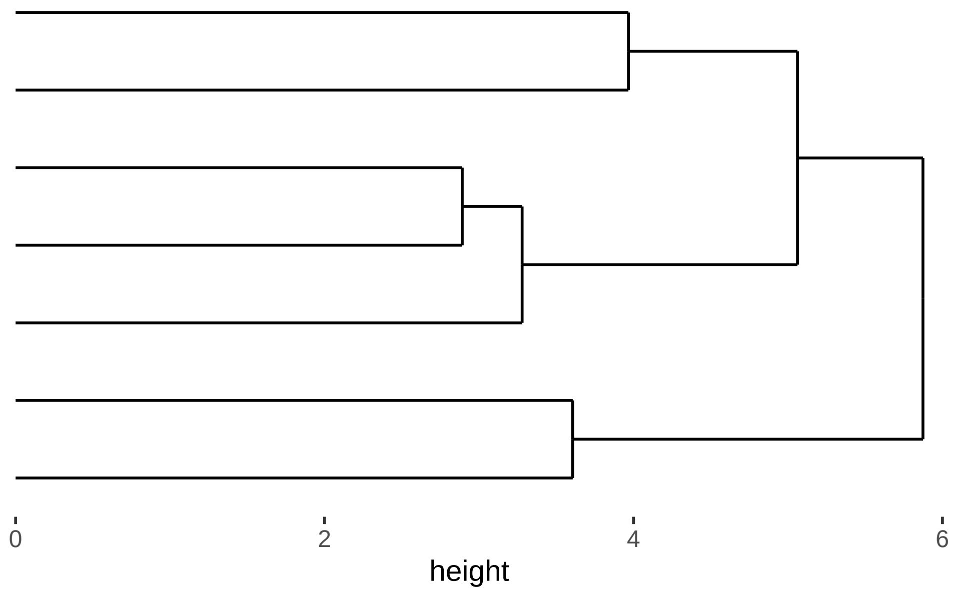
One useful situation for `stack_align()` is with dendrograms, which inherently reorder the observations. The `align_dendro()` function is specifically designed for this purpose. It can reorder the observations, split them into groups, and can add a plot for visualization.

```r
stack_alignh(small_mat) +
    align_dendro() +
    theme(axis.text.y = element_text())
```



We don't provide data to `align_dendro()`, by default, it'll inherit data from the layout. But you can always provide another data, but note that this package use the concept of `"number of observations"` in the vctrs package or `NROW()` function. When aligning the observations, you must ensure the number of observations is equal.

```r
set.seed(123)
stack_alignh(small_mat) +
    align_dendro(data =  matrix(rnorm(70), nrow = 7)) +
    theme(axis.text.y = element_text())
```


0
2
4
6
height

3

`r if (knitr::is_html_output()) ' # References {-} '`