

Package ‘funkyheatmap’

September 23, 2023

Title Generating Funky Heatmaps for Data Frames

Description Allows generating heatmap-like visualisations for benchmark data frames. Funky heatmaps can be fine-tuned by providing annotations of the columns and rows, which allows assigning multiple palettes or geometries or grouping rows and columns together in categories.

Saelens et al. (2019) <[doi:10.1038/s41587-019-0071-9](https://doi.org/10.1038/s41587-019-0071-9)>.

Version 0.5.0

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports assertthat, cli, cowplot, dplyr, ggforce, ggplot2 (>= 3.4.0), grDevices, jsonlite, patchwork, purrr, RColorBrewer, Rdpack, stringr, tibble, tidyr

Suggests kableExtra, knitr, rmarkdown, testthat (>= 3.0.0), magick

VignetteBuilder knitr

RdMacros Rdpack

Depends R (>= 2.10)

LazyData true

URL <https://funkyheatmap.github.io/funkyheatmap/>,
<https://github.com/funkyheatmap/funkyheatmap>

BugReports <https://github.com/funkyheatmap/funkyheatmap/issues>

Config/testthat/edition 3

NeedsCompilation no

Author Robrecht Cannoodt [aut, cre] (<<https://orcid.org/0000-0003-3641-729X>>, rcannood),

Wouter Saelens [aut] (<<https://orcid.org/0000-0002-7114-6248>>, zouter)

Maintainer Robrecht Cannoodt <rcannood@gmail.com>

Repository CRAN

Date/Publication 2023-09-23 06:10:02 UTC

R topics documented:

dynbenchmark_data	2
funky_heatmap	3
geom_rounded_rect	6
position_arguments	8
scale_minmax	9
scib_summary	10
verify_column_groups	10
verify_column_info	12
verify_data	14
verify_legends	14
verify_palettes	17
verify_row_groups	19
verify_row_info	20

Index	22
--------------	-----------

dynbenchmark_data	<i>The results data frame from dynbenchmark.</i>
-------------------	--

Description

This data was generated by running the data-raw/dynbenchmark_data.R script. It is used in the vignette named "vignette("dynbenchmark")" to regenerate the results figures in Saelens et al. 2019.

Usage

```
dynbenchmark_data
```

Format

An object of class `list` of length 6.

References

Saelens W, Cannoodt R, Todorov H, Saeys Y (2019). "A comparison of single-cell trajectory inference methods." *Nature Biotechnology*. doi:10.1038/s4158701900719.

`funky_heatmap`*Generate a funky heatmaps for benchmarks*

Description

Allows generating heatmap-like visualisations for benchmark data frames. Funky heatmaps can be fine-tuned by providing annotations of the columns and rows, which allows assigning multiple palettes or geometries or grouping rows and columns together in categories.

Usage

```
funky_heatmap(  
  data,  
  column_info = NULL,  
  row_info = NULL,  
  column_groups = NULL,  
  row_groups = NULL,  
  palettes = NULL,  
  legends = NULL,  
  position_args = position_arguments(),  
  scale_column = TRUE,  
  add_abc = TRUE,  
  col_annot_offset,  
  col_annot_angle,  
  expand  
)
```

Arguments

- | | |
|--------------------------|---|
| <code>data</code> | A data frame with items by row and features in the columns. Must contain one column named "id". |
| <code>column_info</code> | A data frame describing which columns in data to plot. This data frame should contain the following columns: <ul style="list-style-type: none">• <code>id</code> (character, required): A column name in data to plot. Determines the size of the resulting geoms, and also the color unless <code>color</code> is specified.• <code>id_color</code> (character): A column name in data to use for the color of the resulting geoms. If NA, the <code>id</code> column will be used.• <code>id_size</code> (character): A column name in data to use for the size of the resulting geoms. If NA, the <code>id</code> column will be used.• <code>name</code> (character): A label for the column. If NA or "", no label will be plotted. If this column is missing, <code>id</code> will be used to generate the name column.• <code>geom</code> (character): The geom of the column. Must be one of: "funkyrect", "circle", "rect", "bar", "pie", "text" or "image". For "text", the corresponding column in data must be a character. For "pie", the column must be a list of named numeric vectors. For all other geoms, the column must be a numeric. |

- `group` (character): The grouping id of each column, must match with `column_groups$group`. If this column is missing or all values are NA, columns are assumed not to be grouped.
 - `palette` (character): Which palette to colour the geom by. Each value should have a matching value in `palettes$palette`.
 - `width`: Custom width for this column (default: 1).
 - `overlay`: Whether to overlay this column over the previous column. If so, the width of that column will be inherited.
 - `legend`: Whether or not to add a legend for this column.
 - `hjust`: Horizontal alignment of the bar, must be between [0,1] (only for `geom = "bar"`).
 - `vjust`: Vertical alignment of the label, must be between [0,1] (only for `geom = "text"`).
 - `size`: Size of the label, must be a numeric value (only for `geom = "text"`).
 - `label`: Which column to use as a label (only for `geom = "text"`).
 - `directory`: Which directory to use to find the images (only for `geom = "image"`).
 - `extension`: The extension of the images (only for `geom = "image"`).
 - `draw_outline`: Whether or not to draw bounding guides (only for `geom == "bar"`). Default: TRUE.
 - `options` (list or json): Any of the options above. Any values in this column will be spread across the other columns. This is useful for not having to provide a data frame with 1000s of columns. This column can be a json string.
- `row_info` A data frame describing the rows of data. This data should contain two columns:
- `id` (character): Corresponds to the column `data$id`.
 - `group` (character): The group of the row. If all are NA, the rows will not be split up into groups.
- `column_groups` A data frame describing of how to group the columns in `column_info`. Can consist of the following columns:
- `group` (character): The corresponding group in `column_info$group`.
 - `palette` (character, optional): The palette used to colour the column group backgrounds.
 - `level1` (character): The label at the highest level.
 - `level2` (character, optional): The label at the middle level.
 - `level3` (character, optional): The label at the lowest level (not recommended).
- `row_groups` A data frame describing of how to group the rows in `row_info`. Can consist of the following columns:
- `group` (character): The corresponding group in `row_info$group`.
 - `level1` (character): The label at the highest level.
 - `level2` (character, optional): The label at the middle level.
 - `level3` (character, optional): The label at the lowest level (not recommended).

palettes	<p>A named list of palettes. Each entry in <code>column_info\$palette</code> should have an entry in this object. If an entry is missing, the type of the column will be inferred (categorical or numerical) and one of the default palettes will be applied. Alternatively, the name of one of the standard palette names can be used:</p> <ul style="list-style-type: none"> • numerical: "Greys", "Blues", "Reds", "YlOrBr", "Greens" • categorical: "Set3", "Set1", "Set2", "Dark2"
legends	<p>A list of legends to add to the plot. Each entry in <code>column_info\$legend</code> should have a corresponding entry in this object. Each entry should be a list with the following names:</p> <ul style="list-style-type: none"> • <code>palette</code> (character): The palette to use for the legend. Must be a value in <code>palettes</code>. • <code>geom</code> (character): The geom of the legend. Must be one of: "funkyrect", "circle", "rect", "bar", "pie", "text", "image". • <code>title</code> (character, optional): The title of the legend. Defaults to the palette name. • <code>enabled</code> (logical, optional): Whether or not to add the legend. Defaults to TRUE. • <code>labels</code> (character, optional): The labels to use for the legend. The defaults depend on the selected geom. • <code>size</code> (numeric, optional): The size of the listed geoms. The defaults depend on the selected geom. • <code>color</code> (character, optional): The color of the listed geoms. The defaults depend on the selected geom. • <code>values</code> (optional): Used as values for the text and image geoms. • <code>label_width</code> (numeric, optional): The width of the labels (only when geom is text or pie). Defaults to 1 for text and 2 for images. • <code>value_width</code> (numeric, optional): The width of the values (only for geom = "text"). Defaults to 2. • <code>label_hjust</code> (numeric, optional): The horizontal alignment of the labels (only when geom is circle, rect or funkyrect). Defaults to 0.5.
position_args	Sets parameters that affect positioning within a plot, such as row and column dimensions, annotation details, and the expansion directions of the plot. See <code>position_arguments()</code> for more information.
scale_column	Whether or not to apply min-max scaling to each numerical column.
add_abc	Whether or not to add subfigure labels to the different columns groups.
col_annot_offset	DEPRECATED: use <code>position_args = position_arguments(col_annot_offset = ...)</code> instead.
col_annot_angle	DEPRECATED: use <code>position_args = position_arguments(col_annot_angle = ...)</code> instead.
expand	DEPRECATED: use <code>position_args = position_arguments(expand_* = ...)</code> instead.

Value

A ggplot. `.$width` and `.$height` are suggested dimensions for storing the plot with `ggsave()`.

Examples

```
library(tibble, warn.conflicts = FALSE)

data("mtcars")

data <- rownames_to_column(mtcars, "id")

funky_heatmap(data)
```

geom_rounded_rect *Rounded rectangles*

Description

Does what `ggplot2::geom_rect()` does, only *curvier*. Use the `radius` aesthetic to change the corner radius.

Usage

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

Aesthetics

`geom_tile()` understands the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- colour
- fill
- group
- height
- linetype
- linewidth
- width

Note that `geom_raster()` ignores colour.

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

Examples

```
library(ggplot2)

df <- data.frame(
  x = rep(c(2, 5, 7, 9, 12), 2),
  y = rep(c(1, 2), each = 5),
  z = factor(rep(1:5, each = 2)),
  w = rep(diff(c(0, 4, 6, 8, 10, 14)), 2)
)
```

```
ggplot(df) +  
  geom_rounded_rect(  
    aes(  
      xmin = x - w / 2,  
      xmax = x + w / 2,  
      ymin = y,  
      ymax = y + 1,  
      radius = .5,  
      fill = z  
    ),  
    colour = "white"  
  )  
)
```

position_arguments *Defines parameters for positioning in a plot.*

Description

This function sets parameters that affect positioning within a plot, such as row and column dimensions, annotation details, and the expansion directions of the plot.

Usage

```
position_arguments(  
  row_height = 1,  
  row_space = 0.1,  
  row_bigspace = 1.2,  
  col_width = 1,  
  col_space = 0.1,  
  col_bigspace = 0.5,  
  col_annot_offset = 3,  
  col_annot_angle = 30,  
  expand_xmin = 0,  
  expand_xmax = 2,  
  expand_ymin = 0,  
  expand_ymax = 0  
)
```

Arguments

row_height	The height of the rows.
row_space	The space between rows.
row_bigspace	The large space between row groups.
col_width	The width of the columns.
col_space	The space between columns.
col_bigspace	The large space between column groups.

col_annot_offset	How much the column annotation will be offset by.
col_annot_angle	The angle of the column annotation labels.
expand_xmin	The minimum expansion of the plot in the x direction.
expand_xmax	The maximum expansion of the plot in the x direction.
expand_ymin	The minimum expansion of the plot in the y direction.
expand_ymax	The maximum expansion of the plot in the y direction.

Value

A list of plot positioning parameters.

Examples

```
position_arguments(row_height = 1.2, col_width = 1.5, expand_xmax = 3)
```

scale_minmax	<i>Scale a vector to the range [0, 1]</i>
--------------	---

Description

Scale a vector to the range [0, 1]

Usage

```
scale_minmax(x)
```

Arguments

x	A numeric vector
---	------------------

Value

A numeric vector scaled to the range [0, 1]

Examples

```
scale_minmax(c(1, 2, 3))
```

scib_summary	<i>Summary results from the scIB project</i>
--------------	--

Description

This dataset was generated by running the `data-raw/scib_summary.R` script. This script downloads data for the RNA tasks from the [scIB reproducibility repository](#) and summarises it similarly (but not exactly like) what was done in the original paper to give a final ranking of the top performing methods. It is used in the scIB vignette (`vignette("scIB")`) to reproduce the overall summary figure in Luecken et al. 2021.

Usage

```
scib_summary
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 20 rows and 27 columns.

References

Luecken MD, Büttner M, Chaichoompu K, Danese A, Interlandi M, Mueller MF, Strobl DC, Zappia L, Dugas M, Colomé-Tatché M, Theis FJ (2021). “Benchmarking atlas-level data integration in single-cell genomics.” *Nature methods*. doi:10.1038/s41592021013368.

verify_column_groups	<i>Verify the integrity of the column groups object</i>
----------------------	---

Description

Verify the integrity of the column groups object

Usage

```
verify_column_groups(column_groups, column_info)
```

Arguments

`column_groups` A data frame describing of how to group the columns in `column_info`. Can consist of the following columns:

- `group` (character): The corresponding group in `column_info$group`.
- `palette` (character, optional): The palette used to colour the column group backgrounds.
- `level1` (character): The label at the highest level.
- `level2` (character, optional): The label at the middle level.

- level3 (character, optional): The label at the lowest level (not recommended).
- column_info A data frame describing which columns in data to plot. This data frame should contain the following columns:
- id (character, required): A column name in data to plot. Determines the size of the resulting geoms, and also the color unless color is specified.
 - id_color (character): A column name in data to use for the color of the resulting geoms. If NA, the id column will be used.
 - id_size (character): A column name in data to use for the size of the resulting geoms. If NA, the id column will be used.
 - name (character): A label for the column. If NA or "", no label will be plotted. If this column is missing, id will be used to generate the name column.
 - geom (character): The geom of the column. Must be one of: "funkyrect", "circle", "rect", "bar", "pie", "text" or "image". For "text", the corresponding column in data must be a character. For "pie", the column must be a list of named numeric vectors. For all other geoms, the column must be a numeric.
 - group (character): The grouping id of each column, must match with column_groups\$group. If this column is missing or all values are NA, columns are assumed not to be grouped.
 - palette (character): Which palette to colour the geom by. Each value should have a matching value in palettes\$palette.
 - width: Custom width for this column (default: 1).
 - overlay: Whether to overlay this column over the previous column. If so, the width of that column will be inherited.
 - legend: Whether or not to add a legend for this column.
 - hjust: Horizontal alignment of the bar, must be between [0,1] (only for geom = "bar").
 - vjust: Vertical alignment of the label, must be between [0,1] (only for geom = "text").
 - size: Size of the label, must be a numeric value (only for geom = "text").
 - label: Which column to use as a label (only for geom = "text").
 - directory: Which directory to use to find the images (only for geom = "image").
 - extension: The extension of the images (only for geom = "image").
 - draw_outline: Whether or not to draw bounding guides (only for geom == "bar"). Default: TRUE.
 - options (list or json): Any of the options above. Any values in this column will be spread across the other columns. This is useful for not having to provide a data frame with 1000s of columns. This column can be a json string.

Value

The column groups object with all expected columns.

Examples

```

library(tibble)
column_groups <- tribble(
  ~group, ~level1,
  "foo", "Foo",
  "bar", "Bar"
)
column_info <- tribble(
  ~id, ~geom, ~group,
  "name", "text", NA_character_,
  "foo1", "funkyrect", "foo",
  "foo2", "funkyrect", "foo",
  "bar1", "funkyrect", "bar",
  "bar2", "funkyrect", "bar"
)
verify_column_groups(column_groups, column_info)

```

verify_column_info	<i>Verify the integrity of the column info object</i>
--------------------	---

Description

Verify the integrity of the column info object

Usage

```
verify_column_info(column_info, data)
```

Arguments

column_info	<p>A data frame describing which columns in data to plot. This data frame should contain the following columns:</p> <ul style="list-style-type: none"> • <code>id</code> (character, required): A column name in data to plot. Determines the size of the resulting geoms, and also the color unless color is specified. • <code>id_color</code> (character): A column name in data to use for the color of the resulting geoms. If NA, the <code>id</code> column will be used. • <code>id_size</code> (character): A column name in data to use for the size of the resulting geoms. If NA, the <code>id</code> column will be used. • <code>name</code> (character): A label for the column. If NA or "", no label will be plotted. If this column is missing, <code>id</code> will be used to generate the name column. • <code>geom</code> (character): The geom of the column. Must be one of: "funkyrect", "circle", "rect", "bar", "pie", "text" or "image". For "text", the corresponding column in data must be a character. For "pie", the column must be a list of named numeric vectors. For all other geoms, the column must be a numeric.
-------------	---

- **group** (character): The grouping id of each column, must match with `column_groups$group`. If this column is missing or all values are NA, columns are assumed not to be grouped.
- **palette** (character): Which palette to colour the geom by. Each value should have a matching value in `palettes$palette`.
- **width**: Custom width for this column (default: 1).
- **overlay**: Whether to overlay this column over the previous column. If so, the width of that column will be inherited.
- **legend**: Whether or not to add a legend for this column.
- **hjust**: Horizontal alignment of the bar, must be between [0,1] (only for `geom = "bar"`).
- **vjust**: Vertical alignment of the label, must be between [0,1] (only for `geom = "text"`).
- **size**: Size of the label, must be a numeric value (only for `geom = "text"`).
- **label**: Which column to use as a label (only for `geom = "text"`).
- **directory**: Which directory to use to find the images (only for `geom = "image"`).
- **extension**: The extension of the images (only for `geom = "image"`).
- **draw_outline**: Whether or not to draw bounding guides (only for `geom = "bar"`). Default: TRUE.
- **options** (list or json): Any of the options above. Any values in this column will be spread across the other columns. This is useful for not having to provide a data frame with 1000s of columns. This column can be a json string.

`data` A data frame with items by row and features in the columns. Must contain one column named "id".

Value

The column info object with all expected columns.

Examples

```
library(tibble)
data <- tribble(
  ~id, ~name, ~x, ~y,
  "foo", "Foo", 0.5, 0.7,
  "bar", "Bar", 1.0, 0.1
)
column_info <- tribble(
  ~id, ~geom,
  "name", "text",
  "x", "funkyrect",
  "y", "funkyrect"
)
verify_column_info(column_info, data)
```

verify_data	<i>Verify the integrity of the data object</i>
-------------	--

Description

Verify the integrity of the data object

Usage

```
verify_data(data)
```

Arguments

data	A data frame with items by row and features in the columns. Must contain one column named "id".
------	---

Value

A verified data object

Examples

```
library(tibble)
data <- tribble(
  ~id, ~name, ~x, ~y,
  "foo", "Foo", 0.5, 0.7,
  "bar", "Bar", 1.0, 0.1
)
verify_data(data)
```

verify_legends	<i>Verify the integrity of the legends object</i>
----------------	---

Description

Verify the integrity of the legends object

Usage

```
verify_legends(legends, palettes, column_info, data)
```

Arguments

- legends** A list of legends to add to the plot. Each entry in `column_info$legend` should have a corresponding entry in this object. Each entry should be a list with the following names:
- **palette** (character): The palette to use for the legend. Must be a value in `palettes`.
 - **geom** (character): The geom of the legend. Must be one of: "funkyrect", "circle", "rect", "bar", "pie", "text", "image".
 - **title** (character, optional): The title of the legend. Defaults to the palette name.
 - **enabled** (logical, optional): Whether or not to add the legend. Defaults to TRUE.
 - **labels** (character, optional): The labels to use for the legend. The defaults depend on the selected geom.
 - **size** (numeric, optional): The size of the listed geoms. The defaults depend on the selected geom.
 - **color** (character, optional): The color of the listed geoms. The defaults depend on the selected geom.
 - **values** (optional): Used as values for the text and image geoms.
 - **label_width** (numeric, optional): The width of the labels (only when geom is text or pie). Defaults to 1 for text and 2 for images.
 - **value_width** (numeric, optional): The width of the values (only for geom = "text"). Defaults to 2.
 - **label_hjust** (numeric, optional): The horizontal alignment of the labels (only when geom is circle, rect or funkyrect). Defaults to 0.5.
- palettes** A named list of palettes. Each entry in `column_info$palette` should have an entry in this object. If an entry is missing, the type of the column will be inferred (categorical or numerical) and one of the default palettes will be applied. Alternatively, the name of one of the standard palette names can be used:
- **numerical**: "Greys", "Blues", "Reds", "YlOrBr", "Greens"
 - **categorical**: "Set3", "Set1", "Set2", "Dark2"
- column_info** A data frame describing which columns in data to plot. This data frame should contain the following columns:
- **id** (character, required): A column name in data to plot. Determines the size of the resulting geoms, and also the color unless `color` is specified.
 - **id_color** (character): A column name in data to use for the color of the resulting geoms. If NA, the `id` column will be used.
 - **id_size** (character): A column name in data to use for the size of the resulting geoms. If NA, the `id` column will be used.
 - **name** (character): A label for the column. If NA or "", no label will be plotted. If this column is missing, `id` will be used to generate the name column.
 - **geom** (character): The geom of the column. Must be one of: "funkyrect", "circle", "rect", "bar", "pie", "text" or "image". For "text", the

corresponding column in data must be a character. For "pie", the column must be a list of named numeric vectors. For all other geoms, the column must be a numeric.

- **group** (character): The grouping id of each column, must match with `column_groups$group`. If this column is missing or all values are NA, columns are assumed not to be grouped.
- **palette** (character): Which palette to colour the geom by. Each value should have a matching value in `palettes$palette`.
- **width**: Custom width for this column (default: 1).
- **overlay**: Whether to overlay this column over the previous column. If so, the width of that column will be inherited.
- **legend**: Whether or not to add a legend for this column.
- **hjust**: Horizontal alignment of the bar, must be between [0,1] (only for `geom = "bar"`).
- **vjust**: Vertical alignment of the label, must be between [0,1] (only for `geom = "text"`).
- **size**: Size of the label, must be a numeric value (only for `geom = "text"`).
- **label**: Which column to use as a label (only for `geom = "text"`).
- **directory**: Which directory to use to find the images (only for `geom = "image"`).
- **extension**: The extension of the images (only for `geom = "image"`).
- **draw_outline**: Whether or not to draw bounding guides (only for `geom == "bar"`). Default: TRUE.
- **options** (list or json): Any of the options above. Any values in this column will be spread across the other columns. This is useful for not having to provide a data frame with 1000s of columns. This column can be a json string.

data A data frame with items by row and features in the columns. Must contain one column named "id".

Value

The legends object in the expected format.

Examples

```
library(tibble)
library(grDevices)
library(RColorBrewer)

# explicit form
data <- tribble(
  ~id, ~name, ~x, ~y,
  "foo", "Foo", 0.5, 0.7,
  "bar", "Bar", 1.0, 0.1
)
column_info <- tribble(
  ~id, ~geom, ~palette,
```



```

  "name", "text", NA,
  "foo", "funkyrect", "pal1",
  "bar", "funkyrect", "pal2"
)
palettes <- list(
  pal1 = rev(brewer.pal(9, "Greys"))[-1]),
  pal2 = rev(brewer.pal(9, "Reds"))[-8:-9])
)
legends <- list()
verify_legends(legends, palettes, column_info, data)

```

verify_palettes	<i>Verify the integrity of the palettes object</i>
-----------------	--

Description

Verify the integrity of the palettes object

Usage

```
verify_palettes(palettes, column_info, data)
```

Arguments

- | | |
|-------------|---|
| palettes | <p>A named list of palettes. Each entry in <code>column_info\$palette</code> should have an entry in this object. If an entry is missing, the type of the column will be inferred (categorical or numerical) and one of the default palettes will be applied. Alternatively, the name of one of the standard palette names can be used:</p> <ul style="list-style-type: none"> numerical: "Greys", "Blues", "Reds", "YlOrBr", "Greens" categorical: "Set3", "Set1", "Set2", "Dark2" |
| column_info | <p>A data frame describing which columns in data to plot. This data frame should contain the following columns:</p> <ul style="list-style-type: none"> <code>id</code> (character, required): A column name in data to plot. Determines the size of the resulting geoms, and also the color unless color is specified. <code>id_color</code> (character): A column name in data to use for the color of the resulting geoms. If NA, the <code>id</code> column will be used. <code>id_size</code> (character): A column name in data to use for the size of the resulting geoms. If NA, the <code>id</code> column will be used. <code>name</code> (character): A label for the column. If NA or "", no label will be plotted. If this column is missing, <code>id</code> will be used to generate the name column. <code>geom</code> (character): The geom of the column. Must be one of: "funkyrect", "circle", "rect", "bar", "pie", "text" or "image". For "text", the corresponding column in data must be a character. For "pie", the column must be a list of named numeric vectors. For all other geoms, the column must be a numeric. |

- **group** (character): The grouping id of each column, must match with `column_groups$group`. If this column is missing or all values are NA, columns are assumed not to be grouped.
- **palette** (character): Which palette to colour the geom by. Each value should have a matching value in `palettes$palette`.
- **width**: Custom width for this column (default: 1).
- **overlay**: Whether to overlay this column over the previous column. If so, the width of that column will be inherited.
- **legend**: Whether or not to add a legend for this column.
- **hjust**: Horizontal alignment of the bar, must be between [0,1] (only for `geom = "bar"`).
- **vjust**: Vertical alignment of the label, must be between [0,1] (only for `geom = "text"`).
- **size**: Size of the label, must be a numeric value (only for `geom = "text"`).
- **label**: Which column to use as a label (only for `geom = "text"`).
- **directory**: Which directory to use to find the images (only for `geom = "image"`).
- **extension**: The extension of the images (only for `geom = "image"`).
- **draw_outline**: Whether or not to draw bounding guides (only for `geom == "bar"`). Default: TRUE.
- **options** (list or json): Any of the options above. Any values in this column will be spread across the other columns. This is useful for not having to provide a data frame with 1000s of columns. This column can be a json string.

data A data frame with items by row and features in the columns. Must contain one column named "id".

Value

The palettes object with all expected columns.

Examples

```
library(tibble)
library(grDevices)
library(RColorBrewer)

# explicit form
data <- tribble(
  ~id, ~name, ~x, ~y,
  "foo", "Foo", 0.5, 0.7,
  "bar", "Bar", 1.0, 0.1
)
column_info <- tribble(
  ~id, ~geom, ~palette,
  "name", "text", NA,
  "foo", "funkyrect", "pal1",
  "bar", "funkyrect", "pal2"
```

```

)
palettes <- list(
  pal1 = rev(brewer.pal(9, "Greys")[-1]),
  pal2 = rev(brewer.pal(9, "Reds")[-8:-9])
)
verify_palettes(palettes, column_info, data)

# implicit palettes
palettes <- list(
  pal1 = "Greys",
  pal2 = "Reds"
)
verify_palettes(palettes, column_info, data)

# passing a tibble should also work (for backwards compatibility)
palettes <- tribble(
  ~palette, ~colours,
  "pal1", rev(brewer.pal(9, "Greys")[-1]),
  "pal2", rev(brewer.pal(9, "Reds")[-8:-9])
)
verify_palettes(palettes, column_info, data)

```

verify_row_groups	<i>Verify the integrity of the row groups object</i>
-------------------	--

Description

Verify the integrity of the row groups object

Usage

```
verify_row_groups(row_groups, row_info)
```

Arguments

row_groups	<p>A data frame describing of how to group the rows in row_info. Can consist of the following columns:</p> <ul style="list-style-type: none"> • group (character): The corresponding group in row_info\$group. • level1 (character): The label at the highest level. • level2 (character, optional): The label at the middle level. • level3 (character, optional): The label at the lowest level (not recommended).
row_info	<p>A data frame describing the rows of data. This data should contain two columns:</p> <ul style="list-style-type: none"> • id (character): Corresponds to the column data\$id. • group (character): The group of the row. If all are NA, the rows will not be split up into groups.

Value

The row groups object with all expected rows.

Examples

```
library(tibble)
row_groups <- tribble(
  ~group, ~level1,
  "foo", "Foo",
  "bar", "Bar"
)
row_info <- tribble(
  ~id, ~group,
  "name", NA_character_,
  "foo1", "foo",
  "foo2", "foo",
  "bar1", "bar",
  "bar2", "bar"
)
verify_row_groups(row_groups, row_info)
```

verify_row_info

Verify the integrity of the row info object

Description

Verify the integrity of the row info object

Usage

```
verify_row_info(row_info, data)
```

Arguments

row_info	A data frame describing the rows of data. This data should contain two columns: <ul style="list-style-type: none"> id (character): Corresponds to the column data\$id. group (character): The group of the row. If all are NA, the rows will not be split up into groups.
data	A data frame with items by row and features in the columns. Must contain one column named "id".

Value

The row info object with all expected columns.

Examples

```
library(tibble)
data <- tribble(
  ~id, ~name, ~x, ~y,
  "foo1", "Foo1", 0.5, 0.7,
  "foo2", "Foo2", 0.5, 0.8,
  "bar1", "Bar1", 1.0, 0.2,
  "bar2", "Bar2", 1.0, 0.1
)
row_info <- tribble(
  ~id, ~group,
  "foo1", "foo",
  "foo2", "foo",
  "bar1", "bar",
  "bar2", "bar"
)
verify_row_info(row_info, data)
```

Index

* datasets

dynbenchmark_data, 2
scib_summary, 10

aes(), 6

borders(), 7

dynbenchmark_data, 2

fortify(), 6

funky_heatmap, 3

geom_rounded_rect, 6

ggplot(), 6

ggplot2::geom_rect(), 6

ggsave(), 6

layer(), 7

position_arguments, 8

scale_minmax, 9

scib_summary, 10

verify_column_groups, 10

verify_column_info, 12

verify_data, 14

verify_legends, 14

verify_palettes, 17

verify_row_groups, 19

verify_row_info, 20