Package 'funkyheatmap'

March 1, 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>.

Version 0.3.0

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports assertthat, cli, cowplot, dplyr, ggforce, ggplot2, grDevices, jsonlite, purrr, RColorBrewer, Rdpack, stringr, tibble, tidyr

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

VignetteBuilder knitr

RdMacros Rdpack

Depends R (>= 2.10)

LazyData true

URL https://funkyheatmap.dynverse.org, https://github.com/dynverse/funkyheatmap

BugReports https://github.com/dynverse/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 2023-03-01 22:30:02 UTC

R topics documented:

dynbenchmark_data . . . 2
funky_heatmap . . . 2
geom_rounded_rect . . . 5
verify_column_groups . . . 7
verify_column_info . . . 9
verify_data . . . 10
verify_palettes . . . 11
verify_row_groups . . . 13
verify_row_info . . . 14

Index **16**

`dynbenchmark_data` *The results data frame from dynbenchmark.*

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/s41587-019-0071-9.

`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,
  scale_column = TRUE,
  add_abc = TRUE,
  col_annot_offset = 3,
  col_annot_angle = 30,
  removed_entries = NULL,
  expand = c(xmin = 0, xmax = 2, ymin = 0, ymax = 0)
)
```

Arguments

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

`column_info` A data frame describing which columns in `data` to plot. This data frame should contain the following columns:

- `id` (`character`): The corresponding column name in `data`.
- `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"`, or `"text"`. 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"`).
- `hjust`: Horizontal alignment of the label, must be between [0,1] (only for `geom = "text"`).
- `vjust`: Vertical alignment of the label, must be between [0,1] (only for `geom = "text"`).

- size: Size of the label, must be between [0,1] (only for geom = "text").
- label: Which column to use as a label (only for geom = "text").
- 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 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"

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
How much the column annotation will be offset by.

col_annot_angle
The angle of the column annotation labels.

removed_entries
Which methods to not show in the rows. Missing methods are replaced by a "Not shown" label.

expand A list of directions to expand the plot in.

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 `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

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"
 )
```

`verify_column_groups` *Verify the integrity of the column groups object*

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`): The corresponding column name in `data`. • `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"`, or `"text"`. 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"`).
- `hjust`: Horizontal alignment of the label, must be between [0,1] (only for `geom = "text"`).
- `vjust`: Vertical alignment of the label, must be between [0,1] (only for `geom = "text"`).
- `size`: Size of the label, must be between [0,1] (only for `geom = "text"`).
- `label`: Which column to use as a label (only for `geom = "text"`).
- `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 *Verify the integrity of the column info object*

Description

Verify the integrity of the column info object

Usage

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

Arguments

column_info — A data frame describing which columns in data to plot. This data frame should contain the following columns:

- id (character): The corresponding column name in data.
- 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", or "text". 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").
- hjust: Horizontal alignment of the label, must be between [0,1] (only for geom = "text").
- vjust: Vertical alignment of the label, must be between [0,1] (only for geom = "text").
- size: Size of the label, must be between [0,1] (only for geom = "text").
- label: Which column to use as a label (only for geom = "text").
- 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 *Verify the integrity of the data object*

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_palettes` *Verify the integrity of the palettes object*

Description

Verify the integrity of the palettes object

Usage

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

Arguments

`palettes`	A named list of palettes. Each entry in `column_info$palette` should have an entry in this object. If an entry is missing, the type of the column will be inferred (categorical or numerical) and one of the default palettes will be applied. Alternatively, the name of one of the standard palette names can be used: • `numerical`: `"Greys"`, `"Blues"`, `"Reds"`, `"YlOrBr"`, `"Greens"` • `categorical`: `"Set3"`, `"Set1"`, `"Set2"`, `"Dark2"`
`column_info`	A data frame describing which columns in `data` to plot. This data frame should contain the following columns: • `id` (`character`): The corresponding column name in `data`. • `name` (`character`): A label for the column. If `NA` or `""`, no label will be plotted. If this column is missing, `id` will be used to generate the `name` column. • `geom` (`character`): The geom of the column. Must be one of: `"funkyrect"`, `"circle"`, `"rect"`, `"bar"`, `"pie"`, or `"text"`. For `"text"`, the corresponding column in `data` must be a `character`. For `"pie"`, the column must be a list of named numeric vectors. For all other geoms, the column must be a `numeric`. • `group` (`character`): The grouping id of each column, must match with `column_groups$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"`).
- `hjust`: Horizontal alignment of the label, must be between [0,1] (only for `geom = "text"`).
- `vjust`: Vertical alignment of the label, must be between [0,1] (only for `geom = "text"`).
- `size`: Size of the label, must be between [0,1] (only for `geom = "text"`).
- `label`: Which column to use as a label (only for `geom = "text"`).
- `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` *Verify the integrity of the row groups object*

Description

Verify the integrity of the row groups object

Usage

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

Arguments

`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$group`. • `level1` (`character`): The label at the highest level. • `level2` (`character`, optional): The label at the middle level. • `level3` (`character`, optional): The label at the lowest level (not recommended).
`row_info`	A data frame describing the rows of `data`. This data should contain two columns: • `id` (`character`): Corresponds to the column `data$id`. • `group` (`character`): The group of the row. If all are `NA`, the rows will not be split up into groups.

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: • `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

aes(), *5*

borders(), *6*

dynbenchmark_data, 2

fortify(), *5*
funky_heatmap, 2

geom_rounded_rect, 5
ggplot(), *5*
ggplot2::geom_rect(), *5*
ggsave(), *5*

layer(), *6*

verify_column_groups, 7
verify_column_info, 9
verify_data, 10
verify_palettes, 11
verify_row_groups, 13
verify_row_info, 14