Package 'hhmR'

July 22, 2025

Type Package

Title Hierarchical Heatmaps

Version 0.0.1

Maintainer Michael Mahony <michael.mahony@cantab.net>

Description Allows users to create high-quality heatmaps from labelled, hierarchical data. Specifically, for data with a two-level hierarchical structure, it will produce a heatmap where each row and column represents a category at the lower level. These rows and columns are then grouped by the higher-level group each category belongs to, with the names for each category and groups shown in the margins. While other packages (e.g. 'dendextend') allow heatmap rows and columns to be arranged by groups only, 'hhmR' also allows the labelling of the data at both the category and group level.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/sgmmahon/hhmR, https://sgmmahon.github.io/hhmR/

BugReports https://github.com/sgmmahon/hhmR/issues

Depends R (>= 3.5.0)

Imports dplyr, purrr, tidyr, rlang, grid, ggplot2, patchwork, grDevices, magrittr, utils

Language en-UK

LazyData true

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Michael Mahony [cre, aut, cph] (ORCID: <https://orcid.org/0000-0003-2784-2745>), Francisco Rowe [aut] (ORCID: <https://orcid.org/0000-0003-4137-0246>), Carmen Cabrera-Arnau [aut] (ORCID: <https://orcid.org/0000-0002-2732-6436>)

Repository CRAN

Date/Publication 2025-01-14 15:10:01 UTC

Contents

`cg` *cg*

Description

Creates colour gradient between two hexcodes.

Usage

```
cg(colour1, colour2, n = 15)
```

Arguments

`colour1`	The first hexcode colour.
`colour2`	The second hexcode colour.
`n`	The length of the vector returned by the function.

Value

A vector of hexcodes of length n, containing a colour gradient between colour =1 and colour2.

Examples

```
cg("white","black",20)
```

`decimalplaces` *decimalplaces*

Description

Tests the number of non-zero decimal places within a number.

Usage

```
decimalplaces(x)
```

Arguments

`x`	The number for the number of decimal places is to be measured.

Value

A single number, indicating the number of non-zero decimal places in ‘x‘.

Examples

```
decimalplaces(23.43234525)
decimalplaces(334.3410000000000000)
decimalplaces(2.000)
```

`example_migration` *example_migration*

Description

Fake migration dataset used to demonstrate the functionality of the hhm function in the hhmR package. It contains the information on the number of people who have moved between a series of fictional geographies. The geographies themselves have a hierarchical structure, with each county existing within a smaller subset of regions.

Usage

```
data(example_migration)
```

Format

A data frame with 324 rows and 5 variables.

Origin County The county (lower-level geography) that each migrant began in.

Destination County The county (lower-level geography) that each migrant ended up in.

Origin Region The region (higher-level geography) that each migrant began in.

Destination Region The region (higher-level geography) that each migrant ended up in.

Migration The number of migrants that moved from each origin county to each destination county.

Examples

```
library(dplyr)

# Code to create dataset

# Define names of fake counties
fake_counties = c("Greenridge","Windermoor","Bramblewood","Silverlake",
                  "Thornbury","Maplewood","Hawthorne","Pinehurst",
                  "Riverton","Meadowbrook","Fairhaven","Oakdale","Stonebridge",
                  "Brookfield","Ashford","Glenville","Sunnyvale","Westfield")

# Create region county lookup tables
rc_lkp = data.frame(region = c(rep("North",3),rep("Midlands",5),
                               rep("South West",4),rep("South East",6)),
                    county = fake_counties)
og_lkp = rc_lkp %>% setNames(c("Origin Region"     ,"Origin County"     ))
dn_lkp = rc_lkp %>% setNames(c("Destination Region","Destination County"))

# Create dataframe of fake migration data
set.seed(1234)
example_migration = expand.grid(fake_counties,fake_counties) %>%
                    setNames(paste(c("Origin","Destination"),"County",sep=" ")) %>%
                    full_join(og_lkp) %>% full_join(dn_lkp) %>%
                    mutate(Migration = (1/rgamma(18*18, shape = 17, rate = 0.5)) %>%
                                       {. * 1000} %>% round())
example_migration[example_migration$`Origin County` ==
                  example_migration$`Destination County`,"Migration"] =
 example_migration[example_migration$`Origin County` ==
                   example_migration$`Destination County`,"Migration"] * 10
```

`example_time_series` *example_time_series*

Description

Fake migration dataset used to demonstrate the functionality of the tshm function in the hhmR package. It contains the information on the number of people who have immigrated a series of fictional geographies over the years 2011 to 2015. The geographies themselves have a hierarchical structure, with each county existing within a smaller subset of regions.

Usage

```
data(example_time_series)
```

Format

A data frame with 90 rows and 4 variables.

County The county (lower-level geography) that immigrants move to.

Region The region (higher-level geography) that immigrants move to.

Year The year during which each wave of immigration occured.

Immigration The number of immigrants that moved each county in each year.

Examples

```
library(dplyr)
library(tidyr)

# Define names of fake counties
fake_counties = c("Greenridge","Windermoor","Bramblewood","Silverlake",
                  "Thornbury","Maplewood","Hawthorne","Pinehurst",
                  "Riverton","Meadowbrook","Fairhaven","Oakdale","Stonebridge",
                  "Brookfield","Ashford","Glenville","Sunnyvale","Westfield")

# Create dataframe of fake migration data
set.seed(1234)
example_time_series = data.frame(region = c(rep("North",3),rep("Midlands",5),
                                             rep("South West",4),rep("South East",6)),
                                 county = fake_counties,
                                 year_2011 = sample(1:10000,length(fake_counties)),
                                 year_2012 = sample(1:10000,length(fake_counties)),
                                 year_2013 = sample(1:10000,length(fake_counties)),
                                 year_2014 = sample(1:10000,length(fake_counties)),
                                 year_2015 = sample(1:10000,length(fake_counties))) %>%
  setNames(c("Region","County",2011:2015)) %>%
  pivot_longer(cols = `2011`:`2015`,
                      names_to = "Year",
                      values_to = "Immigration") %>%
  mutate(Year = as.numeric(Year))
example_time_series[sample(1:(length(fake_counties)*5),5),"Immigration"] = NA
```

---

exp_seq *exp_seq*

Description

Creates a vector of exponentially increasing values between 0 and a specified value ‘n‘. If ‘n‘ is specified as 1, the vector will be scaled to between 0 and 1.

Usage

```
exp_seq(n, ln = 15, exponent = 2, round_values = TRUE, rmv_extremes = TRUE)
```

Arguments

`n`	The maximum value that the values in the sequence are scaled to.
`ln`	How long the vector should be (defaults to 15).
`exponent`	The exponential power with which to multiply the sequence by (defaults to 2).
`round_values`	Option to round values to whole numbers (defaults to ‘TRUE‘). If ‘n‘ equals 1, round_values will automatically be set to FALSE.
`rmv_extremes`	Option to remove zero and the maximum value (i.e. ‘n‘) from the beginning and the end of the returned vector (defaults to ‘FALSE‘). Note that this will mean the length of the returned vector will be ‘n‘ - 2.

Value

A vector containing exponentially increasing values between 0 and a specified value ‘n‘.

Examples

```
# Create sequence of length 8, scaled between 0 and 10000
exp_seq(10000,8)
# Set rmv_extremes = FALSE to get full sequence
exp_seq(10000,8,rmv_extremes = FALSE)
# The exponent defaults to 2. Setting it to between 1 and 2 causes it to converge on
# a linear sequence. When exponent is set to 1 the sequence increases linearly
exp_seq(10000,8,exponent=1)
# Setting it to greater than 2 will cause it the values in the sequence to shift towards zero
exp_seq(10000,8,exponent=4)

# Create sequence of length 12, scaled between 0 and 1
exp_seq(1,12)
exp_seq(1,12,rmv_extremes = FALSE)
exp_seq(1,12,exponent=1)
exp_seq(1,12,exponent=4)
```

---

hhm *Hierarchical Heatmap*

Description

Creates a labelled heatmap from heirarchical data. This function is useful if you wish to create a heatmap where the categories shown on both the x and y axis can be grouped in some way. This heatmap will order the categories by their assigned group and present both the categories and group labels along the axes. An example might be a series of smaller geographies (lower categories) which aggregate into larger geographical regions (upper groups).

Usage

```
hhm(
  df,
  ylower,
  yupper,
  xlower,
  xupper,
  values,
  rm_diag = FALSE,
  lgttl = NULL,
  bins = NULL,
  cbrks = NULL,
  cclrs = NULL,
  norm_lgd = FALSE,
  lgdps = 0,
  xttl_height = 0.15,
  yttl_width = 0.15
)
```

Arguments

`df`	A data.frame with containing values with which to populate the heatmap. The data.frame must include columns specifying the lower categories (‘ylower‘, ‘xlower‘) and upper groups (‘yupper‘, ‘xupper‘) that each value corresponds to. These categories and groups will be used to arrange and label the rows and columns of the heatmap. It must also contain a ‘values‘ variable containing the values used to populate the heatmap. Note that the groups will by default be arranged alphabetically (top to bottom / left to right). The ordering of the groups can be manually specified by converting yupper and/or xupper to factors. In this case, the groups will be ordered based on the ordering of the factor levels.
`ylower`	A column in ‘df‘ containing the categories that will be presented as rows along the y-axis of the heatmap.
`yupper`	A column in ‘df‘ containing the groupings that will be used to arrange the heatmap rows.
`xlower`	A column in ‘df‘ containing the categories that will be presented as columns along the x-axis of the heatmap.
`xupper`	A column in ‘df‘ containing the groupings that will be used to arrange the heatmap columns.
`values`	A column in ‘df‘ containing the values used to populate the heatmap.
`rm_diag`	Do not show values for categories along the x and y axes that are identical (defaults to ‘FALSE‘). This is particularly useful for origin-destination heatmaps, where the user may want to hide the diagonal values.
`lgttl`	Option to manually define legend title.
`bins`	Option to break the data into a specified number of groups (defaults to ‘NULL‘). The thresholds between these groups will be equally spaced between zero and the maximum value observed in ‘values‘.

`cbrks`	Vector of custom breaks, if users wish to use a discrete legend colour scheme (defaults to 'NULL'). For example, a supplied vector of 'c(5,10, 20)' would break he values up into 5 ordered groups of ranges 0, 0-5, 5-10, 10-20 and 20+.
`cclrs`	Vector of hexcodes, which to create a custom legend colour scheme (defaults to 'NULL'). If 'cbrks' is supplied, 'cclrs' must have a length two longer than 'cbrks'. If 'bins' is supplied, 'cclrs' must have a length equal to the values provided to 'bins'.
`norm_lgd`	Normalised to between 0 and 1 in legend (defaults to 'FALSE'). Allows for consistency when comparing heatmaps across different datasets. At present, this only works if all heatmap values are positive.
`lgdps`	If using custom breaks, define the number of decimal points to round the legend scale to (defaults to 0). If 'norm_lgd' is 'TRUE', it will default to 3.
`xttl_height`	The space allocated to the group titles on the x-axis as a proportion of the heatmap's height (defaults to 0.15).
`yttl_width`	The space allocated to the group titles on the y-axis as a proportion of the heatmap's width (defaults to 0.15).

Value

A ggplot object containing the final heatmap.

Examples

```
# Import toy demonstration dataset (see `?example_migration` for see details)
data(example_migration)

# Intial heatmap
hierarchical_heatmap = hhm(df = example_migration,
                           ylower = "Origin County",
                           xlower = "Destination County",
                           yupper = "Origin Region",
                           xupper = "Destination Region",
                           values = "Migration",
                           yttl_width = 0.22,
                           xttl_height = 0.4)

# For more details, see the package vignette at
# https://sgmmahon.github.io/hhmR/articles/hhmR_overview.html
```

`log_seq` *log_seq*

Description

Creates a vector of logarithmicly increasing values between 0 and a specified value 'n'. If 'n' is specified as 1, the vector will be scaled to between 0 and 1.

Usage

```
log_seq(n, ln = 15, round_values = TRUE, rmv_extremes = FALSE)
```

Arguments

`n`	The maximum value that the values in the sequence are scaled to.
`ln`	How long the vector should be (defaults to 15).
`round_values`	Option to round values to whole numbers (defaults to ‘TRUE‘).
`rmv_extremes`	Option to remove zero and the maximum value (i.e. ‘n‘) from the beginning and the end of the returned vector (defaults to ‘FALSE‘). Note that this will mean the length of the returned vector will be ‘n‘ - 2.

Value

A vector containing logarithmicly increasing values between 0 and a specified value ‘n‘.

Examples

```
# Create sequence of length 20, scaled between 0 and 500
log_seq(500,20)

# Create sequence of length 15, scaled between 0 and 1
log_seq(1,12)
```

`plt_ttl` *plt_ttl*

Description

Creates plot containing the name of a given upper group. Used in combination with the patchwork package to plot the names of the upper groups within the hhm function.

Usage

```
plt_ttl(ttl, axs = "x", rotate_title = TRUE)
```

Arguments

`ttl`	The name of the upper group.
`axs`	The axis on which the name will appear (defaults to "x"). If ‘x‘, the text will be written at the top-centre of the plot. If ‘y‘, the text will be written at the middle-right of the plot.
`rotate_title`	Whether the title should be rotate to be perpendicular to the axis (defaults to TRUE). If TRUE, the title text on the x and y axes will be printed horizontally and vertically respectively, with the reverse orientation if set to FALSE.

Value

A ggplot object containing the title of a given upper group, for use in the hhm function.

Examples

```
plt_ttl("Group 1", axs = "y")
plt_ttl("Group 2")
plt_ttl("Group 1", axs = "y",rotate_title = FALSE)
plt_ttl("Group 2"          ,rotate_title = FALSE)
```

---

tshhm   *Time-series Hierarchical Heatmap*

Description

Creates a labelled time-series heatmap from heirarchical data. This function is useful if you wish to create a time-series heatmap where the categories shown on the y axis can be grouped in some way. This heatmap will order the categories by their assigned group and present both the categories and group labels along the y-axis. An example might be series of smaller geographies (lower categories) which aggregate into larger geographical regions (upper groups).

Usage

```
tshhm(
  df,
  lower,
  upper,
  times,
  values,
  sort_lower = "alphabetical",
  lgttl = NULL,
  bins = NULL,
  cbrks = NULL,
  cclrs = NULL,
  norm_lgd = FALSE,
  lgdps = 0,
  na_colour = NULL,
  xttl_height = 0.05,
  yttl_width = 0.15
)
```

Arguments

df	A data.frame with containing values with which to populate the heatmap. The data.frame must include columns specifying the lower categories (‘lower‘) and upper groups (‘upper‘) that each value corresponds to. These categories and

	groups will be used to arrange and label the rows of the heatmap. ‘df‘ must also contain a ‘values‘ variable, containing the values used to populate the heatmap, and a ‘times‘ variable, containing the time period during which each value was observed. Note that the groups in ‘upper‘ will by default be arranged alphabetically (top to bottom). The ordering of the groups can be manually specified by converting ‘upper‘ to a factor. In this case, the groups will be ordered based on the ordering of the factor levels. The ordering of rows within each group can also be specified using the ‘sort_lower‘ variable.
`lower`	A column in ‘df‘ containing the categories that will be presented as rows along the y-axis of the heatmap.
`upper`	A column in ‘df‘ containing the groupings that will be used to arrange the heatmap rows.
`times`	A column in ‘df‘ containing the time-period during which each each value in ‘values‘ was observed.
`values`	A column in ‘df‘ containing the values used to populate the heatmap.
`sort_lower`	Option to define how rows (lower) within each group (upper) are ordered. The default option is ‘alphabetical‘, which orders rows in alphabetical order from top to bottom. Other options include ‘sum_ascend‘ and ‘mean_ascend‘, which order rows in ascending order (top to bottom) based on the row totals and row means respectively. This order can be reversed with the options ‘sum_descend‘ and ‘mean_descend‘.
`lgttl`	Option to manually define legend title.
`bins`	Option to break the data into a specified number of groups (defaults to ‘NULL‘). The thresholds between these groups will be equally spaced between the minimum and maximum values observed in ‘values‘.
`cbrks`	Vector of custom breaks, if users wish to use a discrete legend colour scheme (defaults to ‘NULL‘). For example, a supplied vector of ‘c(5,10, 20)‘ would break he values up into 5 ordered groups of ranges 0, 0-5, 5-10, 10-20 and 20+.
`cclrs`	Vector of hexcodes, which to create a custom legend colour scheme (defaults to ‘NULL‘). If ‘cbrks‘ is supplied, ‘cclrs‘ must have a length two longer than ‘cbrks‘. If ‘bins‘ is supplied, ‘cclrs‘ must have a length equal to the values provided to ‘bins‘.
`norm_lgd`	Normalised to between 0 and 1 in legend (defaults to ‘FALSE‘). Allows for consistency when comparing heatmaps across different datasets. At present, this only works if all heatmap values are positive.
`lgdps`	If using custom breaks, define the number of decimal points to round the legend scale to (defaults to 0). If ‘norm_lgd‘ is ‘TRUE‘, it will default to 3.
`na_colour`	Option to define the colour of NA values in the legend (defaults to ‘NULL‘, meaning NA values will be assigned no colour).
`xttl_height`	The space allocated to the title on the x-axis as a proportion of the heatmap’s height (defaults to 0.05).
`yttl_width`	The space allocated to the group titles on the y-axis as a proportion of the heatmap’s width (defaults to 0.15).

Value

A ggplot object containing the final heatmap.

Examples

```
library(dplyr)

# Import toy demonstration dataset (see `?example_time_series` for see details)
data(example_time_series)

# Intial heatmap
time_series_heatmap = tshhm(df = example_time_series,
                            lower  = "County",
                            upper  = "Region",
                            times  = "Year",
                            values = "Immigration",
                            yttl_width  = 0.25)

# View result
time_series_heatmap

# For more details, see the package vignette at
# https://sgmmahon.github.io/hhmR/articles/hhmR_overview.html
```

Index