Package ‘easyalluvial’

January 15, 2019

Title Generate Alluvial Plots with a Single Line of Code

Version 0.1.8

URL https://github.com/erblast/easyalluvial

Description Alluvial plots are similar to sankey diagrams and visualise categorical data over multiple dimensions as flows. (Rosvall M, Bergstrom CT (2010) Mapping Change in Large Networks. PLoS ONE 5(1): e8694. <doi:10.1371/journal.pone.0008694> Their graphical grammar however is a bit more complex then that of a regular x/y plots. The 'ggalluvial' package made a great job of translating that grammar into 'ggplot2' syntax and gives you many options to tweak the appearance of an alluvial plot, however there still remains a multi-layered complexity that makes it difficult to use 'ggalluvial' for explorative data analysis. 'easyalluvial' provides a simple interface to this package that allows you to produce a decent alluvial plot from any dataframe in either long or wide format from a single line of code while also handling continuous data. It is meant to allow a quick visualisation of entire dataframes with a focus on different colouring options that can make alluvial plots a great tool for data exploration.

License CC0

Encoding UTF-8

LazyData true

Depends R(>= 2.10)

Suggests testthat, covr, ISLR, nycflights13, vdiffr

RoxygenNote 6.1.1

Imports purrr , tidyr , dplyr , forcats , ggalluvial , ggplot2 , RColorBrewer , recipes , rlang , stringr , magrittr , tibble

Language en-US

NeedsCompilation no

Author Bjoern Koneswarakantha [aut, cre] (<https://orcid.org/0000-0003-4585-7799>)

Maintainer Bjoern Koneswarakantha <datistics@gmail.com>

Repository CRAN

Date/Publication 2019-01-15 05:30:03 UTC

R topics documented:

`alluvial_long` *alluvial plot of data in long format*

Description

Plots two variables of a dataframe on an alluvial plot. A third variable can be added either to the left or the right of the alluvial plot to provide coloring of the flows. All numerical variables are scaled, centered and YeoJohnson transformed before binning.

Usage

```
alluvial_long(data, key, value, id, fill = NULL, fill_right = T,
  bins = 5, bin_labels = c("LL", "ML", "M", "MH", "HH"),
  NA_label = "NA", order_levels_value = NULL,
  order_levels_key = NULL, order_levels_fill = NULL, complete = TRUE,
  fill_by = "first_variable", col_vector_flow = palette_qualitative()
  %>% palette_filter(greys = F),
  col_vector_value = RColorBrewer::brewer.pal(9, "Greys")[c(3, 6, 4, 7,
  5)], verbose = F, stratum_labels = T, stratum_width = 1/4,
  auto_rotate_xlabs = T)
```

Arguments

`data`	a dataframe
`key`	unquoted column name or string of x axis variable
`value`	unquoted column name or string of y axis variable
`id`	unquoted column name or string of id column
`fill`	unquoted column name or string of fill variable which will be used to color flows, Default: NULL
`fill_right`	logical, TRUE fill variable is added to the right FALSE to the left, Default: T

`bins` number of bins for automatic binning of numerical variables, Default: 5

`bin_labels` labels for bins, Default: c("LL", "ML", "M", "MH", "HH")

`NA_label` character vector define label for missing data

`order_levels_value`
: character vector denoting order of y levels from low to high, does not have to be complete can also just be used to bring levels to the front, Default: NULL

`order_levels_key`
: character vector denoting order of x levels from low to high, does not have to be complete can also just be used to bring levels to the front, Default: NULL

`order_levels_fill`
: character vector denoting order of color fill variable levels from low to high, does not have to be complete can also just be used to bring levels to the front, Default: NULL

`complete` logical, insert implicitly missing observations, Default: TRUE

`fill_by` one_of(c('first_variable', 'last_variable', 'all_flows', 'values')), Default: 'first_variable'

`col_vector_flow`
: HEX color values for flows, Default: palette_filter(greys = F)

`col_vector_value`
: HEX color values for y levels/values, Default:RColorBrewer::brewer.pal(9, 'Greys')[c(3,6,4,7,5)]

`verbose` logical, print plot summary, Default: F

`stratum_labels` logical, Default: TRUE

`stratum_width` double, Default: 1/4

`auto_rotate_xlabs`
: logical, Default: TRUE

Value

ggplot2 object

See Also

alluvial_wide ,geom_flow, geom_stratum

Examples

```
 data = quarterly_flights

 alluvial_long( data, key = qu, value = mean_arr_delay, id = tailnum, fill_by = 'last_variable' )

## Not run:
 # more flow coloring variants ------------------------------------

 alluvial_long( data, key = qu, value = mean_arr_delay, id = tailnum, fill_by = 'first_variable' )
 alluvial_long( data, key = qu, value = mean_arr_delay, id = tailnum, fill_by = 'all_flows' )
 alluvial_long( data, key = qu, value = mean_arr_delay, id = tailnum, fill_by = 'value' )
```

```
 # color by additional variable carrier ---------------------------

 alluvial_long( data, key = qu, value = mean_arr_delay, fill = carrier, id = tailnum )

 # use same color coding for flows and y levels -------------------

 palette = c('green3', 'tomato')

 alluvial_long( data, qu, mean_arr_delay, tailnum, fill_by = 'value'
               , col_vector_flow = palette
               , col_vector_value = palette )


 # reorder levels ------------------------------------------------

 alluvial_long( data, qu, mean_arr_delay, tailnum, fill_by = 'first_variable'
              , order_levels_value = c('on_time', 'late') )

 alluvial_long( data, qu, mean_arr_delay, tailnum, fill_by = 'first_variable'
              , order_levels_key = c('Q4', 'Q3', 'Q2', 'Q1') )

require(dplyr)
require(magrittr)

 order_by_carrier_size = data %>%
   group_by(carrier) %>%
   count() %>%
   arrange( desc(n) ) %>%
   .[['carrier']]

 alluvial_long( data, qu, mean_arr_delay, tailnum, carrier
               , order_levels_fill = order_by_carrier_size )


## End(Not run)
```

`alluvial_wide` *alluvial plot of data in wide format*

Description

plots a dataframe as an alluvial plot. All numerical variables are scaled, centered and YeoJohnson transformed before binning. Plots all variables in the sequence as they appear in the dataframe until maximum number of values is reached.

Usage

```
alluvial_wide(data, id = NULL, max_variables = 20, bins = 5,
  bin_labels = c("LL", "ML", "M", "MH", "HH"), NA_label = "NA",
```

```
order_levels = NULL, fill_by = "first_variable",
col_vector_flow = palette_qualitative() %>% palette_filter(greys =
F), col_vector_value = RColorBrewer::brewer.pal(9, "Greys")[c(4, 7, 5,
8, 6)], verbose = F, stratum_labels = T, stratum_width = 1/4,
auto_rotate_xlabs = T)
```

Arguments

`data`	a dataframe
`id`	unquoted column name of id column or character vector with id column name
`max_variables`	maximum number of variables, Default: 20
`bins`	number of bins for numerical variables, Default: 5
`bin_labels`	labels for the bins from low to high, Default: c("LL", "ML", "M", "MH", "HH")
`NA_label`	character vector define label for missing data, Default: 'NA'
`order_levels`	character vector denoting levels to be reordered from low to high
`fill_by`	one_of(c('first_variable', 'last_variable', 'all_flows', 'values')), Default: 'first_variable'
`col_vector_flow`	HEX colors for flows, Default: palette_filter(greys = F)
`col_vector_value`	Hex colors for y levels/values, Default: RColorBrewer::brewer.pal(9, "Greys")[c(3, 6, 4, 7, 5)]
`verbose`	logical, print plot summary, Default: F
`stratum_labels`	logical, Default: TRUE
`stratum_width`	double, Default: 1/4
`auto_rotate_xlabs`	logical, Default: TRUE

Details

Under the hood this function converts the wide format into long format. ggalluvial also offers a way to make alluvial plots directly from wide format tables but it does not allow individual colouring of the stratum segments. The tradeoff is that we can only order levels as a whole and not individually by variable, Thus if some variables have levels with the same name the order will be the same. If we want to change level order independently we have to assign unique level names first.

Value

ggplot2 object

See Also

`alluvial_wide` , `geom_flow`, `geom_stratum`

Examples

```
require(magrittr)
require(dplyr)

data = as_tibble(mtcars)
categoricals = c('cyl', 'vs', 'am', 'gear', 'carb')
numericals = c('mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec')
max_variables = 5

data = data %>%
  mutate_at( vars(categoricals), as.factor )


alluvial_wide( data = data
                , max_variables = max_variables
                , fill_by = 'first_variable' )
## Not run:

# more coloring variants----------------------
alluvial_wide( data = data
                , max_variables = max_variables
                , fill_by = 'last_variable' )

alluvial_wide( data = data
                , max_variables = max_variables
                , fill_by = 'all_flows' )

alluvial_wide( data = data
                , max_variables = max_variables
                , fill_by = 'first_variable' )

# manually order variable values---------------

alluvial_wide( data = data
                 , max_variables = max_variables
                 , fill_by = 'values'
                 , order_levels = c('1', '0') )

## End(Not run)
```

`manip_bin_numerics` *bin numerical columns*

Description

centers, scales and Yeo Johnson transforms numeric variables in a dataframe before binning into n bins of equal range. Outliers based on boxplot stats are capped (set to min or max of boxplot stats).

Usage

```
manip_bin_numerics(x, bins = 5, bin_labels = c("LL", "ML", "M", "MH",
  "HH"), center = T, scale = T, transform = T, round_numeric = T,
  digits = 2)
```

Arguments

x	dataframe with numeric variables, or numeric vector
bins	number of bins for numerical variables, Default: 5
bin_labels	labels for the bins from low to high, Default: c("LL", "ML", "M", "MH", "HH"). Can also be one of c('mean', 'median', 'min_max', 'cuts'), the corresppnding summary function will supply the labels.
center	logical, Default: T
scale	logical, Default: T
transform	logical, apply Yeo Johnson Transformation, Default: T
round_numeric,	logical, rounds numeric results if bin_labels is supplied with a supported summary function name.
digits,	integer, number of digits to round to

Value

dataframe

Examples

```
summary( mtcars )
summary( manip_bin_numerics(mtcars) )
summary( manip_bin_numerics(mtcars, bin_labels = 'mean'))
summary( manip_bin_numerics(mtcars, bin_labels = 'cuts'
  , scale = FALSE, center = FALSE, transform = FALSE))
```

---

manip_factor_2_numeric *converts factor to numeric preserving numeric levels and order in character levels.*

Description

before converting we check whether the levels contain a number, if they do the number will be preserved.

Usage

```
manip_factor_2_numeric(vec)
```

Arguments

vec	vector

Value

vector

See Also

`str_detect`

Examples

```
fac_num = factor( c(1,3,8) )
fac_chr = factor( c('foo','bar') )
fac_chr_ordered = factor( c('a','b','c'), ordered = TRUE )

manip_factor_2_numeric( fac_num )
manip_factor_2_numeric( fac_chr )
manip_factor_2_numeric( fac_chr_ordered )
```

---

palette_filter — *color filters for any vector of hex color values*

Description

filters are based on rgb values

Usage

```
palette_filter(palette = palette_qualitative(), similar = F,
  greys = T, reds = T, greens = T, blues = T, dark = T,
  medium = T, bright = T, thresh_similar = 25)
```

Arguments

palette	any vector with hex color values, Default: palette_qualitative()
similar,	logical, allow similar colours, similar colours are detected using a threshold (thresh_similar), two colours are similar when each value for RGB is within threshold range of the corresponding RGB value of the second colour, Default: F
greys,	logical, allow grey colours, blue == green == blue , Default: T
reds,	logical, allow red colours, blue < 50 & green < 50 & red > 200 , Default: T
greens,	logical, allow green colours, green > red & green > blue, Default: T
blues,	logical, allow blue colours, blue > green & green > red, Default: T

`dark,`	logical, allow colours of dark intensity, sum(red, green, blue) < 420 , Default: T
`medium,`	logical, allow colours of medium intensity, between(sum(red, green, blue), 420, 600) , Default: T
`bright,`	logical, allow colours of bright intensity, sum(red, green, blue) > 600, Default: T
`thresh_similar,`	int, threshold for defining similar colours, see similar, Default: 25

Value

vector with hex colors

Examples

```
require(magrittr)

palette_qualitative() %>%
  palette_filter(thresh_similar = 0) %>%
  palette_plot_intensity()

## Not run:
# more examples---------------------------

palette_qualitative() %>%
  palette_filter(thresh_similar = 25) %>%
  palette_plot_intensity()

palette_qualitative() %>%
  palette_filter(thresh_similar = 0, blues = FALSE) %>%
  palette_plot_intensity()

## End(Not run)
```

`palette_increase_length`   *increases length of palette by repeating colours*

Description

works for any vector

Usage

```
palette_increase_length(palette = palette_qualitative(), n = 100)
```

Arguments

`palette`	any vector, Default: palette_qualitative()
`n,`	int, length, Default: 100

Value

vector with increased length

Examples

```
require(magrittr)

length(palette_qualitative())

palette_qualitative() %>%
  palette_increase_length(100) %>%
  length()
```

`palette_plot_intensity`

plot colour intensity of palette

Description

sum of red green and blue values

Usage

```
palette_plot_intensity(palette)
```

Arguments

`palette`	any vector containing color hex values

Value

ggplot2 plot

See Also

`palette_plot_rgp`

Examples

```
## Not run:
if(interactive()){
palette_qualitative() %>%
  palette_filter( thresh = 25) %>%
  palette_plot_intensity()
 }

## End(Not run)
```

---

palette_plot_rgp *plot rgb values of palette*

Description

grouped bar chart

Usage

```
palette_plot_rgp(palette)
```

Arguments

palette	any vector containing color hex values

Value

ggplot2 plot

See Also

palette_plot_intensity

Examples

```
## Not run:
if(interactive()){
palette_qualitative() %>%
  palette_filter( thresh = 50) %>%
  palette_plot_rgp()
 }

## End(Not run)
```

`palette_qualitative` *compose palette from qualitative RColorBrewer palettes*

Description

combines all unique values found in all qualitative RColorBrewer palettes

Usage

```
palette_qualitative()
```

Value

vector with hex values

See Also

RColorBrewer

Examples

```
palette_qualitative()
```

`plot_condensation` *Plot dataframe condensation potential*

Description

plotting the condensation potential is meant as a decision aid for which variables to include in an alluvial plot. All variables are transformed to categoric variables and then two variables are selected by which the dataframe will be grouped and summarized by. The pair that results in the greatest condensation of the original dataframe is selected. Then the next variable which offers the greatest condensation potential is chosen until all variables have been added. The condensation in percent is then plotted for each step along with the number of groups (flows) in the dataframe. By experience it is not advisable to have more than 1500 flows because then the alluvial plot will take a long time to render. If there is a particular variable of interest in the dataframe this variable can be chosen as a starting variable.

Usage

```
plot_condensation(df, first = NULL)
```

Arguments

df	dataframe
first	unquoted expression or string denoting the first variable to be picked for condensation, Default: NULL

Value

ggplot2 plot

See Also

quosure reexports RColorBrewer

Examples

```
require(magrittr)
require(dplyr)

df = mtcars %>%
  mutate( cyl = as.factor(cyl)
         , gear = as.factor(gear)
         , vs = as.factor(vs)
         , am = as.factor(am))

plot_condensation(df)

plot_condensation(df, first = 'disp')
```

---

quarterly_flights *Quarterly mean arrival delay times for a set of 402 flights*

Description

Created from nycflights13::flights

Usage

```
quarterly_flights
```

Format

A data frame with 1608 rows and 6 variables

tailnum a unique identifier created from tailnum, origin, destination and carrier
carrier carrier code
origin origin code
dest destination code
qu quarter
mean_arr_delay average delay on arrival as either on_time or late

Source

nycflights13::flights

Index