Package ‘easyalluvial’

December 9, 2019

Title Generate Alluvial Plots with a Single Line of Code

Version 0.2.2

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 (>= 0.3.1)

RoxygenNote 7.0.0

Imports purrr , tidyr (>= 1.0.0) , dplyr , forcats , ggalluvial (>= 0.9.1) , ggplot2 (>= 3.2.0) , ggridges , RColorBrewer , recipes (>= 0.1.5) , rlang , stringr , magrittr , tibble , caret , progress , gridExtra , randomForest , e1071

Language en-US

NeedsCompilation no

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

Maintainer Bjoern Koneswarakantha <datistics@gmail.com>
add_imp_plot

Description

adds bar plot of important features to model response alluvial plot

Usage

add_imp_plot(grid, p = NULL, data_input, plot = T, ...)

R topics documented:

- add_imp_plot
- add_marginal_histograms
- alluvial_long
- alluvial_model_response
- alluvial_model_response_caret
- alluvial_wide
- get_data_space
- get_pdp_predictions
- manip_bin_numerics
- manip_factor_2_numeric
- mtcars2
- palette_filter
- palette_increase_length
- palette_plot_intensity
- palette_plot_rgp
- palette_qualitative
- plot_all_hists
- plot_condensation
- plot_hist
- plot_imp
- quarterly_flights
- quarterly_sunssets
- tidy_imp
- titanic
- use_e1071
add_marginal_histograms

add marginal histograms to alluvial plot

Description

will add density histograms and frequency plots of original data to alluvial plot

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid</td>
<td>gtable or ggplot</td>
</tr>
<tr>
<td>p</td>
<td>alluvial plot, optional if alluvial plot has already been passed as grid. Default: NULL</td>
</tr>
<tr>
<td>data_input</td>
<td>dataframe used to generate alluvial plot</td>
</tr>
<tr>
<td>plot</td>
<td>logical if plot should be drawn or not</td>
</tr>
<tr>
<td>...</td>
<td>additional parameters passed to plot_imp</td>
</tr>
</tbody>
</table>

Value

gtable

See Also

arrangeGrob plot_imp

Examples

```r
## Not run:
library(ggplot2)
library(alluvial)

df = mtcars2[, ! names(mtcars2)]
train = caret::train( disp ~ . , df , method = 'rf' , trControl = caret::trainControl( method = 'none' ) , importance = TRUE )
pred_train = caret::predict.train(train, df)
p = alluvial_model_response_caret(train, degree = 4, pred_train = pred_train)
p_grid = add_marginal_histograms(p, data_input = df)
p_grid = add_imp_plot(p_grid, p, data_input = df)
## End(Not run)
```
add_marginal_histograms

Usage

add_marginal_histograms(
  p,
  data_input,
  top = TRUE,
  keep_labels = FALSE,
  plot = TRUE,
  ...
)

Arguments

p alluvial plot
data_input dataframe, input data that was used to create dataframe
top logical, position of histograms, if FALSE adds them at the bottom, Default: TRUE
keep_labels logical, keep title and caption, Default: FALSE
plot logical if plot should be drawn or not
... additional arguments for model response alluvial plot concerning the response variable
  pred_train display training prediction, not necessary if pred_train has already been passed to alluvial_model_response()
  scale int, y-axis distance between the ridge plots, Default: 400
  pred_var character vector, specify response variable in data_input, if not set response variable will try to be inferred, Default: NULL

Value
gtable

See Also

arrangeGrob

Examples

p = alluvial_wide(mtcars2, max_variables = 4)
p_grid = add_marginal_histograms(p, mtcars2)
alluvial_long

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

```r
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_label_size = 4.5,
  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
alluvial_long

- **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_label_size**: numeric, Default: 4.5
- **stratum_width**: double, Default: 1/4
- **auto_rotate_xlabs**: logical, Default: TRUE
- **...**: additional parameter passed to manip_bin_numerics

**Value**

- ggplot2 object

**See Also**

- alluvial_wide.geom_flow, geom_stratum, manip_bin_numerics

**Examples**

```r
data = quarterly_flights

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

```R
### 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_model_response

create model response plot

Description

Alluvial plots are capable of displaying higher dimensional data on a plane, thus lend themselves to plot the response of a statistical model to changes in the input data across multiple dimensions. The practical limit here is 4 dimensions. We need the data space (a sensible range of data calculated...
based on the importance of the explanatory variables of the model as created by `get_data_space` and the predictions returned by the model in response to the data space.

**Usage**

```r
alluvial_model_response(
  pred,
  dspace,
  imp,
  degree = 4,
  bins = 5,
  bin_labels = c("LL", "ML", "M", "MH", "HH"),
  col_vector_flow = c("#FF0065", "#009850", "#A56F2B", "#005EAA", "#710500", "#7B5380", 
                      "#9DD1D1"),
  method = "median",
  force = FALSE,
  params_bin_numeric_pred = list(center = T, transform = T, scale = T),
  pred_train = NULL,
  stratum_label_size = 3.5,
  ...
)
```

**Arguments**

- `pred` vector, predictions, if method = ‘pdp’ use `get_pdp_predictions` to calculate predictions
- `dspace` data frame, returned by `get_data_space`
- `imp` dataframe, with not more then two columns one of them numeric containing importance measures and one character or factor column containing corresponding variable names as found in training data.
- `degree` integer, number of top important variables to select. For plotting more than 4 will result in too many flows and the alluvial plot will not be very readable, Default: 4
- `bins` integer, number of bins for numeric variables, increasing this number might result in too many flows, Default: 5
- `bin_labels` labels for the bins from low to high, Default: c("LL", "ML", "M", "MH", "HH")
                     "#005EAA", "#710500")
- `method`, character vector, one of c(‘median’, ‘pdp’)
  - `median` sets variables that are not displayed to median mode, use with regular predictions
  - `pdp` partial dependency plot method, for each observation in the training data the displayed variable is set to the indicated values. The predict function is called for each modified observation and the result is averaged, calculate predictions using `get_pdp_predictions`
alluvial_model_response

- Default: 'median'

force

logical, force plotting of over 1500 flows, Default: FALSE

params_bin_numeric_pred

list, additional parameters passed to `manip_bin_numerics` which is applied to the pred parameter. Default: list( bins = 5, center = T, transform = T, scale = T)

pred_train

numeric vector, base the automated binning of the pred vector on the distribution of the training predictions. This is useful if marginal histograms are added to the plot later. Default = NULL

stratum_label_size

numeric, Default: 3.5

... additional parameters passed to `alluvial_wide`

Details

this model visualisation approach follows the "visualising the model in the dataspace" principle as described in Wickham H, Cook D, Hofmann H (2015) Visualizing statistical models: Removing the blindfold. Statistical Analysis and Data Mining 8(4) <doi:10.1002/sam.11271>

Value

ggplot2 object

See Also

alluvial_wide, get_data_space, alluvial_model_response_caret

Examples

df = mtcars2[, ! names(mtcars2) %in% 'ids']
m = randomForest::randomForest( disp ~ ., df)
imp = m$importance
dspace = get_data_space(df, imp, degree = 3)
pred = predict(m, newdata = dspace)
alluvial_model_response(pred, dspace, imp, degree = 3)

# partial dependency plotting method
## Not run:
pred = get_pdp_predictions(df, imp
  , .f_predict = randomForest::predict.randomForest
    , m
  , degree = 3
  , bins = 5)

  alluvial_model_response(pred, dspace, imp, degree = 3, method = 'pdp')

## End(Not run)
alluvial_model_response_caret

create model response plot for caret models

Description

Wraps `alluvial_model_response` and `get_data_space` into one call for caret models.

Usage

```r
alluvial_model_response_caret(
  train,
  degree = 4,
  bins = 5,
  bin_labels = c("LL", "ML", "M", "MH", "HH"),
  col_vector_flow = c("#FF0065", "#009850", "#A56F2B", "#005EAA", "#710500", "#7B5380", "#9DD1D1"),
  method = "median",
  params_bin_numeric_pred = list(center = T, transform = T, scale = T),
  pred_train = NULL,
  stratum_label_size = 3.5,
  force = F,
  ...
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>train</code></td>
<td>caret train object</td>
</tr>
<tr>
<td><code>degree</code></td>
<td>integer, number of top important variables to select. For plotting more than 4 will result in too many flows and the alluvial plot will not be very readable, Default: 4</td>
</tr>
<tr>
<td><code>bins</code></td>
<td>integer, number of bins for numeric variables, increasing this number might result in too many flows, Default: 5</td>
</tr>
<tr>
<td><code>bin_labels</code></td>
<td>labels for the bins from low to high, Default: c(&quot;LL&quot;, &quot;ML&quot;, &quot;M&quot;, &quot;MH&quot;, &quot;HH&quot;)</td>
</tr>
<tr>
<td><code>col_vector_flow</code></td>
<td>character vector, defines flow colours, Default: c(&quot;#FF0065&quot;, &quot;#009850&quot;, &quot;#A56F2B&quot;, &quot;#005EAA&quot;, &quot;#710500&quot;)</td>
</tr>
<tr>
<td><code>method</code></td>
<td>character vector, one of c(&quot;median&quot;, &quot;pdp&quot;)</td>
</tr>
<tr>
<td><code>params_bin_numeric_pred</code></td>
<td>list(center = T, transform = T, scale = T)</td>
</tr>
<tr>
<td><code>pred_train</code></td>
<td>partial dependency plot method, for each observation in the training data the displayed variable are set to the indicated values. The predict function is called for each modified observation and the result is averaged. Default: &quot;median&quot;</td>
</tr>
<tr>
<td><code>stratum_label_size</code></td>
<td>3.5</td>
</tr>
<tr>
<td><code>force</code></td>
<td>logical, if <code>TRUE</code> force alluvial plot even if some of the displayed variables are not numeric or not numeric valued</td>
</tr>
</tbody>
</table>

Examples

```r
alluvial_model_response_caret(
  train = ...,
  degree = 4,
  bins = 5,
  bin_labels = c("LL", "ML", "M", "MH", "HH"),
  col_vector_flow = c("#FF0065", "#009850", "#A56F2B", "#005EAA", "#710500", "#7B5380", "#9DD1D1"),
  method = "median",
  params_bin_numeric_pred = list(center = T, transform = T, scale = T),
  pred_train = NULL,
  stratum_label_size = 3.5,
  force = F,
  ...
)
```
params_bin_numeric_pred

list, additional parameters passed to `manip_bin_numerics` which is applied to the `pred` parameter. Default: `list(bins = 5, center = T, transform = T, scale = T)`

pred_train

numeric vector, base the automated binning of the `pred` vector on the distribution of the training predictions. This is useful if marginal histograms are added to the plot later. Default = NULL

stratum_label_size

numeric, Default: 3.5

force

logical, force plotting of over 1500 flows, Default: FALSE

 additional parameters passed to `alluvial_wide`

Details

this model visualisation approach follows the "visualising the model in the dataspace" principle as described in Wickham H, Cook D, Hofmann H (2015) Visualizing statistical models: Removing the blindfold. Statistical Analysis and Data Mining 8(4) <doi:10.1002/sam.11271>

Value

ggplot2 object

See Also

`alluvial_wide, get_data_space, varImp, extractPrediction, get_data_space, get_pdp_predictions`

Examples

df = mtcars2[, ! names(mtcars2) %in% 'ids' ]

train = caret::train( disp ~ .
   , df
   , method = 'rf'
   , trControl = caret::trainControl( method = 'none' )
   , importance = TRUE )

alluvial_model_response_caret(train, degree = 3)

# partial dependency plotting method
## Not run:
alluvial_model_response_caret(train, degree = 3, method = 'pdp')

## 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

```r
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)],
  colorful_fill_variable_stratum = T,
  verbose = F,
  stratum_labels = T,
  stratum_label_size = 4.5,
  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)
alluvial_wide

col_vector_value
Hex colors for y levels/values, Default: RColorBrewer::brewer.pal(9, "Greys")[c(3, 6, 4, 7, 5)]

colorful_fill_variable_stratum
logical, use flow colors to colorize fill variable stratum, Default: TRUE

verbose
logical, print plot summary, Default: F

stratum_labels
logical, Default: TRUE

stratum_label_size
numeric, Default: 4.5

stratum_width
double, Default: 1/4

auto_rotate_xlabs
logical, Default: TRUE

... additional arguments passed to manip_bin_numerics

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, manip_bin_numerics

Examples

alluvial_wide( data = mtcars2, id = ids
, max_variables = 5
, fill_by = 'first_variable' )

## Not run:

# more coloring variants-----------------------
alluvial_wide( data = mtcars2, id = ids
, max_variables = 5
, fill_by = 'last_variable' )

alluvial_wide( data = mtcars2, id = ids
, max_variables = 5
, fill_by = 'all_flows' )

alluvial_wide( data = mtcars2, id = ids
, max_variables = 5
, fill_by = 'first_variable' )
# manually order variable values and colour by stratum value

```r
alluvial_wide( data = mtcars2, id = ids
  , max_variables = 5
  , fill_by = 'values'
  , order_levels = c('4', '8', '6')
)
```

## End(Not run)

---

**get_data_space**

**calculate data space**

### Description

calculates a dataspace based on the modelling dataframe and the importance of the explanatory variables. It only considers the most important variables as defined by the degree parameter. It selects a number (defined by bins) of sensible single values spread over the range of the numeric variables and creates all possible value combinations among the most important variables. The values of the remaining variables are set to mode(factors) or median(numerics).

### Usage

```r
get_data_space(df, imp, degree = 4, bins = 5, max_levels = 10)
```

### Arguments

- **df**: dataframe, training data
- **imp**: dataframe, with not more than two columns, one of them numeric containing importance measures and one character or factor column containing corresponding variable names as found in training data.
- **degree**: integer, number of top important variables to select. For plotting more than 4 will result in too many flows and the alluvial plot will not be very readable, Default: 4
- **bins**: integer, number of bins for numeric variables, and maximum number of levels for factor variables, increasing this number might result in too many flows, Default: 5
- **max_levels**: integer, maximum number of levels per factor variable, Default: 10

### Details

It selects the top most important variables based on the degree parameter and bins the numeric variables using `manip_bin_numerics`, while leaving categoric variables unchanged. The number of bins for each numeric variable is set to bins -2. Next the median is picked for each of the bins and the min and the max value is added for each numeric variable so that we get median(bin) X bins -2, max, min for each numeric variable. Then all possible combinations between those values and the categoric factor levels are created. The total number of all possible combinations defines
get_pdp_predictions

the range of the data space. The values of the remaining variables are set to mode(factors) or median(numerics).

This model visualisation approach follows the "visualising the model in the dataspace" principle as described in Wickham H, Cook D, Hofmann H (2015) Visualizing statistical models: Removing the blindfold. Statistical Analysis and Data Mining 8(4) <doi:10.1002/sam.11271>

Value

data frame

See Also

alluvial_wide, manip_bin_numerics

Examples

df = mtcars2[, ! names(mtcars2) %in% 'ids' ]
m = randomForest::randomForest( disp ~ ., df)
imp = m$importance
dspace = get_data_space(df, imp)

generate_predictions <- function(df, imp, m, degree = 4, bins = 5, .f_predict = predict)

generate_predictions(df, imp, m, degree = 4, bins = 5, .f_predict = predict)

Description

Alluvial plots are capable of displaying higher dimensional data on a plane, thus lend themselves to plot the response of a statistical model to changes in the input data across multiple dimensions. The practical limit here is 4 dimensions while conventional partial dependence plots are limited to 2 dimensions.

Briefly the 4 variables with the highest feature importance for a given model are selected and 5 values spread over the variable range are selected for each. Then a grid of all possible combinations is created. All none-plotted variables are set to the values found in the first row of the training data set. Using this artificial data space model predictions are being generated. This process is then repeated for each row in the training data set and the overall model response is averaged in the end. Each of the possible combinations is plotted as a flow which is coloured by the bin corresponding to the average model response generated by that particular combination.

Usage

generate_predictions(df, imp, m, degree = 4, bins = 5, .f_predict = predict)
manip_bin_numerics

Arguments

- **df**: dataframe, training data
- **imp**: dataframe, with not more than two columns one of them numeric containing importance measures and one character or factor column containing corresponding variable names as found in training data.
- **m**: model object
- **degree**: integer, number of top important variables to select. For plotting more than 4 will result in too many flows and the alluvial plot will not be very readable, Default: 4
- **bins**: integer, number of bins for numeric variables, increasing this number might result in too many flows, Default: 5
- **.f_predict**: corresponding model predict() function. Needs to accept 'm' as the first parameter and use the 'newdata' parameter. Supply a wrapper for predict functions with x-y syntax.

Details

see https://christophm.github.io/interpretable-ml-book/pdp.html

Value

vector, predictions

See Also

- progress_bar

Examples

```r
df = mtcars[, !names(mtcars) %in% 'ids']
m = randomForest::randomForest( disp ~ ., df)
imp = m$importance

pred = get_pdp_predictions(df, imp, m
, degree = 3
, bins = 5)
```

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

```r
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,
  NA_label = "NA"
)
```

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 corresponding 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
- `NA_label`: character vector, define label for missing data, Default: 'NA'

Value

dataframe

Examples

```r
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 )

mtcars2

mtcars dataset with cyl, vs, am ,gear, carb as factor variables and car model names as id

Description

mtcars dataset with cyl, vs, am ,gear, carb as factor variables and car model names as id

Usage

mtcars2
Format

A data frame with 32 rows and 12 variables

mpg  Miles/(US) gallon

cyl  Number of cylinders

disp  Displacement (cu.in.)

hp  Gross horsepower

drat  Rear axle ratio

wt  Weight (1000 lbs)

qsec  1/4 mile time

vs  Engine

am  Transmission

gear  Number of forward gears

carb  Number of carburetors

ids  car model name

Source

datasets

describe(palette_filter)

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

```r
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

```r
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

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

---

describe

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
```

---
palette_qualitative

Examples

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

## End(Not run)
```

description

compose palette from qualitative RColorBrewer palettes

Usage

```
palette_qualitative()
```

Value

vector with hex values

See Also

```
RColorBrewer
```

Examples

```
palette_qualitative()
```

plot_all_hists

plot marginal histograms of alluvial plot

Description

will create gtable with density histograms and frequency plots of all variables of a given alluvial plot.

Usage

```
plot_all_hists(p, data_input, top = TRUE, keep_labels = FALSE, ...)
```
plot_condensation

Arguments

p alluvial plot
data_input dataframe, input data that was used to create dataframe
top logical, position of histograms, if FALSE adds them at the bottom, Default: TRUE
keep_labels logical, keep title and caption, Default: FALSE
... additional arguments for specific alluvial plot types: pred_train can be used to pass training predictions for model response alluvials

Value
gtable

See Also

arrangeGrob
add_marginal_histograms

Examples

p = alluvial_wide(mtcars2, max_variables = 4)
plot_all_hists(p, mtcars2)

plot_condensation(df, first = NULL)

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
plot_hist

Value

ggplot2 plot

See Also

quosure reexports RColorBrewer

Examples

plot_condensation(mtcars2)
plot_condensation(mtcars2, first = 'disp')

plot_hist

plot histogram of alluvial plot variable

Description

helper function used by add_marginal_histograms

Usage

plot_hist(var, p, data_input, ...)

Arguments

var character vector, variable name
p alluvial plot
data_input datafram used to create alluvial plot
... additional arguments for specific alluvial plot types: pred_train can be used to pass training predictions for model response alluvials

Value

ggplot object
plot_imp

plot feature importance

Description

plot important features of model response alluvial as bars

Usage

plot_imp(p, data_input, truncate_at = 50, color = "darkgrey")

Arguments

- `p`: alluvial plot
- `data_input`: dataframe used to generate alluvial plot
- `truncate_at`: integer, limit number of features to that value, Default: 50
- `color`: character vector, Default: 'darkgrey'

Value

ggplot object

Examples

```r
## Not run:
df = mtcars2[, !names(mtcars2)]

train = caret::train( disp ~ .,
  df
  , method = 'rf'
  , trControl = caret::trainControl( method = 'none'
    , importance = TRUE )

pred_train = caret::predict.train(train, df)

p = alluvial_model_response_caret(train, degree = 3, pred_train = pred_train)

plot_imp(p, mtcars2)

## End(Not run)
```
**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

---

**quarterly_sunspots**  
*Quarterly mean relative sunspots number from 1749-1983*

**Description**

Quarterly mean relative sunspots number from 1749-1983

**Usage**

`quarterly_sunspots`

**Format**

A data frame with 940 rows and 4 variables

- **year**
- **qu**  quarter
- **spots**  total number of sunspots
- **mean_spots_per_year**
tidy_imp

*tidy up dataframe containing model feature importance*

**Description**

returns dataframe with exactly two columns, vars and imp and aggregates dummy encoded variables. Helper function called by all functions that take an imp parameter. Can be called manually if formula for aggregating dummy encoded variables must be modified.

**Usage**

```
tidy_imp(imp, df, .f = max)
```

**Arguments**

- **imp** 
  dataframe or matrix with feature importance information
- **df** 
  dataframe, modelling training data
- **.f** 
  window function, Default: max

**Value**

dataframe

- **vars** character column with feature names
- **imp** numerical column, importance values

**Examples**

```
# randomforest
df = mtcars2[, !names(mtcars2) %in% 'ids']
m = randomForest::randomForest( disp ~ ., df)
imp = m$importance
tidy_imp(imp, mtcars2)
```
titanic
titanic data set'

Description
titanic data set'

Usage
titanic

Format
A data frame with 891 rows and 10 variables

Survived  Survived
Pclass   Pclass
Sex      Sex
Age      Age
SibSp    SibSp
Parch    Parch
Fare     Fare
Cabin    Cabin
Embarked Embarked
title    title

Source
datasets
calls e1071::skewness

Description
if e1071 is not listed a a dependency I get an error. I assume caret uses it to calculate feature
importance. However, e1071 is not listed as a caret dependency. I have to add a function that
directly calls it, so I do not get a NOTE from RMD Check on Linux.

Usage
use_e1071(x)
Arguments

x PARAM_DESCRIPTION

Value

OUTPUT_DESCRIPTION

See Also

skewness
Index

*Topic datasets
  mtcars2, 18
  quarterly_flights, 27
  quarterly_sunspots, 27
  titanic, 29

add_imp_plot, 2
add_marginal_histograms, 3, 24
alluvial_long, 5
alluvial_model_response, 7, 10
alluvial_model_response_caret, 9, 10
alluvial_wide, 6, 9, 11, 12, 13, 15
arrangeGrob, 3, 4, 24

extractPrediction, 11

geom_flow, 6, 13
geom_stratum, 6, 13
get_data_space, 8–11, 14
get_pdp_predictions, 8, 11, 15

manip_bin_numerics, 6, 9, 11, 13–15, 16
manip_factor_2_numeric, 18
mtcars2, 18

palette_filter, 19
palette_increase_length, 21
palette_plot_intensity, 21, 22
palette_plot_rgb, 22, 22
palette_qualitative, 23
plot_all_hists, 23
plot_condensation, 24
plot_hist, 25
plot_imp, 3, 26
progress_bar, 16

quarterly_flights, 27
quarterly_sunspots, 27
quosure, 25

RColorBrewer, 23, 25

reexports, 25
skewness, 30
str_detect, 18
tidy_imp, 28
titanic, 29
use_e1071, 29
varImp, 11

31