Package 'lares'

May 18, 2023

Type Package

Title Analytics & Machine Learning Sidekick

Version 5.2.2

Maintainer Bernardo Lares <laresbernardo@gmail.com>

Description Auxiliary package for better/faster analytics, visualization, data mining, and machine learning tasks. With a wide variety of family functions, like Machine Learning, Data Wrangling, Exploratory, API, and Scrapper, it helps the analyst or data scientist to get quick and robust results, without the need of repetitive coding or extensive R programming skills.

Depends R (>= 3.5)

Imports dplyr, ggplot2, h2o, httr, jsonlite, lubridate, openxlsx, patchwork, pROC, rlang, rpart, rpart.plot, rvest, stringr, tidyr, yaml

Suggests beepr, DALEX, googleAuthR, googlesheets4, knitr, quantmod, rdrop2, rmarkdown

URL https://github.com/laresbernardo/lares, https://laresbernardo.github.io/lares/

BugReports https://github.com/laresbernardo/lares/issues

VignetteBuilder knitr

RoxygenNote 7.2.3

License AGPL-3

Encoding UTF-8

LazyData true

NeedsCompilation no

Author Bernardo Lares [aut, cre]

Repository CRAN

Date/Publication 2023-05-18 10:40:02 UTC

R topics documented:

autoline *New Line Feed for Long Strings (Wrapper)*

Description

Add a break or new line without breaking words. Automatically, the function can detect your plot's width and will dynamically set an auto width. You can adjust the relation (rel) parameter for different fonts and sizes until perfect harmony found. Quite similar to `stringr::str_wrap` but, if the text vector is a factor, the levels will be kept in order and transformed.

Usage

```
autoline(text, top = "auto", rel = 9)
```

Arguments

`text`	Character or factor vector.
`top`	Integer. How many characters aprox. should be on each line?
`rel`	Numeric. Relation of pixels and characters per line

Value

Character. String (vector) including some \n within.

See Also

Other Tools: `bind_files()`, `bring_api()`, `db_download()`, `db_upload()`, `export_plot()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

Examples

```
cat(autoline("This is a long text that may not fit into a single line", 8))

text <- factor(c("First value", "Second value", "First value"),
  levels = c("First value", "Second value")
)
autoline(text, 1)

path <- file.path(R.home("doc"), "THANKS")
text <- paste(readLines(path), collapse = " ")
cat(autoline(text))
```

balance_data *Balance Binary Data by Resampling: Under-Over Sampling*

Description

This function lets the user balance a given data.frame by resampling with a given relation rate and a binary feature.

Usage

```
balance_data(df, variable, rate = 1, target = "auto", seed = 0, quiet = FALSE)
```

Arguments

df	Vector or Dataframe. Contains different variables in each column, separated by a specific character
variable	Variable. Which variable should we used to re-sample dataset?
rate	Numeric. How many X for every Y we need? Default: 1. If there are more than 2 unique values, rate will represent percentage for number of rows
target	Character. If binary, which value should be reduced? If kept in "auto", then the most frequent value will be reduced.
seed	Numeric. Seed to replicate and obtain same values
quiet	Boolean. Keep quiet? If not, messages will be printed

Value

data.frame. Reduced sampled data.frame following the rate of appearance of a specific variable.

See Also

Other Data Wrangling: categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
data(dft) # Titanic dataset
df <- balance_data(dft, Survived, rate = 0.5)
df <- balance_data(dft, .data$Survived, rate = 0.1, target = "TRUE")
```

bind_files *Bind Files into Dataframe*

Description

This function imports and binds multiple files into a single data.frame. Files must be inserted with absolute roots files names.

Usage

```
bind_files(files)
```

Arguments

files	Character vector. Files names.

Value

data.frame with data joined from all `files` passed.

See Also

Other Tools: autoline(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

bring_api *Get API (JSON) and Transform into data.frame*

Description

This function lets the user bring API data as JSON format and transform it into data.frame.

Usage

```
bring_api(url, status = TRUE)
```

Arguments

url	Character. API's URL to GET.
status	Boolean. Display status message?

Value

data.frame of `url GET` results or `NULL` if no results returned by API.

See Also

Other Tools: `autoline()`, `bind_files()`, `db_download()`, `db_upload()`, `export_plot()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

Other API: `fb_accounts()`, `fb_ads()`, `fb_creatives()`, `fb_insights()`, `fb_process()`, `fb_report_check()`, `fb_rf()`, `fb_token()`, `gpt_ask()`, `li_auth()`, `li_profile()`, `queryGA()`, `slackSend()`

---

cache_write *Cache Save and Load (Write and Read)*

Description

This function lets the user save and load a cache of any R object to improve timings and UX.

Usage

```
cache_write(
  data,
  base = "temp",
  cache_dir = getOption("LARES_CACHE_DIR"),
  ask = FALSE,
  quiet = FALSE,
  ...
)

cache_read(
  base,
  cache_dir = getOption("LARES_CACHE_DIR"),
  ask = FALSE,
  quiet = FALSE,
  ...
)

cache_exists(base = NULL, cache_dir = getOption("LARES_CACHE_DIR"), ...)

cache_clear(cache_dir = getOption("LARES_CACHE_DIR"), quiet = FALSE, ...)
```

Arguments

data	Object
base	Character vector. Unique name for your cache file. You can pass a character vector with multiple elements that will be concatenated. All cache files with start with lares_cache_* automatically to quickly detect these cache files.
cache_dir	Character. Where do you want to save you cache files? By default they'll be stored on tempdir() but you can change it using this parameter or setting a global option called "LARES_CACHE_DIR".
ask	Boolean. If cache exists, when reading: (interactive) ask the user if the cache should be used to proceed or ignored; when writing, (interactive) ask the user if the cache should be overwritten. Note that you can only ask for one cache file at a time because vectors are concatenated.
quiet	Boolean. Keep quiet? If not, message will be shown.
...	Additional parameters.

Value

cache_write. No return value, called for side effects.

cache_read. R object. Data from cache file or NULL if no cache found.

cache_exists. Boolean. Result of base existence.

cache_clear. Invisible vector containing cache file names removed.

Examples

```
x <- list(a = 1, b = 2:4)
base <- c(as.character(Sys.Date()), "A", "B")
cache_write(x, base)
cache_read(base, ask = FALSE)
cache_exists("lares_cache_2021-06-01.A.B.C")
cache_clear()
```

---

categ_reducer *Reduce categorical values*

Description

This function lets the user reduce categorical values in a vector. It is tidyverse friendly for use on pipelines

Usage

```
categ_reducer(
  df,
  var,
  nmin = 0,
  pmin = 0,
  pcummax = 100,
  top = NA,
  pvalue_max = 1,
  cor_var = "tag",
  limit = 20,
  other_label = "other",
  ...
)
```

Arguments

df	Categorical Vector
var	Variable. Which variable do you wish to reduce?
nmin	Integer. Number of minimum times a value is repeated
pmin	Numerical. Percentage of minimum times a value is repeated
pcummax	Numerical. Top cumulative percentage of most repeated values
top	Integer. Keep the n most frequently repeated values
pvalue_max	Numeric (0-1]. Max pvalue categories
cor_var	Character. If pvalue_max < 1, you must define which column name will be compared with (numerical or binary).
limit	Integer. Limit one hot encoding to the n most frequent values of each column. Set to NA to ignore argument.
other_label	Character. With which text do you wish to replace the filtered values with?
...	Additional parameters

Value

data.frame df on which var has been transformed

See Also

Other Data Wrangling: balance_data(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
data(dft) # Titanic dataset
categ_reducer(dft, Embarked, top = 2) %>% freqs(Embarked)
categ_reducer(dft, Ticket, nmin = 7, other_label = "Other Ticket") %>% freqs(Ticket)
categ_reducer(dft, Ticket, pvalue_max = 0.05, cor_var = "Survived") %>% freqs(Ticket)
```

---

check_opts *Validate inputs (attributions, options, ...)*

Description

This function validates if inputs match all/any of your options and return error/message with possible options to use. Similar to `match.arg()` but more flexible.

This function checks if an object has a specific attribute and stops if not.

Usage

```
check_opts(
  inputs,
  opts,
  input_name = "input",
  type = "all",
  not = "stop",
  quiet = TRUE
)

check_attr(object, attr = "type", check = NULL, stop = TRUE)
```

Arguments

`inputs`	Vector character. Check options.
`opts`	Vector character. Valid options.
`input_name`	Character. Custom your message and change "input" for any other string. For example: "column names".
`type`	Character. Options: "all", "any."
`not`	Character. Options: "stop", "message", "print", "return".
`quiet`	Boolean. Keep quiet? If not, returns logical value.
`object`	Object of any kind
`attr`	Character. Attribute to check
`check`	Character. Attribute value
`stop`	Boolean. Stop if doesn't check?

Value

Boolean. Result of `inputs` in `opts` (options). Depending on `type` and/or `stop` arguments, errors or messages will be shown.

No return value, called for side effects.

Examples

```
opts <- c("A", "B", "C")
# Let's check the "all" logic
check_opts(inputs = c("A", "B"), opts, quiet = FALSE)
check_opts(inputs = c("X"), opts, not = "message", quiet = FALSE)
check_opts(inputs = c("A", "X"), opts, input_name = "value", not = "warning")
# Now let's check the "any" logic
check_opts(inputs = c("A", "X"), opts, type = "any")
check_opts(inputs = c("X"), opts, type = "any", not = "message")
check_opts(inputs = c("A", NA), opts, type = "any")
# Final trick: just ignore results
check_opts(inputs = "X", opts, not = "invisible")
test <- data.frame()
attributes(test)
check_attr(test, "class", "data.frame")
# check_attr(test, "class", "not.data.frame")
```

---

ci_lower *Lower/Upper Confidence Intervals*

Description

Calculate lower and upper confidence intervals given a mean, standard deviation, sample size, and confidence level. You may want to use `ci_var()` to calculate all values quickly.

Usage

```
ci_lower(mean, ssd, n, conf = 0.95)

ci_upper(mean, ssd, n, conf = 0.95)
```

Arguments

`mean`	Numeric. Mean: `mean(var, na.rm = TRUE)`
`ssd`	Numeric. Standard deviation: `sd(var, na.rm = TRUE)`
`n`	Integer. Amount of observations: `n()`
`conf`	Numeric (0-1). Confidence level.

Value

Vector with confidence limit value.

See Also

Other Confidence: ci_var()

Examples

```
ci_lower(100, 5, 10)
ci_upper(100, 5, 10)
```

---

ci_var *Confidence Intervals on Dataframe*

Description

Calculate confidence intervals for a continuous numerical column on a dataframe, given a confidence level. You may also group results using another variable. Tidyverse friendly.

Usage

```
ci_var(df, var, group_var = NULL, conf = 0.95)
```

Arguments

df	Dataframe
var	Variable name. Must be a numerical column.
group_var	Variable name. Group results by another variable.
conf	Numeric. Confidence level (0-1).

Value

data.frame mean, standard deviation, counter, upper and lower CIs.

See Also

Other Confidence: ci_lower()

Examples

```
data(dft) # Titanic dataset
ci_var(dft, Fare)
ci_var(dft, Fare, Pclass)
ci_var(dft, Fare, Pclass, conf = 0.99)
```

cleanText *Clean text strings automatically*

Description

cleanText: Clean character strings automatically. Options to keep ASCII characters only, keep certain characters, lower caps, title format, are available.

cleanNames: Resulting names are unique and consist only of the _ character, numbers, and ASCII letters. Capitalization preferences can be specified using the lower parameter.

Usage

```
cleanText(
  text,
  spaces = TRUE,
  keep = "",
  lower = TRUE,
  ascii = TRUE,
  title = FALSE
)

cleanNames(df, num = "x", keep = "_", ...)
```

Arguments

text	Character Vector
spaces	Boolean. Keep spaces? If character input, spaces will be transformed into passed argument.
keep	Character. String (concatenated or as vector) with all characters that are accepted and should be kept, in addition to alphanumeric.
lower	Boolean. Transform all to lower case?
ascii	Boolean. Only ASCII characters?
title	Boolean. Transform to title format (upper case on first letters).
df	data.frame/tibble.
num	Add character before only-numeric names.
...	Additional parameters passed to cleanText().

Details

Inspired by janitor::clean_names.

Value

Character vector with transformed strings.

data.frame/tibble with transformed column names.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Text Mining: ngrams(), remove_stopwords(), replaceall(), sentimentBreakdown(), textCloud(), textFeats(), textTokenizer(), topics_rake()

Examples

```
cleanText("Bernardo Lares 123")
cleanText("Bèrnärdo LáreS 123", lower = FALSE)
cleanText("Bernardo Lare$", spaces = ".", ascii = FALSE)
cleanText("\\@@ì÷å   %ñS  ..-X", spaces = FALSE)
cleanText(c("maría", "€", "núñez_a."), title = TRUE)
cleanText("29_Feb-92()#", keep = c("#", "_"), spaces = FALSE)

# For a data.frame directly:
df <- dft[1:5, 1:6] # Dummy data
colnames(df) <- c("ID.", "34", "x_2", "Num 123", "Nòn-äscì", "  white   Spaces  ")
print(df)
cleanNames(df)
cleanNames(df, lower = FALSE)
```

---

clusterKmeans *Automated K-Means Clustering + PCA/t-SNE*

Description

This function lets the user cluster a whole data.frame automatically. As you might know, the goal of kmeans is to group data points into distinct non-overlapping subgroups. If needed, one hot encoding will be applied to categorical values automatically with this function. For consideration: Scale/standardize the data when applying kmeans. Also, kmeans assumes spherical shapes of clusters and does not work well when clusters are in different shapes such as elliptical clusters.

Usage

```
clusterKmeans(
  df,
  k = NULL,
  wss_var = 0,
  limit = 15,
  drop_na = TRUE,
  ignore = NULL,
  ohse = TRUE,
  norm = TRUE,
  algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"),
```

```
  dim_red = "PCA",
  comb = c(1, 2),
  seed = 123,
  quiet = FALSE,
  ...
)
```

Arguments

df	Dataframe
k	Integer. Number of clusters
wss_var	Numeric. Used to pick automatic k value, when k is NULL based on WSS variance while considering limit clusters. Values between (0, 1). Default value could be 0.05 to consider convergence.
limit	Integer. How many clusters should be considered?
drop_na	Boolean. Should NA rows be removed?
ignore	Character vector. Names of columns to ignore.
ohse	Boolean. Do you wish to automatically run one hot encoding to non-numerical columns?
norm	Boolean. Should the data be normalized?
algorithm	character: may be abbreviated. Note that "Lloyd" and "Forgy" are alternative names for one algorithm.
dim_red	Character. Select dimensionality reduction technique. Pass any of: c("PCA", "tSNE", "all", "none").
comb	Vector. Which columns do you wish to plot? Select which two variables by name or column position.
seed	Numeric. Seed for reproducibility
quiet	Boolean. Keep quiet? If not, print messages.
...	Additional parameters to pass sub-functions.

Value

List. If no k is provided, contains nclusters and nclusters_plot to determine optimal k given their WSS (Within Groups Sum of Squares). If k is provided, additionally we get:

- df data.frame with original df plus cluster column
- clusters integer which is the same as k
- fit kmeans object used to fit clusters
- means data.frame with means and counts for each cluster
- correlations plot with correlations grouped by clusters
- PCA list with PCA results (when dim_red="PCA")
- tSNE list with t-SNE results (when dim_red="tSNE")

See Also

Other Clusters: clusterOptimalK(), clusterVisualK(), reduce_pca(), reduce_tsne()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data("iris")
df <- subset(iris, select = c(-Species))

# If dataset has +5 columns, feel free to reduce dimenstionalities
# with reduce_pca() or reduce_tsne() first

# Find optimal k
check_k <- clusterKmeans(df, limit = 10)
check_k$nclusters_plot
# Or pick k automatically based on WSS variance
check_k <- clusterKmeans(df, wss_var = 0.05, limit = 10)
# You can also use our other functions:
# clusterOptimalK(df) and clusterVisualK(df)

# Run with selected k
clusters <- clusterKmeans(df, k = 3)
names(clusters)

# Cross-Correlations for each cluster
plot(clusters$correlations)

# PCA Results (when dim_red = "PCA")
plot(clusters$PCA$plot_explained)
plot(clusters$PCA$plot)
```

---

clusterOptimalK *Visualize K-Means Clusters for Several K Methods*

Description

Visualize cluster data for assorted values of k and methods such as WSS, Silhouette and Gap Statistic. See `factoextra::fviz_nbclust` for more.

Usage

```
clusterOptimalK(
  df,
  method = c("wss", "silhouette", "gap_stat"),
  drop_na = TRUE,
  ohse = TRUE,
  norm = TRUE,
  quiet = TRUE,
  ...
)
```

Arguments

df	Dataframe
method	Character vector.
drop_na	Boolean. Should NA rows be removed?
ohse	Boolean. Do you wish to automatically run one hot encoding to non-numerical columns?
norm	Boolean. Should the data be normalized?
quiet	Boolean. Keep quiet? If not, print messages.
...	Additional parameters passed to factoextra::fviz_nbclust

Value

Plot. Optimal number of clusters of df data.frame given a selected method.

See Also

Other Clusters: clusterKmeans(), clusterVisualK(), reduce_pca(), reduce_tsne()

Examples

```
# You must have "factoextra" library to use this auxiliary function:
## Not run:
data("iris")
df <- subset(iris, select = c(-Species))
# Calculate and plot optimal k clusters
clusterOptimalK(df)

## End(Not run)
```

---

clusterVisualK *Visualize K-Means Clusters for Several K*

Description

Visualize cluster data for assorted values of k.

Usage

```
clusterVisualK(df, ks = 2:6, ...)
```

Arguments

df	Dataframe
ks	Integer vector. Which k should be tested?
...	Additional parameters passed to clusterKmeans

Value

List. Plot and data.frame results of clustering `df` data.frame into `ks` integer clusters.

See Also

Other Clusters: `clusterKmeans()`, `clusterOptimalK()`, `reduce_pca()`, `reduce_tsne()`

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data("iris")
df <- subset(iris, select = c(-Species))
df <- df[sample(nrow(df)), ]

# Calculate and plot
result <- clusterVisualK(df, ks = 2:4)
plot(result$plot)

# You can use the data generated as well
lapply(result$data, function(x) head(x$cluster, 10))
```

---

conf_mat *Confussion Matrix*

Description

This function calculates a Confussion Matrix using crosstab for 2 or more categories. You can either set the score and threshold or the labels you wish to cross with.

Usage

```
conf_mat(tag, score, thresh = 0.5, sense = ">=", diagonal = TRUE, plot = FALSE)
```

Arguments

`tag`	Vector. Real known label
`score`	Vector. Predicted value or model's result
`thresh`	Integer. Threshold for selecting binary or regression models: this number is the threshold of unique values we should have in `'tag'` (more than: regression; less than: classification)
`sense`	Character. Inequation sense for threshold: <, <=, >=, >
`diagonal`	Boolean. `FALSE` to convert diagonal numbers to zeroes. Ideal to detect must confusing categories.
`plot`	Boolean. Plot result? Uses `mplot_conf()`

Details

You may use `mplot_conf()` or set `plot=TRUE`.

Value

data.frame. Result of counting `tag` and `score`'s tag given a `threshold`, similar to `base::table()`.

See Also

Other Machine Learning: `ROC()`, `export_results()`, `gain_lift()`, `h2o_automl()`, `h2o_predict_API()`, `h2o_predict_MOJO()`, `h2o_predict_binary()`, `h2o_predict_model()`, `h2o_selectmodel()`, `impute()`, `iter_seeds()`, `lasso_vars()`, `model_metrics()`, `model_preprocess()`, `msplit()`

Other Model metrics: `ROC()`, `errors()`, `gain_lift()`, `loglossBinary()`, `model_metrics()`

Examples

```
data(dfr) # Results for AutoML Predictions
lapply(dfr[c(1, 2)], head)

# Results for Binomial Model
conf_mat(dfr$class2$tag, dfr$class2$scores)
conf_mat(dfr$class2$tag, dfr$class2$scores, thresh = 0.3)
conf_mat(dfr$class2$tag, dfr$class2$scores, sense = "<=")

# Results for Multi-Categorical Model
conf_mat(dfr$class3$tag, dfr$class3$score)
```

---

corr *Correlation table*

Description

This function correlates a whole dataframe, running one hot smart encoding (`ohse`) to transform non-numerical features. Note that it will automatically suppress columns with less than 3 non missing values and warn the user.

Usage

```
corr(
  df,
  method = "pearson",
  use = "pairwise.complete.obs",
  pvalue = FALSE,
  padjust = NULL,
  half = FALSE,
  dec = 6,
  ignore = NULL,
  dummy = TRUE,
```

```
  redundant = NULL,
  logs = FALSE,
  limit = 10,
  top = NA,
  ...
)
```

Arguments

df	Dataframe. It doesn't matter if it's got non-numerical columns: they will be filtered.
method	Character. Any of: c("pearson", "kendall", "spearman").
use	Character. Method for computing covariances in the presence of missing values. Check `stats::cor` for options.
pvalue	Boolean. Returns a list, with correlations and statistical significance (p-value) for each value.
padjust	Character. NULL to skip or any of `p.adjust.methods` to calculate adjust p-values for multiple comparisons using `p.adjust()`.
half	Boolean. Return only half of the matrix? The redundant symmetrical correlations will be NA.
dec	Integer. Number of decimals to round correlations and p-values.
ignore	Vector or character. Which column should be ignored?
dummy	Boolean. Should One Hot (Smart) Encoding (`ohse()`) be applied to categorical columns?
redundant	Boolean. Should we keep redundant columns? i.e. If the column only has two different values, should we keep both new columns? Is set to NULL, only binary variables will dump redundant columns.
logs	Boolean. Calculate log(x)+1 for numerical columns?
limit	Integer. Limit one hot encoding to the n most frequent values of each column. Set to NA to ignore argument.
top	Integer. Select top N most relevant variables? Filtered and sorted by mean of each variable's correlations.
...	Additional parameters passed to `ohse`, `corr`, and/or `cor.test`.

Value

data.frame. Squared dimensions (N x N) to match every correlation between every `df` data.frame column/variable. Notice that when using `ohse()` you may get more dimensions.

See Also

Other Calculus: `dist2d()`, `model_metrics()`, `quants()`

Other Correlations: `corr_cross()`, `corr_var()`

Examples

```
data(dft) # Titanic dataset
df <- dft[, 2:5]

# Correlation matrix (without redundancy)
corr(df, half = TRUE)

# Ignore specific column
corr(df, ignore = "Pclass")

# Calculate p-values as well
corr(df, pvalue = TRUE, limit = 1)

# Test when no more than 2 non-missing values
df$trash <- c(1, rep(NA, nrow(df) - 1))
# and another method...
corr(df, method = "spearman")
```

`corr_cross` *Ranked cross-correlation across all variables*

Description

This function creates a correlation full study and returns a rank of the highest correlation variables obtained in a cross-table.

Usage

```
corr_cross(
  df,
  plot = TRUE,
  pvalue = TRUE,
  max_pvalue = 1,
  type = 1,
  max = 1,
  top = 25,
  local = 1,
  ignore = NULL,
  contains = NA,
  grid = TRUE,
  rm.na = FALSE,
  quiet = FALSE,
  ...
)
```

Arguments

df	Dataframe. It doesn't matter if it's got non-numerical columns: they will be filtered.
plot	Boolean. Show and return a plot?
pvalue	Boolean. Returns a list, with correlations and statistical significance (p-value) for each value.
max_pvalue	Numeric. Filter non-significant variables. Range (0, 1]
type	Integer. Plot type. 1 is for overall rank. 2 is for local rank.
max	Numeric. Maximum correlation permitted (from 0 to 1)
top	Integer. Return top n results only. Only valid when type = 1. Set value to NA to use all cross-correlations
local	Integer. Label top n local correlations. Only valid when type = 2
ignore	Vector or character. Which column should be ignored?
contains	Character vector. Filter cross-correlations with variables that contains certain strings (using any value if vector used).
grid	Boolean. Separate into grids?
rm.na	Boolean. Remove NAs?
quiet	Boolean. Keep quiet? If not, show messages
...	Additional parameters passed to corr

Details

DataScience+ Post: Find Insights with Ranked Cross-Correlations

Value

Depending on input plot, we get correlation and p-value results for every combination of features, arranged by descending absolute correlation value, with a data.frame plot = FALSE or plot plot = TRUE.

See Also

Other Correlations: corr_var(), corr()

Other Exploratory: corr_var(), crosstab(), df_str(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), plot_nums(), tree_var()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset

# Only data with no plot
corr_cross(dft, plot = FALSE, top = 10)
```

```
# Show only most relevant results filtered by pvalue
corr_cross(dft, rm.na = TRUE, max_pvalue = 0.05, top = 15)

# Cross-Correlation for certain variables
corr_cross(dft, contains = c("Survived", "Fare"))

# Cross-Correlation max values per category
corr_cross(dft, type = 2, top = NA)
```

`corr_var` *Correlation between variable and dataframe*

Description

This function correlates a whole dataframe with a single feature. It automatically runs `ohse` (one-hot-smart-encoding) so no need to input only numerical values.

Usage

```
corr_var(
  df,
  var,
  ignore = NULL,
  trim = 0,
  clean = FALSE,
  plot = TRUE,
  top = NA,
  ceiling = 1,
  max_pvalue = 1,
  limit = 10,
  ranks = FALSE,
  zeroes = FALSE,
  save = FALSE,
  quiet = FALSE,
  ...
)

## S3 method for class 'corr_var'
plot(x, var, max_pvalue = 1, top = NA, limit = NULL, ...)
```

Arguments

`df`	Dataframe. It doesn't matter if it's got non-numerical columns: they will be filtered.
`var`	Variable. Name of the variable to correlate. Note that if the variable `var` is not numerical, 1. you may define which category to select from using 'var_category'; 2. You may have to add `redundant = TRUE` to enable all categories (instead of `n-1`).

ignore	Character vector. Which columns do you wish to exclude?
trim	Integer. Trim words until the nth character for categorical values (applies for both, target and values)
clean	Boolean. Use lares::cleanText for categorical values (applies for both, target and values)
plot	Boolean. Do you wish to plot the result? If set to TRUE, the function will return only the plot and not the result's data
top	Integer. If you want to plot the top correlations, define how many
ceiling	Numeric. Remove all correlations above... Range: (0-1]
max_pvalue	Numeric. Filter non-significant variables. Range (0, 1]
limit	Integer. Limit one hot encoding to the n most frequent values of each column. Set to NA to ignore argument.
ranks	Boolean. Add ranking numbers?
zeroes	Do you wish to keep zeroes in correlations too?
save	Boolean. Save output plot into working directory
quiet	Boolean. Keep quiet? If not, show messages
...	Additional parameters passed to corr and cor.test
x	corr_var object

Value

data.frame. With variables, correlation and p-value results for each feature, arranged by descending absolute correlation value.

See Also

Other Exploratory: corr_cross(), crosstab(), df_str(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), plot_nums(), tree_var()

Other Correlations: corr_cross(), corr()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset

corr_var(dft, Survived, method = "spearman", plot = FALSE, top = 10)

# With plots, results are easier to compare:

# Correlate Survived with everything else and show only significant results
dft %>% corr_var(Survived_TRUE, max_pvalue = 0.05)

# Top 15 with less than 50% correlation and show ranks
dft %>% corr_var(Survived_TRUE, ceiling = .6, top = 15, ranks = TRUE)
```

cran_logs *Download and plot daily downloads of CRAN packages*

Description

Download daily downloads stats from CRAN for any package, and plot. It can also be used as an auxiliary function to plot (`cranlogs::cran_downloads`) results.

Usage

```
cran_logs(
  input = "lares",
  from = Sys.Date() - 31,
  to = Sys.Date() - 1,
  type = "daily",
  plot = TRUE
)
```

Arguments

`input`	Character vector with package names or data.frame product of `cranlogs::cran_downloads`.
`from, to`	Dates. Range of dates to fetch downloads metrics.
`type`	Character. Any of: "daily" or "total".
`plot`	Boolean. Create a plot?

Value

List with data.frame and plot if `plot=TRUE`.

Examples

```
cran_logs(c("lares", "dplyr"), from = "2021-05-31")
```

crosstab *Weighted Cross Tabulation*

Description

A cross-tabulation function with output similar to STATA, tidy friendly, with weighting possibility.

Usage

```
crosstab(
  df,
  ...,
  wt = NULL,
  prow = FALSE,
  pcol = FALSE,
  pall = FALSE,
  decimals = 2,
  rm.na = FALSE,
  total = TRUE,
  order = TRUE
)
```

Arguments

df	Data.frame.
...	Variables. Dependent and independent variables.
wt	Variable, numeric. Weights.
prow, pcol, pall	Boolean. Calculate percent values for rows, columns, or the whole table, respectively.
decimals	Integer. How many decimals should be returned?
rm.na	Boolean. Remove NA values?
total	Boolean. Return total values column?
order	Boolean. Sort columns and rows by frequencies? Else, will be sorted alphabetically

Value

data.frame. Result of crossing the variables provided in ... and counting how many observations (rows) fall into each criteria.

See Also

Other Exploratory: corr_cross(), corr_var(), df_str(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), plot_nums(), tree_var()

Examples

```
data(dft) # Titanic dataset
crosstab(dft, Survived, Pclass, total = FALSE)
# Show values in percentages
crosstab(dft, Pclass, Survived, prow = TRUE)
crosstab(dft, Pclass, Survived, pall = TRUE)
# Weighted by another variable
crosstab(dft, Survived, Pclass, wt = Fare, prow = TRUE)
```

daily_portfolio *Daily Portfolio Dataframe*

Description

This function creates a dataframe will all relevant metrics and values, for the overall portfolio, for every day since inception.

Usage

```
daily_portfolio(hist, trans, cash, cash_fix = 0, window = "MAX")
```

Arguments

hist	Dataframe. Result from stocks_hist()
trans	Dataframe. Result from stocks_file()$transactions
cash	Dataframe. Result from stocks_file()$cash
cash_fix	Numeric. If, for some reason, you need to fix your cash amount for all reports, set the amount here
window	Character. Choose any of: "1W", "1M", "6M", "1Y", "YTD", "5Y", "MAX"

Value

data.frame. Processed at date and portfolio level.

See Also

Other Investment: daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

daily_stocks *Daily Stocks Dataframe*

Description

This function creates a dataframe with all relevant metrics and values, for each ticker or symbol, for every day since inception.

Usage

```
daily_stocks(hist, trans, tickers = NA, window = "MAX")
```

Arguments

hist	Dataframe. Result from stocks_hist()
trans	Dataframe. Result from stocks_file()$transactions
tickers	Dataframe. Result from stocks_file()$portfolio
window	Character. Choose any of: "1W", "1M", "6M", "1Y", "YTD", "5Y", "MAX"

Value

data.frame. Processed at date and symbol level.

See Also

Other Investment: daily_portfolio(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

---

dalex_local *DALEX Local*

Description

DALEX function for local interpretations

Usage

```
dalex_local(explainer, observation = NA, row = 1, type = "break_down")
```

Arguments

explainer	Object. Result from h2o_explainer function
observation	Data.frame. If you want to use an observation that was not in the original explainer function, add here. Else, use row
row	Dataframe. Row number from the data.frame used in explainer.
type	Character. The type of variable attributions. Either shap, oscillations, break_down or break_down_interactions.

Value

List. Containing observation, breakdown results, and breakdown plot.

See Also

Other Interpretability: dalex_residuals(), dalex_variable(), h2o_explainer()

dalex_residuals *DALEX Residuals*

Description

DALEX function for residuals

Usage

```
dalex_residuals(explainer)
```

Arguments

explainer	Object. Result from h2o_explainer function

Value

Plot. Based of explainer residual results.

See Also

Other Interpretability: dalex_local(), dalex_variable(), h2o_explainer()

dalex_variable *DALEX Partial Dependency Plots (PDP)*

Description

DALEX auxiliary function for creating Partial Dependency Plots and study variable's responses vs independent vector.

Usage

```
dalex_variable(explainer, vars, force_class = NA, ...)
```

Arguments

explainer	Object. Result from h2o_explainer function.
vars	Character vector. Which features do you wish to study?
force_class	Character. If you wish to force a class on your vars, which one do you need?
...	Additional parameters passed to model_profile.

Value

List. Containing PDP results, plot and vars input.

See Also

Other Interpretability: dalex_local(), dalex_residuals(), h2o_explainer()

Examples

```
# You must have "DALEX" library to use this auxiliary function:
## Not run:
# Having an "explainer" object created with \code{h2o_explainer}:
# For numerical variables
dalex_variable(explainer, vars = c("Age", "Fare"))
# For categorical variables
dalex_variable(explainer, vars = c("Pclass", "Sex"))

## End(Not run)
```

---

date_cuts *Convert Date into Year + Cut*

Description

This function returns categorical values for any date(s) using year cuts such as bimonths, quarters, terms, and halves.

Usage

```
date_cuts(date = Sys.Date(), type = "Q")
```

Arguments

date	Date. Date we wish to transform
type	Character. Any of the following: B (2 months), Q (3 months), T (4 months), H (6 months)

Value

Vector with date cut for each date

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
date_cuts(Sys.Date(), type = "Q")
date_cuts(Sys.Date(), type = "H")
```

date_feats *One Hot Encoding for Date/Time Variables (Dummy Variables)*

Description

This function lets the user automatically create new columns out of a dataframe or vector with date/time variables.

Usage

```
date_feats(
  dates,
  drop = FALSE,
  only = NA,
  append = FALSE,
  holidays = FALSE,
  country = "Venezuela",
  currency_pair = NA,
  quiet = FALSE
)
```

Arguments

dates	Vector or dataframe. Non-date/time columns will be automatically ignored/extracted.
drop	Boolean. Should the original date/time columns be kept in the results? Only valid when input is a dataframe.
only	Character or vector. Which columns do you wish to process? If non are explicitly defined, all will be processed
append	Boolean. Append results to existing data.frame? If FALSE, only calculated values will be returned.
holidays	Boolean. Include holidays as new columns?
country	Character or vector. For which countries should the holidays be included?
currency_pair	Character. Which currency exchange do you wish to get the history from? i.e, USD/COP, EUR/USD...
quiet	Boolean. Quiet all messages?

Value

data.frame with additional features calculated out of time or date vectors.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(),

quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Feature Engineering: holidays(), ohse()

Other One Hot Encoding: holidays(), ohe_commas(), ohse()

Examples

```
df <- data.frame(
  dates = sample(seq(Sys.Date() - 365, Sys.Date(), by = 1), 50),
  times = sample(seq(Sys.time() - 1e7, Sys.time(), by = 1), 50)
)

# Input as a vector or dataframe
date_feats(df, drop = TRUE) %>% head(10)

# Holidays given a date range and country
## Not run:
hol <- date_feats(
  seq(Sys.Date() - 365, Sys.Date(), by = 1),
  holidays = TRUE,
  country = "Venezuela"
)
head(hol[!is.na(hol$holiday_name), ])

## End(Not run)
```

---

db_download *Download/Import Dropbox File by File's Name*

Description

This function lets the user download a file from Dropbox, specifying its name, using a previously created token or with interactive window.

Usage

```
db_download(
  query,
  local_path = NULL,
  xlsx = TRUE,
  token_dir = NA,
  token_name = "token_pers.rds",
  quiet = FALSE
)
```

Arguments

query	Search string. This string is split (on spaces) into individual words. Files will be used if they contain all words in the search string.
local_path	Character. Path to save file to. If NULL (the default), saves file to working directory with same name. If not, but a valid folder, file will be saved in this folder with same basename as path. If not NULL and not a folder, file will be saved to this path exactly.
xlsx	Boolean. Is it an Excel file? Can be returned as a list for each tab and not as a file if needed. Will delete downloaded file.
token_dir	Character. RDS with token local directory. You may set to NA if you already set your credentials (see get_creds())
token_name	Character. RDS file name with your token's data.
quiet	Boolean. Keep quiet? If not, show informative messages.

Value

If query returns a .xlsx file and xlsx=TRUE, will return a data.frame. Else, local_path string.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Other Credentials: db_upload(), get_credentials(), get_tweets(), mail_send(), queryDB(), queryGA(), slackSend(), stocks_file(), stocks_report()

Other Dropbox: db_upload()

Examples

```
## Not run:
# Download a specific file
db_download("stocksReport.Rmd", local_path = "~/Desktop/generic.Rmd")
# Import an Excel file from Dropbox into a data.frame
df <- db_download("Portfolio LC.xlsx", xlsx = FALSE)

## End(Not run)
```

db_upload *Upload Local Files to Dropbox*

Description

This function lets the user upload a local file to Dropbox, using a previously created token or with interactive window.

Usage

```
db_upload(
  filename,
  dir,
  delete_file = FALSE,
  token_dir = NA,
  token_name = "token_pers.rds"
)
```

Arguments

filename	String. Local file's name to upload.
dir	String. Directory you wish to upload the file to.
delete_file	Boolean. Delete local file after uploading?
token_dir	Character. RDS with token local directory. You may set to NA if you already set your credentials (see `get_creds()`)
token_name	Character. RDS file name with your token's data.

Value

`TRUE` when successfully uploads file.

See Also

Other Tools: `autoline()`, `bind_files()`, `bring_api()`, `db_download()`, `export_plot()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

Other Credentials: `db_download()`, `get_credentials()`, `get_tweets()`, `mail_send()`, `queryDB()`, `queryGA()`, `slackSend()`, `stocks_file()`, `stocks_report()`

Other Dropbox: `db_download()`

dfr *Results for AutoML Predictions*

Description

List with categorical (2 and 3 classes) and continuous predictions, generated with `h2o_automl()` and the `dft`. Note that the models per se won't work to predict.

Usage

```
data(dfr)
```

Format

An object of class `"list"` with 3 `"data.frame"`

class2 Predictions for a Binomial Classification Model

class3 Predictions for a Multi-Categorical Classification Model

regr Predictions for a Continuous Regression Model

Value

List

See Also

Other Dataset: `dft`

Examples

```
data(dfr)
lapply(dfr, head)
```

dft *Titanic Dataset*

Description

The sinking of the Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the widely considered "unsinkable" RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren't enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew. While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others. This dataset contains the details of a subset of the passengers on board (891 to be exact) taken from Kaggle's Titanic Train.csv.

Usage

```
data(dft)
```

Format

An object of class `"data.frame"`

PassengerId Unique ID for each passenger (1-891)

Survived Did the passenger survive? (TRUE, FALSE)

Pclass Ticket class, from first to third (1, 2, 3)

Sex Gender (female, male)

Age Age for each passenger in years (0.42-80)

SibSp Amount of siblings / spouses aboard the Titanic (0-8)

Parch Amount of parents / children aboard the Titanic (0-6)

Ticket Ticket IDs

Fare Amount paid for passenger's ticket (0-512.3292)

Cabin width of top of diamond relative to widest point (43-95)

Embarked Port of Embarkation (43-95)

Value

data.frame

See Also

Other Dataset: `dfr`

Examples

```
data(dft)
head(dft)
```

`df_str` *Dataset columns and rows structure*

Description

This function lets the user to check quickly the structure of a dataset (data.frame). It returns multiple counters for useful metrics, a plot, and a list of column names for each of the column metrics.

Usage

```
df_str(df, return = "plot", subtitle = NA, quiet = FALSE)
```

Arguments

df	Dataframe
return	Character. Return "skimr" for skim report, "numbers" for stats and numbers, "names" for a list with the column names of each of the class types, "plot" for a nice plot with "numbers" output, "distr" for an overall summary plot showing categorical, numeric, and missing values by using plot_df distributions
subtitle	Character. Add subtitle to plot
quiet	Boolean. Keep quiet or show other options available?

Value

Depending on return input and based on your df structure:

- list with the names of the columns classified by class
- data.frame with numbers: total values, row, columns, complete rows
- plot with visualizations

See Also

Other Exploratory: corr_cross(), corr_var(), crosstab(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), plot_nums(), tree_var()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset
df_str(dft, "names")
df_str(dft, "numbers", quiet = TRUE)
df_str(dft, "plot", quiet = TRUE)
```

---

dist2d *Distance from specific point to line*

Description

This function lets the user calculate the mathematical linear distance Between a specific point and a line (given geometrical 3 points)

Usage

```
dist2d(x, a = c(0, 0), b = c(1, 1))
```

Arguments

x	Vector. Coordinates of the point from which we want to measure the distance
a	Vector. Coordinates of 1st point over the line
b	Vector. Coordinates of 2st point over the line

Value

Numeric value result

See Also

Other Calculus: `corr()`, `model_metrics()`, `quants()`

Examples

```
dist2d(x = c(5, 2))
dist2d(x = c(5, 2), a = c(0, 0), b = c(0, 1))
dist2d(x = c(5, 2), a = c(0, 0), b = c(1, 0))
```

---

distr *Compare Variables with their Distributions*

Description

Compare the distribution of a target variable vs another variable. This function automatically splits into quantiles for numerical variables. Custom and tidyverse friendly.

Usage

```
distr(
  data,
  ...,
  type = 1,
  ref = TRUE,
  note = NA,
  top = 10,
  breaks = 10,
  na.rm = FALSE,
  force = "none",
  trim = 0,
  clean = FALSE,
  abc = FALSE,
  custom_colours = FALSE,
  plot = TRUE,
  chords = FALSE,
  save = FALSE,
  subdir = NA
)
```

Arguments

data	Dataframe
...	Variables. Main (target variable) and secondary (values variable) to group by (if needed).
type	Integer. 1 for both plots, 2 for counter plot only, 3 for percentages plot only.
ref	Boolean. Show a reference line if levels = 2? Quite useful when data is unbalanced (not 50/50) because a reference line is drawn.
note	Character. Caption for the plot.
top	Integer. Filter and plot the most n frequent for categorical values.
breaks	Integer. Number of splits for numerical values.
na.rm	Boolean. Ignore NAs if needed.
force	Character. Force class on the values data. Choose between 'none', 'character', 'numeric', 'date'
trim	Integer. Trim labels until the nth character for categorical values (applies for both, target and values)
clean	Boolean. Use cleanText() for categorical values (applies for both, target and values)
abc	Boolean. Do you wish to sort by alphabetical order?
custom_colours	Boolean. Use custom colours function?
plot	Boolean. Return a plot? Otherwise, a table with results
chords	Boolean. Use a chords plot?
save	Boolean. Save the output plot in our working directory
subdir	Character. Into which subdirectory do you wish to save the plot to?

Value

Plot when plot=TRUE with two plots in one: counter distribution grouped by cuts, and proportions distribution grouped by same cuts. data.frame when plot=FALSE with counting, percentages, and cumulative percentages results. When type argument is used, single plots will be returned.

See Also

Other Exploratory: corr_cross(), corr_var(), crosstab(), df_str(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), plot_nums(), tree_var()

Other Visualization: freqs_df(), freqs_list(), freqs_plot(), freqs(), noPlot(), plot_chord(), plot_survey(), plot_timeline(), tree_var()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset

# Relation for categorical/categorical values
distr(dft, Survived, Sex)

# Relation for categorical/numeric values
dft %>%
  distr(Survived, Fare, plot = FALSE) %>%
  head(10)
# Sort values
dft %>% distr(Survived, Fare, abc = TRUE)
# Less splits/breaks
dft %>% distr(Survived, Fare, abc = TRUE, breaks = 5)

# Distribution of numerical only
dft[dft$Fare < 20, ] %>% distr(Fare)

# Distribution of numerical/numerical
dft %>% distr(Fare, Age)

# Select only one of the two default plots of distr()
dft %>% distr(Survived, Age, type = 2)
dft %>% distr(Survived, Age, type = 3)
```

---

errors *Calculate Continuous Values Errors*

Description

This function lets the user calculate all errors and R squared simultaneously.

This function lets the user calculate Root Mean Squared Error

This function lets the user calculate Mean Absolute Error

This function lets the user calculate Mean Squared Error

This function lets the user calculate Mean Squared Error

This function lets the user calculate R Squared

This function lets the user calculate Adjusted R Squared

Usage

```
errors(tag, score)

rmse(tag, score)

mae(tag, score)
```

```
mse(tag, score)

mape(tag, score)

rsq(tag, score)

rsqa(tag, score)
```

Arguments

tag	Vector. Real known label
score	Vector. Predicted value or model's result

Value

data.frame or numeric values results for multiple error metrics on continuous numerical vectors inputs.

See Also

Other Model metrics: ROC(), conf_mat(), gain_lift(), loglossBinary(), model_metrics()

Examples

```
data(dfr) # Results for AutoML Predictions
head(dfr$regr)
df <- errors(dfr$regr$tag, dfr$regr$score)
head(df)
```

---

etf_sector *ETF's Sectors Breakdown*

Description

This function scraps etf.com data for sector breakdown on ETFs. Use splot_etf() for visualization.

Usage

```
etf_sector(etf = "VTI", quiet = FALSE, cache = TRUE)
```

Arguments

etf	Character Vector. Which ETFs you wish to scrap?
quiet	Boolean. Keep quiet? If not, informative messages will be printed.
cache	Boolean. Use daily cache if available?

Value

data.frame with ETF break.down data by sector

See Also

Other Investment: daily_portfolio(), daily_stocks(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

Examples

```
etf_sector(etf = "VTI")
```

---

export_plot *Export ggplot2, gridExtra, or any plot object into rendered file*

Description

Export any ggplot2, gridExtra, or any plot object created with R into rendered png or jpg file.

Usage

```
export_plot(
  p,
  name = "plot",
  vars = NA,
  sep = ".vs.",
  width = 8,
  height = 6,
  format = "png",
  res = 300,
  dir = getwd(),
  subdir = NA,
  quiet = FALSE
)
```

Arguments

p	Plot object. Plot to render and export.
name	Character. File's name or suffix if vars is not NA. No need to include file format on file name.
vars	Vector. Variable names to identify by filename.
sep	Character. Separator for vars.

`width, height, res`
: Numeric. Plot's width, height, and res (for grids).

`format`
: Character. One of: `png` or `jpeg`.

`dir, subdir`
: Character. In which directory/subdirectory do you wish to save the plot? Working directory as default `dir`.

`quiet`
: Boolean. Display successful message with filename when saved?

Value

No return value, called for side effects.

See Also

Other Tools: `autoline()`, `bind_files()`, `bring_api()`, `db_download()`, `db_upload()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

Examples

```
p <- noPlot()
export_plot(p, name = "noplot", width = 10, height = 8, res = 300, dir = tempdir())
export_plot(p, name = "noplot2", subdir = "newplots", dir = tempdir())
```

---

export_results *Export h2o_automl's Results*

Description

Export RDS, TXT, POJO, MOJO and all results from `h2o_automl()`.

Usage

```
export_results(
  results,
  thresh = 10,
  which = c("txt", "csv", "rds", "binary", "mojo", "plots", "dev", "production"),
  note = NA,
  subdir = NA,
  save = TRUE,
  seed = 0
)
```

Arguments

results	h2o_automl or h2o model
thresh	Integer. Threshold for selecting binary or regression models: this number is the threshold of unique values we should have in 'tag' (more than: regression; less than: classification)
which	Character vector. Select which file format to export: Possible values: txt, csv, rds, binary, mojo, plots. You might also use dev (txt, csv, rds) or production (binary, mojo) or simply don't use parameter to export everything
note	Character. Add a note to the txt file. Useful when lots of models are trained and saved to remember which one is which one
subdir	Character. In which directory do you wish to save the results?
save	Boolean. Do you wish to save/export results?
seed	Numeric. For reproducible results and random splits.

Value

No return value, called for side effects.

See Also

Other Machine Learning: ROC(), conf_mat(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_binary(), h2o_predict_model(), h2o_selectmodel(), impute(), iter_seeds(), lasso_vars(), model_metrics(), model_preprocess(), msplit()

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

---

fb_accounts *Facebook Ad Accounts*

Description

This returns all ad accounts for a FB Business Account FB. For more information on Ad Insights' API, go to the original documentaion

Usage

```
fb_accounts(
  token,
  business_id = "904189322962915",
  type = c("owned", "client"),
  limit = 100,
```

```
  api_version = NULL,
  ...
)
```

Arguments

token	Character. Valid access token with sufficient privileges. Visit the Facebook API Graph Explorer to acquire one.
business_id	Character. Business ID.
type	Character vector. Values: owned, client.
limit	Integer. Query limit by pagination.
api_version	Character. Facebook API version.
...	Additional parameters

Value

data.frame with un-nested processed results fetched with API.

See Also

Other API: bring_api(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token(), gpt_ask(), li_auth(), li_profile(), queryGA(), slackSend()

Other Meta: fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token()

Examples

```
## Not run:
# Query all accounts (owned and with permissions) of a Business ID
accounts <- fb_accounts(YOURTOKEN, YOURBUSINESS)

## End(Not run)
```

---

fb_ads *Facebook Ads API*

Description

This returns all available FB ads for any account, campaign, or ad set id. For more information on Ad' API, go to the original documentaion

Usage

```
fb_ads(
  token,
  which,
  start_date = Sys.Date() - 31,
  end_date = Sys.Date(),
  fields = NA,
  api_version = NULL,
  process = TRUE,
  ...
)
```

Arguments

token	Character. Valid access token with sufficient privileges. Visit the Facebook API Graph Explorer to acquire one.
which	Character vector. This is the accounts, campaigns, adsets, or ads IDs to be queried. Remember: if report_level = "account", you must start the ID with act_.
start_date, end_date	Character. The first and last full day to report, in the format "YYYY-MM-DD".
fields	Character, json format. Leave NA for default fields OR NULL to ignore.
api_version	Character. Facebook API version.
process	Boolean. Process GET results to a more friendly format?
...	Additional parameters

Details

This function was based on FBinsightsR.

Value

data.frame with un-nested processed results if process=TRUE or raw API results as list when process=FALSE.

See Also

Other API: bring_api(), fb_accounts(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token(), gpt_ask(), li_auth(), li_profile(), queryGA(), slackSend()

Other Meta: fb_accounts(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token()

Examples

```
## Not run:
token <- YOURTOKEN
account <- act_ADACCOUNT

# Query all ads for "which" (account) with results in the last 10 days
ads <- fb_ads(token, account, start_date = Sys.Date() - 10)

## End(Not run)
```

`fb_creatives` *Facebook Creatives API*

Description

For more information: Marketing API

Usage

```
fb_creatives(token, which, api_version = NULL, process = TRUE, ...)
```

Arguments

`token`	Character. Valid access token with sufficient privileges. Visit the Facebook API Graph Explorer to acquire one.
`which`	Character vector. This is the accounts, campaigns, adsets, or ads IDs to be queried. Remember: if `report_level = "account"`, you must start the ID with `act_`.
`api_version`	Character. Facebook API version.
`process`	Boolean. Process GET results to a more friendly format?
`...`	Additional parameters.

Value

data.frame with un-nested processed results if `process=TRUE` or raw API results as list when `process=FALSE`.

See Also

Other API: `bring_api()`, `fb_accounts()`, `fb_ads()`, `fb_insights()`, `fb_process()`, `fb_report_check()`, `fb_rf()`, `fb_token()`, `gpt_ask()`, `li_auth()`, `li_profile()`, `queryGA()`, `slackSend()`

Other Meta: `fb_accounts()`, `fb_ads()`, `fb_insights()`, `fb_process()`, `fb_report_check()`, `fb_rf()`, `fb_token()`

Examples

```
## Not run:
token <- YOURTOKEN
account <- act_ADACCOUNT

# Query all creatives for "which" (account in this case)
creatives <- fb_creatives(token, account)

## End(Not run)
```

`fb_insights` *Facebook Insights API*

Description

This returns all available FB insights per day including any given breakdown to the specified report level, and place into a data frame. For more information on Ad Insights' API, go to the original documentaion.

Usage

```
fb_insights(
  token,
  which,
  start_date = Sys.Date() - 7,
  end_date = Sys.Date(),
  time_increment = "1",
  report_level = "campaign",
  ad_object = "insights",
  breakdowns = NA,
  fields = NA,
  filtering = NULL,
  limit = 100,
  api_version = NULL,
  process = TRUE,
  async = FALSE,
  ...
)
```

Arguments

`token`	Character. Valid access token with sufficient privileges. Visit the Facebook API Graph Explorer to acquire one.
`which`	Character vector. This is the accounts, campaigns, adsets, or ads IDs to be queried. Remember: if `report_level = "account"`, you must start the ID with `act_`.

start_date, end_date
: Character. The first and last full day to report, in the format "YYYY-MM-DD".

time_increment
: Character. Group by months ("monthly"), everything together ("all_days") or an integer per days [1-90]. Default: each day separately (i.e. "1").

report_level
: Character. One of "ad", "adset", "campaign", or "account"

ad_object
: Character. One of: "insights" (default), "adsets", ...

breakdowns
: Character Vector. One or more of breakdowns for segmentation results. Set to NA for no breakdowns

fields
: Character, json format. Leave NA for default fields OR NULL to ignore.

filtering
: List. Each filter will be a list containing "field", "operator", and "value". Read more about the operators in the official docs. Example: dplyr::tibble(field = "country", operator = "IN", value = list("PE"))).

limit
: Integer. Query limit by pagination.

api_version
: Character. Facebook API version.

process
: Boolean. Process GET results to a more friendly format?

async
: Boolean. Run an async query. When set to TRUE, instead of making a GET query, it'll run a POST query and will return a report run ID.

...
: Additional parameters

Value

data.frame with un-nested processed results if process=TRUE or raw API results as list when process=FALSE.

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_process(), fb_report_check(), fb_rf(), fb_token(), gpt_ask(), li_auth(), li_profile(), queryGA(), slackSend()

Other Meta: fb_accounts(), fb_ads(), fb_creatives(), fb_process(), fb_report_check(), fb_rf(), fb_token()

Examples

```
## Not run:
token <- "YOURTOKEN"
which <- "act_20846447"

# Platforms' Insights: all ad-sets platforms of "which" account,
# aggregated, for the last 30 days
platforms <- fb_insights(
  token, which,
  start_date = Sys.Date() - 30,
  time_increment = "all_days",
  report_level = "adset",
  fields = c(
    "account_name",
    "adset_id",
```

```
    "adset_start",
    "adset_end"
  ),
  breakdowns = c(
    "publisher_platform",
    "platform_position",
    "device_platform"
  )
)

# Daily results for all campaigns of "which" account,
# with custom performance fields with no breakdowns.
insights_adset <- fb_insights(
  token, which,
  time_increment = "1",
  report_level = "campaign",
  fields = c(
    "adset_id",
    "reach",
    "frequency",
    "spend",
    "cpm",
    "objective",
    "optimization_goal"
  )
)

## End(Not run)
```

`fb_process` *Paginate and Process Facebook's API Results*

Description

Process and paginate raw results from Facebook's API, result of querying the API with `httr::GET` or by passing an API link.

Usage

```
fb_process(input, paginate = TRUE, sleep = 0, quiet = FALSE, ...)
```

Arguments

`input`	GET's output object (response) or link (character).
`paginate`	Boolean or integer. Run through all paginations? If set to `FALSE`, only the first one will be processed. If set to any other integer value, will process the first N paginations.
`sleep`	Numeric value. How much should each loop wait until until running the next pagination query?

quiet	Boolean. Quiet messages?
...	Additional parameters.

Value

data.frame with un-nested processed results or NULL if no results found.

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_report_check(), fb_rf(), fb_token(), gpt_ask(), li_auth(), li_profile(), queryGA(), slackSend()

Other Meta: fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_report_check(), fb_rf(), fb_token()

---

fb_report_check *Facebook API Report Status Check*

Description

This returns all available FB insights per day including any given breakdown to the specified report level, and place into a data frame. For more information on Ad Insights' API, go to the original documentaion.

Usage

```
fb_report_check(
  token,
  report_run_id,
  api_version = NULL,
  live = FALSE,
  sleep = 10,
  quiet = FALSE
)
```

Arguments

token	Character. Valid access token with sufficient privileges. Visit the Facebook API Graph Explorer to acquire one.
report_run_id	Integer. Report ID to check status.
api_version	Character. Facebook API version.
live	Boolean. Run until status report is finished?
sleep	Boolean. If live=TRUE, then how many seconds should we wait until next check?
quiet	Boolean. Quiet messages?

Value

List with API status results.

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_rf(), fb_token(), gpt_ask(), li_auth(), li_profile(), queryGA(), slackSend()

Other Meta: fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_rf(), fb_token()

Examples

```
## Not run:
token <- "YOURTOKEN"
report_run_id <- "123456789"
fb_report_check(token, report_run_id, live = TRUE, quiet = FALSE)

## End(Not run)
```

---

fb_rf *Facebook Reach and Frequency API*

Description

Create or query reach and frequency predictions using Facebook's Reach and Frequency API. For more information on the API and its parameters, go to the original documentaion.

Usage

```
fb_rf(
  token,
  ad_account = NA,
  prediction = NA,
  objective = "REACH",
  days = 28,
  budget = 2e+06,
  destination_ids = NA,
  countries = "MX",
  frequency_cap = 8,
  prediction_mode = 1,
  curve = TRUE,
  api_version = NULL,
  process = TRUE,
  ...
)
```

Arguments

token	Character. Valid access token with sufficient privileges. Visit the Facebook API Graph Explorer to acquire one.
ad_account	Character. Ad Account. Remember to start with act_. If you use the prediction argument, no need to provide this parameter.
prediction	Integer. Prediction ID if you already created the prediction and wish to query the curve's data. As this prediction already exists, the rest of arguments of this function will be ignored.
objective	Character. Any of: "BRAND_AWARENESS", "LINK_CLICKS", "POST_ENGAGEMENT", "MOBILE_APP_INSTALLS", "CONVERSIONS", "REACH", or "VIDEO_VIEWS".
days	Integer. Amount of days for your campaign's predictions.
budget	Integer. The budget in the Ad Account currency in cents.
destination_ids	Integer vector. Page ID and/or Instagram Account ID.
countries	Character vector. Country's acronyms.
frequency_cap	Integer. Frequency cap over all the campaign duration.
prediction_mode	Integer. "1" for predicting Reach by providing budget, "2" is for predicting Budget given a specific Reach.
curve	Boolean. Return curve data? If not, only prediction will be created.
api_version	Character. Facebook API version.
process	Boolean. Process GET results to a more friendly format?
...	Additional parameters passed to target specs.

Value

data.frame with un-nested processed results if process=TRUE or raw API results as list when process=FALSE.

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_token(), gpt_ask(), li_auth(), li_profile(), queryGA(), slackSend()

Other Meta: fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_token()

Examples

```
## Not run:
token <- "YOURTOKEN"
account_id <- "act_20846447"

# BASIC 1: Create and return data for a new prediction
basic1 <- fb_rf(token, account_id, destination_ids = 187071108930, countries = "AR")
```

```
# BASIC 2: Fetch data for an existing prediction ID
basic2 <- fb_rf(token, account_id, prediction = 6317720998974)

# ADVANCED (Fully custom prediction)
advanced <- fb_rf(token, account_id,
  objective = "REACH",
  days = 28,
  budget = 2000000,
  destination_ids = c(187071108930, 1142958119078556),
  age_min = 15,
  age_max = 65,
  genders = 2,
  countries = "MX",
  publisher_platforms = c(
    "facebook",
    "instagram",
    #' audience_network',
    "messenger"
  ),
  # interests_ids = NA,
  facebook_positions = c(
    "feed",
    #' instant_article',
    "marketplace",
    "video_feeds",
    "story",
    "search",
    "instream_video"
  ),
  instagram_positions = c(
    "stream",
    "story",
    "explore"
  ),
  # audience_network_positions = c(
  #  'classic',
  #  'instream_video')
  messenger_positions = c(
    "messenger_home",
    "sponsored_messages",
    "story"
  ),
  device_platforms = c(
    "mobile",
    "desktop"
  )
)

## End(Not run)
```

fb_token *Facebook's Long-Life User API Token*

Description

Using a 1-hour generic user token you can generate a 60 day token. You will need to have an App ID and App secret, and a valid token. Generate a new valid User Token with the API Graph.

Usage

```
fb_token(app_id, app_secret, token, api_version = NULL)
```

Arguments

app_id, app_secret	Character. Application ID and Secret.
token	Character. User token, created with API Graph or with this same fb_token()'s token.
api_version	Character. Facebook API version.

Details

More info: Long-Lived Access Tokens

Value

Character. String with token requested. If successful, it'll contain an attribute called "expiration" with date and time of expiration.

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), gpt_ask(), li_auth(), li_profile(), queryGA(), slackSend()

Other Meta: fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf()

---

filesGD *Google Drive Files (API v4)*

Description

Authenticate and find Google Drive files and IDs by name.

Usage

```
filesGD(title, server = FALSE, json = NULL, api_key = NULL, email = NULL)
```

Arguments

title	Character. Title of Google Drive file. Uses regular expressions so you may fetch with patterns instead of names.
server	Boolean. Force interacting auth process?
json	Character. JSON filename with service auth
email, api_key	Character. If you have multiple pre-authorized accounts in your machine, you may non-interactively select which one you wish to use by email and/or api_key.

Value

Vector with found file names based on title on Google Drive.

See Also

Other Scrapper: get_mp3(), gtrends_related(), holidays(), ip_data(), readGS(), splot_etf(), stocks_quote()

Other Google: gtrends_related(), queryGA(), readGS()

---

files_functions *List all functions used in R script files by package*

Description

Parses all functions called by an R script and then lists them by package. Wrapper for 'getParseData'. May be of great use for those developing a package to help see what namespace 'importsFrom' calls will be required.

Usage

```
files_functions(filename, abc = TRUE, quiet = FALSE)
```

Arguments

filename	Character. Path to an R file (or directory) containing R code files.
abc	Boolean. List functions alphabetically. If FALSE, will list in order of frequency.
quiet	Boolean. Keep quiet? If not, print messages and statusbar.

Value

data.frame. Each row is a function and columns stating number of appearances, percentage, packages, and files searched.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
## Not run:
# Choose an R script file with functions
rfile <- file.choose()
files_functions(rfile)

## End(Not run)
```

---

file_name *Extract file raw name and type from file names*

Description

Extract file raw name and type from file names

Get file extensions without file names

Usage

```
file_name(filepath)

file_type(filepath)
```

Arguments

filepath	Character vector. File path(s) to get file raw names without extension nor path OR extension without path nor raw name.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
file_name("file.aux")
file_name("temp/file.R")
file_name("/temp/temp3/music.mp3")
file_type("file.aux")
file_type("temp/file.R")
file_type("/temp/temp3/music.mp3")
```

`font_exists` *Check if Font is Installed*

Description

This function checks if a font is installed in your machine.

Usage

```
font_exists(font = "Arial Narrow", quiet = FALSE)
```

Arguments

`font`	Character. Which font to check
`quiet`	Boolean. Keep quiet? If not, show message

Value

Boolean result of the existing fonts check.

See Also

Other Tools: `autoline()`, `bind_files()`, `bring_api()`, `db_download()`, `db_upload()`, `export_plot()`, `export_results()`, `files_functions()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

Examples

```
font_exists(font = "Arial")
font_exists(font = "XOXO")
font_exists(font = "")
```

`forecast_arima` *ARIMA Forecast*

Description

This function automates the ARIMA iterations and modeling for time forecasting. For the moment, units can only be days.

Usage

```
forecast_arima(
  time,
  values,
  n_future = 30,
  ARMA = 8,
  ARMA_min = 5,
  AR = NA,
  MA = NA,
  wd_excluded = NA,
  plot = TRUE,
  plot_days = 90,
  project = NA
)
```

Arguments

`time`	POSIX. Vector with date values
`values`	Numeric. Vector with numerical values
`n_future`	Integer. How many steps do you wish to forecast?
`ARMA`	Integer. How many days should the model look back for ARMA? Between 5 and 10 days recommmended. If set to 0 then it will forecast until the end of max date's month; if set to -1, until the end of max date's following month
`ARMA_min`	Integer. How many days should the model look back for ARMA? Between 5 and 10 days recommmended. If set to 0 then it will forecast until the end of max date's month; if set to -1, until the end of max date's following month
`AR`	Integer. Force AR value if known
`MA`	Integer. Force MA value if known
`wd_excluded`	Character vector. Which weekdays are excluded in your training set. If there are, please define know which ones. Example: c('Sunday','Thursday'). If set to 'auto' then it will detect automatically which weekdays have no data and forcast without these days.
`plot`	Boolean. If you wish to plot your results
`plot_days`	Integer. How many days back you wish to plot?
`project`	Character. Name of your forecast project

Details

The ARIMA method is appropriate only for a time series that is stationary (i.e., its mean, variance, and autocorrelation should be approximately constant through time) and it is recommended that there are at least 50 observations in the input data.

The model consists of two parts, an autoregressive (AR) part and a moving average (MA) part. The AR part involves regressing the variable on its own lagged (i.e., past) values. The MA part involves modeling the error term as a linear combination of error terms occurring contemporaneously and at various times in the past.

One thing to keep in mind when we think about ARIMA models is given by the great power to capture very complex patters of temporal correlation (Cochrane, 1997: 25)

Value

List. Containing the trained model, forecast accuracy results, data.frame for forecast (test) and train, and if `plot=TRUE`, a plot.

See Also

Other Forecast: `prophesize()`

---

formatColoured *Print Coloured Messages*

Description

Print Coloured Messages

Usage

```
formatColoured(
  txt,
  colour = c("yellow", "blue", "grey"),
  bold = FALSE,
  cat = TRUE
)
```

Arguments

`txt`	Character. Text to print or transform.
`colour`	Character. Any of: grey, red, green, yellow, blue, or purple.
`bold`	Boolean. Set bold text?
`cat`	Boolean. Print with cat? If not, raw string

Value

Depends on `cat`: NULL if TRUE or character string if FALSE.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
opts <- c("GREY", "RED", "GREEN", "YELLOW", "BLUE", "PURPLE")
for (colour in opts) formatColoured(paste("Colour:", colour, "\n"), colour)
formatColoured("my bold coloured text", bold = TRUE, cat = TRUE)
```

---

formatHTML *Format a string text as markdown/HTML*

Description

Format any character string to HTML or markdown format. We recommend using this format with the `ggtext::geom_richtext` function to format text in `ggplot2` objects.

This function lets the user format numerical values nicely

Usage

```
formatHTML(text, color = "black", size = 20, bold = FALSE)

formatNum(
  x,
  decimals = 2,
  signif = NULL,
  type = Sys.getenv("LARES_NUMFORMAT"),
  pre = "",
  pos = "",
  sign = FALSE,
  abbr = FALSE,
  ...
)
```

Arguments

text	Character. Strings to format.
color	Character. Hex colour code.
size	Numeric. Text size.
bold	Boolean. Should the text be bold?
x	Numerical Vector

decimals	Integer. Amount of decimals to display. If set to NULL, then getOption("digits") will be used.
signif	Integer. Rounds the values in its first argument to the specified number of significant digits.
type	Integer. 1 for International standards. 2 for American Standards. Use Sys.setenv("LARES_NUMFORMAT" = 2) to set this parameter globally.
pre, pos	Character. Add string before or after number.
sign	Boolean. Add + sign to positive values.
abbr	Boolean. Abbreviate using num_abbr()? You can use the 'decimals' parameter to set abbr's n(-1) parameter.
...	Additional lazy eval parameters.

Value

String with format characters included.

Character. String vector with reformatted continuous numbers

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
formatHTML("Text test", color = "#000000")
formatHTML(c(123, 456), color = "orange", size = 120, bold = TRUE)

# If you want to use it with \code{ggtext}:
## Not run:
col1 <- "grey"
col2 <- "orange"
pt <- data.frame(
  label = paste0(
    formatHTML(123, color = col2, size = 120, bold = TRUE), "<br/>",
    formatHTML("of children had a", col1), "<br/>",
    formatHTML("traditional stay-at-home mom", color = col2, bold = TRUE), "<br/>",
    formatHTML(paste0("in 2012, compared to ", 321, " in 1970"), color = col1)
  )
)
ggplot(pt, aes(x = 0, y = 0)) +
```

```
  ggtext::geom_richtext(
    aes(label = label),
    hjust = 0,
    label.color = NA,
    lineheight = 1.5
  ) +
  xlim(0, 0.01) +
  theme_void()

## End(Not run)
formatNum(1.23456, decimals = 3)
formatNum(1.23456, type = 1)
formatNum(1.23456, pre = "$", pos = "/person")
formatNum(123456, abbr = TRUE)
formatNum(1234567890, abbr = TRUE, signif = 2)
formatNum(1234567890, decimals = 0, abbr = TRUE)
formatNum(c(-3:3), sign = TRUE)
```

`freqs` *Frequencies Calculations and Plot*

Description

This function lets the user group, count, calculate percentages and cumulatives. It also plots results if needed. Tidyverse friendly.

Usage

```
freqs(
  df,
  ...,
  wt = NULL,
  rel = FALSE,
  results = TRUE,
  variable_name = NA,
  plot = FALSE,
  rm.na = FALSE,
  title = NA,
  subtitle = NA,
  top = 20,
  abc = FALSE,
  save = FALSE,
  subdir = NA
)
```

Arguments

`df`	Data.frame

...	Variables. Variables you wish to process. Order matters. If no variables are passed, the whole data.frame will be considered
wt	Variable, numeric. Weights.
rel	Boolean. Relative percentages (or absolute)?
results	Boolean. Return results in a dataframe?
variable_name	Character. Overwrite the main variable's name
plot	Boolean. Do you want to see a plot? Three variables tops.
rm.na	Boolean. Remove NA values in the plot? (not filtered for numerical output; use na.omit() or filter() if needed)
title	Character. Overwrite plot's title with.
subtitle	Character. Overwrite plot's subtitle with.
top	Integer. Filter and plot the most n frequent for categorical values. Set to NA to return all values
abc	Boolean. Do you wish to sort by alphabetical order?
save	Boolean. Save the output plot in our working directory
subdir	Character. Into which subdirectory do you wish to save the plot to?

Value

Plot when `plot=TRUE` and data.frame with grouped frequency results when `plot=FALSE`.

See Also

Other Frequency: `freqs_df()`, `freqs_list()`, `freqs_plot()`

Other Exploratory: `corr_cross()`, `corr_var()`, `crosstab()`, `df_str()`, `distr()`, `freqs_df()`, `freqs_list()`, `freqs_plot()`, `lasso_vars()`, `missingness()`, `plot_cats()`, `plot_df()`, `plot_nums()`, `tree_var()`

Other Visualization: `distr()`, `freqs_df()`, `freqs_list()`, `freqs_plot()`, `noPlot()`, `plot_chord()`, `plot_survey()`, `plot_timeline()`, `tree_var()`

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset

# How many survived?
dft %>% freqs(Survived)

# How many survived per Class?
dft %>% freqs(Pclass, Survived, abc = TRUE)

# How many survived per Class with relative percentages?
dft %>% freqs(Pclass, Survived, abc = TRUE, rel = TRUE)

# Using a weighted feature
dft %>% freqs(Pclass, Survived, wt = Fare / 100)
```

```
# Let's check the results with plots:

#' # How many survived and see plot?
dft %>% freqs(Survived, plot = TRUE)

# How many survived per class?
dft %>% freqs(Survived, Pclass, plot = TRUE)

# Per class, how many survived?
dft %>% freqs(Pclass, Survived, plot = TRUE)

# Per sex and class, how many survived?
dft %>% freqs(Sex, Pclass, Survived, plot = TRUE)

# Frequency of tickets + Survived
dft %>% freqs(Survived, Ticket, plot = TRUE)

# Frequency of tickets: top 10 only and order them alphabetically
dft %>% freqs(Ticket, plot = TRUE, top = 10, abc = TRUE)
```

`freqs_df` *Plot for All Frequencies on Dataframe*

Description

This function lets the user analize data by visualizing the frequency of each value of each column from a whole data frame.

Usage

```
freqs_df(
  df,
  max = 0.9,
  min = 0,
  novar = TRUE,
  plot = FALSE,
  top = 30,
  quiet = FALSE,
  save = FALSE,
  subdir = NA
)
```

Arguments

`df`	Data.frame
`max`	Numeric. Top variance threshold. Range: (0-1]. These variables will be excluded

min	Numeric. Minimum variance threshold. Range: [0-1). These values will be grouped into a high frequency (HF) value
novar	Boolean. Remove no variance columns?
plot	Boolean. Do you want to see a plot? Three variables tops
top	Integer. Plot most relevant (less categories) variables
quiet	Boolean. Keep quiet? (or show variables exclusions)
save	Boolean. Save the output plot in our working directory
subdir	Character. Into which subdirectory do you wish to save the plot to?

Value

Plot when `plot=TRUE` and data.frame with grouped frequency results when `plot=FALSE`.

See Also

Other Frequency: `freqs_list()`, `freqs_plot()`, `freqs()`

Other Exploratory: `corr_cross()`, `corr_var()`, `crosstab()`, `df_str()`, `distr()`, `freqs_list()`, `freqs_plot()`, `freqs()`, `lasso_vars()`, `missingness()`, `plot_cats()`, `plot_df()`, `plot_nums()`, `tree_var()`

Other Visualization: `distr()`, `freqs_list()`, `freqs_plot()`, `freqs()`, `noPlot()`, `plot_chord()`, `plot_survey()`, `plot_timeline()`, `tree_var()`

Examples

```
data(dft) # Titanic dataset
freqs_df(dft)
freqs_df(dft, plot = TRUE)
```

---

freqs_list *Frequencies on Lists and UpSet Plot*

Description

Visualize frequency of elements on a list, list vector, or vector with comma separated values. Detect which combinations and elements are the most frequent and how much they represent of your total observations. This is similar to the UpSet Plots which may be used as an alternative to Venn diagrams.

Usage

```
freqs_list(
  df,
  var = NULL,
  wt = NULL,
  fx = "mean",
```

```
  rm.na = FALSE,
  min_elements = 1,
  limit = 10,
  limit_x = NA,
  limit_y = NA,
  tail = TRUE,
  size = 10,
  unique = TRUE,
  abc = FALSE,
  title = "",
  plot = TRUE
)
```

Arguments

df	Data.frame
var	Variable. Variables you wish to process.
wt	Variable, numeric. Select a numeric column to use in the colour scale, used as sum, mean... of those values for each of the combinations.
fx	Character. Set operation: mean, sum
rm.na	Boolean. Remove NA value from wt?
min_elements	Integer. Exclude combinations with less than n elements
limit, limit_x, limit_y	Integer. Show top n combinations (x) and/or elements (y). The rest will be grouped into a single element. Set argument to 0 to ignore. limit_x/limit_y answer to limit's argument.
tail	Boolean. Show tail grouped into "..." on the plots?
size	Numeric. Text base size
unique	Boolean. a,b = b,a?
abc	Boolean. Do you wish to sort by alphabetical order?
title	Character. Overwrite plot's title with.
plot	Boolean. Plot viz? Will be generated anyways in the output object

Value

List. data.frame with the data results, elements and combinations.

See Also

Other Frequency: freqs_df(), freqs_plot(), freqs()

Other Exploratory: corr_cross(), corr_var(), crosstab(), df_str(), distr(), freqs_df(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), plot_nums(), tree_var()

Other Visualization: distr(), freqs_df(), freqs_plot(), freqs(), noPlot(), plot_chord(), plot_survey(), plot_timeline(), tree_var()

Examples

```
## Not run:
df <- dplyr::starwars
head(df[, c(1, 4, 5, 12)], 10)

# Characters per movies combinations in a list column
head(df$films, 2)
freqs_list(df, films)

# Skin colours in a comma-separated column
head(df$skin_color)
x <- freqs_list(df, skin_color, min_elements = 2, limit = 5, plot = FALSE)
# Inside "x" we'll have:
names(x)

# Using the 'wt' argument to add a continuous value metric
# into an already one-hot encoded columns dataset (and hide tail)
csv <- "https://raw.githubusercontent.com/hms-dbmi/UpSetR/master/inst/extdata/movies.csv"
movies <- read.csv(csv, sep = ";")
head(movies)
freqs_list(movies,
  wt = AvgRating, min_elements = 2, tail = FALSE,
  title = "Movies\nMixed Genres\nRanking"
)
# So, please: no more Comedy+SciFi and more Drama+Horror films (based on ~50 movies)!

## End(Not run)
```

---

freqs_plot *Combined Frequencies Plot for Categorical Features*

Description

Plot frequencies of multiple categories within a data.frame in a new fancy way. Tidyverse friendly, based on `lares::freqs()`, no limits on amount of features to evaluate.

Usage

```
freqs_plot(
  df,
  ...,
  top = 10,
  rm.na = FALSE,
  abc = FALSE,
  title = NA,
  subtitle = NA
)
```

Arguments

df	Data.frame
...	Variables. Variables you wish to process. Order matters. If no variables are passed, the whole data.frame will be considered
top	Integer. Filter and plot the most n frequent for categorical values. Set to NA to return all values
rm.na	Boolean. Remove NA values in the plot? (not filtered for numerical output; use na.omit() or filter() if needed)
abc	Boolean. Do you wish to sort by alphabetical order?
title	Character. Overwrite plot's title with.
subtitle	Character. Overwrite plot's subtitle with.

Value

Plot. Result of the frequency of combined variables.

See Also

Other Frequency: freqs_df(), freqs_list(), freqs()

Other Exploratory: corr_cross(), corr_var(), crosstab(), df_str(), distr(), freqs_df(), freqs_list(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), plot_nums(), tree_var()

Other Visualization: distr(), freqs_df(), freqs_list(), freqs(), noPlot(), plot_chord(), plot_survey(), plot_timeline(), tree_var()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset

df <- freqs_plot(dft, Pclass, Survived)
head(df$data)
plot(df)

freqs_plot(dft, Pclass, Survived, Sex, Embarked)

freqs_plot(dft, Pclass, Survived, Sex, Embarked, top = 15)
```

---

gain_lift *Cumulative Gain, Lift and Response*

Description

This function calculates cumulative gain, lift, and response values for a predictive score of a specific target. You can use the mplot_gain() function to create a plot.

Usage

```
gain_lift(
  tag,
  score,
  target = "auto",
  splits = 10,
  plot = FALSE,
  quiet = FALSE
)
```

Arguments

tag	Vector. Real known label
score	Vector. Predicted value or model's result
target	Value. Which is your target positive value? If set to 'auto', the target with largest mean(score) will be selected. Change the value to overwrite. Only used when binary categorical model.
splits	Integer. Number of percentiles to split the data
plot	Boolean. Plot results? Uses `mplot_gain()`
quiet	Boolean. Quiet all messages, warnings, recommendations?

Value

data.frame when `plot=FALSE` or plot when `plot=TRUE`.

See Also

Other Machine Learning: `ROC()`, `conf_mat()`, `export_results()`, `h2o_automl()`, `h2o_predict_API()`, `h2o_predict_MOJO()`, `h2o_predict_binary()`, `h2o_predict_model()`, `h2o_selectmodel()`, `impute()`, `iter_seeds()`, `lasso_vars()`, `model_metrics()`, `model_preprocess()`, `msplit()`

Other Model metrics: `ROC()`, `conf_mat()`, `errors()`, `loglossBinary()`, `model_metrics()`

Examples

```
data(dfr) # Results for AutoML Predictions
head(dfr$class2)

# Results for Binomial Model
gain_lift(dfr$class2$tag, dfr$class2$scores, target = "FALSE")
gain_lift(dfr$class2$tag, dfr$class2$scores, target = "TRUE", splits = 5)
```

`get_credentials` *Load Credentials from a YML File*

Description

Load credentials from a local YML file. You can set your `.Renviron` and the `LARES_CREDS` parameter to remember (forever) the directory of your credentials' file. To use it later, you may leave `dir = NA`. You may also use this function for external (non-`lares`) code/use.

Usage

```
get_credentials(
  from = NA,
  dir = NA,
  filename = "config.yml",
  env = "LARES_CREDS",
  ...
)

get_creds(
  from = NA,
  dir = NA,
  filename = "config.yml",
  env = "LARES_CREDS",
  ...
)
```

Arguments

`from`	Character. Family of values to import from the YML file. If you don't know these names, set `from = NA` and a warning will display all possible values, depending on your YML file.
`dir`	Character. Credentials directory where your YML file is. If used frequently, set your directory by using the `.Renviron` file. To do so, leave `dir` as `NA` and follow the steps. If `dir` is a list, it'll return `dir` (manual credentials input).
`filename`	Character. YML filename with your credentials.
`env`	Character. Environment variable name. No need to set differently for any function that uses this library. Only for external use.
`...`	Additional parameters.

Value

List. Result of reading your credential's YML file, filtered by your `from` input if provided.

Set the default directory

The first time you use any function that has the `creds` parameter, if the `dir` parameter is set to `NA`, this function will ask you to set the directory where you save your YML local file with your credentials. This will be asked once and will be set for further R sessions. Remember to reset your session for this setup to start working properly.

YML file format

A YML file is a text file, with `.yml` file format. You may start from the dummy YML file shared which shows the structure you must follow to set your credentials file. Check it out here or find it locally using `system.file("docs", "config.yml", package = "lares")`.

See Also

Other Tools: `autoline()`, `bind_files()`, `bring_api()`, `db_download()`, `db_upload()`, `export_plot()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

Other Credentials: `db_download()`, `db_upload()`, `get_tweets()`, `mail_send()`, `queryDB()`, `queryGA()`, `slackSend()`, `stocks_file()`, `stocks_report()`

Examples

```
## Not run:
# Load dummy config.yml file from the library
# Recommendation: set dir with NA (read documentation)
# We need the directory, not the file
yml <- dirname(system.file("docs", "config.yml", package = "lares"))

# Let's see which credentials we have in our file
get_credentials(dir = yml)
# Warning message: No credentials for NA found in your YML file.
# Try any of the following: 'service1', 'service2', 'service3'

# Get credentials for service2
get_credentials("service2", dir = yml)

## End(Not run)
```

`get_currency` *Download Historical Currency Exchange Rate*

Description

This function lets the user download historical currency exchange rate between two currencies.

Usage

```
get_currency(
  currency_pair,
  from = Sys.Date() - 99,
  to = Sys.Date(),
  fill = FALSE
)
```

Arguments

currency_pair	Character. Which currency exchange do you wish to get the history from? i.e, USD/COP, EUR/USD...
from	Date. From date
to	Date. To date
fill	Boolean. Fill weekends and non-quoted dates with previous values?

Value

data.frame. Result of fetching online data for currency_pair grouped by date.

Examples

```
# For today (or any one single date)
get_currency("USD/COP", from = Sys.Date())
# For multiple dates
get_currency("EUR/USD", from = Sys.Date() - 7, fill = TRUE)
```

---

get_mp3 *Download MP3 from URL*

Description

This function downloads YouTube videos or Soundcloud or any other platform supported by the youtube-dl library, and converts them into high quality MP3 files. The URL can be for a single video or a whole playlist. It also returns metadata into an (invisible) list.

Usage

```
get_mp3(
  id,
  mp3 = TRUE,
  repo = "youtube-dl",
  params = "--no-check-certificate",
  start_time = 0,
  end_time = NA,
```

```
  overwrite = TRUE,
  info = TRUE,
  cover = FALSE,
  quiet = FALSE
)
```

Arguments

id	Character. YouTube URL or ID to search for.
mp3	Boolean. Add mp3 optimal parameters?
repo	Character. Chose repository you installed youtube-dl from. Any of: "youtube-dl" (latest stable version), "yt-dlp" (latest dev version).
params	Character. Additional parameters.
start_time, end_time	Numeric. Start and end time to trim the audio output in seconds.
overwrite	Boolean. Overwrite original file?
info	Boolean. Import and return metadata?
cover	Boolean. Google Search its squared cover?
quiet	Boolean. Keep quiet? If not, print messages.

Value

(Invisible) list with id's meta-data.

youtube-dl

More info from the original developers and its code: youtube-dl's Github

See Also

Other Scrapper: filesGD(), gtrends_related(), holidays(), ip_data(), readGS(), splot_etf(), stocks_quote()

Other Audio: trim_mp3()

Examples

```
# You must have "youtube-dl" installed in your OS:
## Not run:
# Download video from YouTube and convert to MP3
get_mp3("https://www.youtube.com/watch?v=lrlKcCdVw9Q")
# OR simply
get_mp3("lrlKcCdVw9Q")
# For dev version, use:
get_mp3("m3RX4LJh0iI", repo = "yt-dlp")

## End(Not run)
```

get_tweets *Get Tweets*

Description

This function downloads tweets with personal credentials

Usage

```
get_tweets(q, n = 10000, creds = NA)
```

Arguments

q	Query. Check for ?rtweet::search_tweets()
n	Integer. Total of tweets to return
creds	Character. Credential's user (see `get_creds()`)

Value

data.frame with API response results.

See Also

Other Credentials: `db_download()`, `db_upload()`, `get_credentials()`, `mail_send()`, `queryDB()`, `queryGA()`, `slackSend()`, `stocks_file()`, `stocks_report()`

gg_fill_customs *Custom fill, colour and text colours for ggplot2*

Description

This function lets the user use pre-defined default colours. Check your `lares_pal()$labels` scale. Feel free to use `gg_vals()` to debug colours used in latest plot.

Usage

```
gg_fill_customs(column = "fill", ...)

gg_colour_customs(column = "colour", ...)

gg_text_customs(column = "colour", ...)

gg_vals(layer = "fill", column = layer)
```

Arguments

column	Character. Select any of "fill" or "colour" to use on your lares_pal()$labels palette.
...	Allow additional parameters not used.
layer	Character. Select any of "fill", "colour", or "label" to get the layer containing the colours labels you wish to colour.

Details

Notice that when the layer defined is any of GeomPoint, GeomLine, GeomText or GeomLabel, gg_colour_customs() will force column = "fill" parameter.

Value

Same as scale_fill_manual or scale_colour_manual but with custom palette applied.

See Also

Other Themes: lares_pal(), plot_palette(), theme_lares()

Examples

```
library("ggplot2")
# Generic plot function to run examples to
run_plot <- function(add_fxs = TRUE) {
  p <- data.frame(station = c("spring", "summer", "fall", "winter"), num = 1:4) %>%
    ggplot(aes(x = station, y = num, fill = station)) +
    geom_col() +
    geom_text(aes(y = 0.5, label = num, colour = station), size = 6)
  if (add_fxs) p <- p + gg_fill_customs() + gg_colour_customs()
  return(p)
}
# Default colours
run_plot()
# Check last colours used
gg_vals("fill", "fill")
gg_vals("colour", "colour")
# Change any default colour
options("lares.colours.custom" = data.frame(
  values = c("summer", "winter"),
  fill = c("pink", "black"),
  colour = c("black", "white")
))
run_plot()
# Check last colours used
gg_vals("fill", "fill")
gg_vals("colour", "colour")
# Reset to default colours
options("lares.colours.custom" = NULL)
# Notice you can use 'pal = 4' argument on theme_lares() too
run_plot(add_fxs = FALSE) + theme_lares(pal = 4)
```

glued *Interpolate a string [glue wrapper]*

Description

Format and interpolate a string using a glue wrapper. Allows simple operations, NULL values as input, and interactions with internal (created within glued) and external (environment) objects.

Usage

```
glued(..., .sep = "", empty_lines = "keep", .envir = parent.frame())
```

Arguments

...	[expressions] Unnamed arguments are taken to be expression string(s) to format. Multiple inputs are concatenated together before formatting. Named arguments are taken to be temporary variables available for substitution.
.sep	[character(1): '""'] Separator used to separate elements.
empty_lines	Character. Set to "keep" to keep or "drop" to drop empty lines.
.envir	[environment: parent.frame()] Environment to evaluate each expression in. Expressions are evaluated from left to right. If .x is an environment, the expressions are evaluated in that environment and .envir is ignored. If NULL is passed, it is equivalent to emptyenv().

Value

Same as input but transformed (glued).

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
name <- "Bernardo"
age <- 29
anniversary <- as.Date("2016-04-30")
glued("
  My name is {name},
  my age next year will be {age + 1},
```

```
  and I got married on {format(anniversary, '%A, %B %d, %Y')}.")

# Single braces can be inserted by doubling them
glued("My name is {name}, not {{name}}.")

# You can also used named arguments
glued(
  "Her name is {name}, ",
  "and her age next year will be {age + 1}.",
  name = "Maru",
  age = 6
)

# And run operations with memories (beware!)
glued("My name, {name}, has {n <- nchar(name); n} characters.
       If we multiply by ten, we'll have {10 * n} characters!")

# If you pass a vector, the operation will be repeated for each element
glued("Here's the value #{1:3}")
```

---

gpt_ask *ChatGPT API Interaction with R*

Description

This function lets the user ask ChatGPT via its API, and returns the rendered reply. There are a couple of specific verbs (functions) with a preset prompt to help fetch the data in specific formats. We also store the prompts and replies in current session with their respective time-stamps so user can gather historical results.

Usage

```
gpt_ask(
  ask,
  secret_key = get_credentials()$openai$secret_key,
  url = Sys.getenv("LARES_GPT_URL"),
  model = Sys.getenv("LARES_GPT_MODEL"),
  quiet = FALSE,
  ...
)

gpt_history()

gpt_classify(x, categories, quiet = TRUE, ...)

gpt_tag(x, tags, quiet = TRUE, ...)

gpt_extract(x, extract, quiet = TRUE, ...)
```

```
gpt_format(x, format, quiet = TRUE, ...)

gpt_convert(x, unit, quiet = TRUE, ...)

gpt_table(ask, quiet = TRUE, ...)

gpt_translate(x, language, quiet = TRUE, ...)
```

Arguments

ask	Character. Redacted prompt to ask ChatGPT. If multiple asks are requested, they will be concatenated with "+" into a single request.
secret_key	Character. Secret Key. Get yours in: platform.openai.com.
url	Character. Base URL for OpenAI's ChatGPT API.
model	Character. OpenAI model to use.
quiet	Boolean. Keep quiet? If not, show informative messages.
...	Additional parameters.
x	Vector. List items you wish to process
categories, tags	Vector. List of possible categories/tags to consider.
extract, format, unit	Character. Length 1 or same as x to extract/format/unit information from x. For example: email, country of phone number, country, amount as number, currency ISO code, ISO, Fahrenheit, etc.
language	Character. Language to translate to

Value

(Invisible) list. Content returned from API POST and processed.

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token(), li_auth(), li_profile(), queryGA(), slackSend()

Examples

```
## Not run:
api_key <- get_credentials()$openai$secret_key
# Open question:
gpt_ask("Can you write an R function to plot a dummy histogram?", api_key)

##### The following examples return dataframes:
# Classify each element based on categories:
gpt_classify(1:10, c("odd", "even"))

# Add all tags that apply to each element based on tags:
```

```
gpt_tag(
  c("I love chocolate", "I hate chocolate", "I like Coke"),
  c("food", "positive", "negative", "beverage")
)

# Extract specific information:
gpt_extract(
 c("My mail is 123@test.com", "30 Main Street, Brooklyn, NY, USA", "+82 2-312-3456", "$1.5M"),
  c("email", "full state name", "country of phone number", "amount as number")
)

# Format values
gpt_format(
  c("March 27th, 2021", "12-25-2023 3:45PM", "01.01.2000", "29 Feb 92"),
  format = "ISO Date getting rid of timestamps"
)

# Convert units
gpt_convert(c("50C", "300K"), "Fahrenheit")

# Create a table with data
gpt_table("5 random people's address in South America, email, phone, age between 18-30")

# Translate text to any language
gpt_translate(
  rep("I love you with all my heart", 5),
  language = c("spanish", "chinese", "japanese", "russian", "german")
)

# Now let's read the historical prompts and replies from current session
gpt_history()

## End(Not run)
```

`grepl_letters` *Pattern Matching for Letters considering Blanks*

Description

Match pattern of letters considering blanks within each element of a character vector, allowing counted characters between and around each letter. Used as an auxiliary function for the Scrabble family of functions.

Usage

```
grepl_letters(x, pattern, blank = "_")
```

Arguments

x	Character vector
pattern	Character. Character string containing a semi-regular expression which uses the following logic: "a_b" means any character that contains "a" followed by something followed by "b", anywhere in the string.
blank	Character. String to use between letters.

Value

Boolean check for each value on x.

Examples

```
x <- c("aaaa", "bbbb", "baba", "aabb", "a", "ab")
grepl_letters(x, "ab")
grepl_letters(x, "_ab")
grepl_letters(x, "a_a")
grepl_letters(x, "c")
```

---

grepm *Pattern Matching for Any or All Multiple Matches*

Description

This function returns a boolean vector of the same length as ‘x‘, each element of which is the result of applying the ‘type‘ of matches to the corresponding element of ‘x‘, using regular expressions.

Usage

```
grepm(pattern, x, type = "all", ...)
```

Arguments

pattern	character string containing a regular expression (or character string for fixed = TRUE) to be matched in the given character vector. Coerced by as.character to a character string if possible. If a character vector of length 2 or more is supplied, the first element is used with a warning. Missing values are allowed except for regexpr, gregexpr and regexec.
x	Character vector. Text where matches are sought, or an object which can be coerced by as.character to a character vector. Long vectors are supported.
type	Character. Type of match. Choose one of: any, all
...	Additional arguments to pass to grepl

Value

Boolean of same length as x

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
x <- c(123, 876, 18761)
patterns <- c(1, 2)
grepm(patterns, x, type = "any")
grepm(patterns, x, type = "all")
```

---

gtrends_related *Google Trends: Related and Time Plots*

Description

This function creates a plot with Google Trend's related topics and queries, and let the user compare different keywords.

This function creates a plot with google trend's data on timelines and let the user compare different keywords.

Usage

```
gtrends_related(gtrend, top = NA, title = NA, note = NA, exclude = NULL)

gtrends_time(gtrend, title = NA)
```

Arguments

gtrend	List. Result from gtrendsR::gtrends(keyword, geo, time)
top	Integer. Filter top n results only.
title	Character. Custom title for the plot.
note	Character. Add a note to the plot if needed.
exclude	Character vector. Which observations do you wish to exclude?

Value

plot for Google Trend's results input gtrend.

Plot for Google Trend's results input gtrend.

See Also

Other Scrapper: filesGD(), get_mp3(), holidays(), ip_data(), readGS(), splot_etf(), stocks_quote()

Other Google: filesGD(), queryGA(), readGS()

---

h2o_automl *Automated H2O's AutoML*

Description

This function lets the user create a robust and fast model, using H2O's AutoML function. The result is a list with the best model, its parameters, datasets, performance metrics, variables importance, and plots. Read more about the h2o_automl() pipeline here.

Usage

```
h2o_automl(
  df,
  y = "tag",
  ignore = NULL,
  train_test = NA,
  split = 0.7,
  weight = NULL,
  target = "auto",
  balance = FALSE,
  impute = FALSE,
  no_outliers = TRUE,
  unique_train = TRUE,
  center = FALSE,
  scale = FALSE,
  thresh = 10,
  seed = 0,
  nfolds = 5,
  max_models = 3,
  max_time = 10 * 60,
  start_clean = FALSE,
  exclude_algos = c("StackedEnsemble", "DeepLearning"),
  include_algos = NULL,
  plots = TRUE,
  alarm = TRUE,
  quiet = FALSE,
  print = TRUE,
  save = FALSE,
  subdir = NA,
  project = "AutoML Results",
  verbosity = NULL,
  ...
```

```
)

## S3 method for class 'h2o_automl'
plot(x, ...)

## S3 method for class 'h2o_automl'
print(x, importance = TRUE, ...)
```

Arguments

df	Dataframe. Dataframe containing all your data, including the independent variable labeled as 'tag'. If you want to define which variable should be used instead, use the y parameter.
y	Variable or Character. Name of the independent variable.
ignore	Character vector. Force columns for the model to ignore
train_test	Character. If needed, df's column name with 'test' and 'train' values to split
split	Numeric. Value between 0 and 1 to split as train/test datasets. Value is for training set. Set value to 1 to train with all available data and test with same data (cross-validation will still be used when training). If train_test is set, value will be overwritten with its real split rate.
weight	Column with observation weights. Giving some observation a weight of zero is equivalent to excluding it from the dataset; giving an observation a relative weight of 2 is equivalent to repeating that row twice. Negative weights are not allowed.
target	Value. Which is your target positive value? If set to 'auto', the target with largest mean(score) will be selected. Change the value to overwrite. Only used when binary categorical model.
balance	Boolean. Auto-balance train dataset with under-sampling?
impute	Boolean. Fill NA values with MICE?
no_outliers	Boolean/Numeric. Remove y's outliers from the dataset? Will remove those values that are farther than n standard deviations from the independent variable's mean (Z-score). Set to TRUE for default (3) or numeric to set a different multiplier.
unique_train	Boolean. Keep only unique row observations for training data?
center, scale	Boolean. Using the base function scale, do you wish to center and/or scale all numerical values?
thresh	Integer. Threshold for selecting binary or regression models: this number is the threshold of unique values we should have in 'tag' (more than: regression; less than: classification)
seed	Integer. Set a seed for reproducibility. AutoML can only guarantee reproducibility if max_models is used because max_time is resource limited.
nfolds	Number of folds for k-fold cross-validation. Must be >= 2; defaults to 5. Use 0 to disable cross-validation; this will also disable Stacked Ensemble (thus decreasing the overall model performance).

`max_models, max_time`
: Numeric. Max number of models and seconds you wish for the function to iterate. Note that max_models guarantees reproducibility and max_time not (because it depends entirely on your machine's computational characteristics)

`start_clean`
: Boolean. Erase everything in the current h2o instance before we start to train models? You may want to keep other models or not. To group results into a custom common AutoML project, you may use `project_name` argument.

`exclude_algos, include_algos`
: Vector of character strings. Algorithms to skip or include during the model-building phase. Set NULL to ignore. When both are defined, only `include_algos` will be valid.

`plots`
: Boolean. Create plots objects?

`alarm`
: Boolean. Ping (sound) when done. Requires `beepr`.

`quiet`
: Boolean. Quiet all messages, warnings, recommendations?

`print`
: Boolean. Print summary when process ends?

`save`
: Boolean. Do you wish to save/export results into your working directory?

`subdir`
: Character. In which directory do you wish to save the results? Working directory as default.

`project`
: Character. Your project's name

`verbosity`
: Verbosity of the backend messages printed during training; Optional. Must be one of NULL (live log disabled), "debug", "info", "warn", "error". Defaults to "warn".

`...`
: Additional parameters on `h2o::h2o.automl`

`x`
: h2o_automl object

`importance`
: Boolean. Print important variables?

Value

List. Trained model, predicted scores and datasets used, performance metrics, parameters, importance data.frame, seed, and plots when `plots=TRUE`.

List of algorithms

-> Read more here

DRF Distributed Random Forest, including Random Forest (RF) and Extremely-Randomized Trees (XRT)

GLM Generalized Linear Model

XGBoost eXtreme Grading Boosting

GBM Gradient Boosting Machine

DeepLearning Fully-connected multi-layer artificial neural network

StackedEnsemble Stacked Ensemble

Methods

print Use `print` method to print models stats and summary

plot Use `plot` method to plot results using `mplot_full()`

See Also

Other Machine Learning: `ROC()`, `conf_mat()`, `export_results()`, `gain_lift()`, `h2o_predict_API()`, `h2o_predict_MOJO()`, `h2o_predict_binary()`, `h2o_predict_model()`, `h2o_selectmodel()`, `impute()`, `iter_seeds()`, `lasso_vars()`, `model_metrics()`, `model_preprocess()`, `msplit()`

Examples

```
## Not run:
# CRAN
data(dft) # Titanic dataset
dft <- subset(dft, select = -c(Ticket, PassengerId, Cabin))

# Classification: Binomial - 2 Classes
r <- h2o_automl(dft, y = Survived, max_models = 1, impute = FALSE, target = "TRUE", alarm = FALSE)

# Let's see all the stuff we have inside:
lapply(r, names)

# Classification: Multi-Categorical - 3 Classes
r <- h2o_automl(dft, Pclass, ignore = c("Fare", "Cabin"), max_time = 30, plots = FALSE)

# Regression: Continuous Values
r <- h2o_automl(dft, y = "Fare", ignore = c("Pclass"), exclude_algos = NULL, quiet = TRUE)
print(r)

# WITH PRE-DEFINED TRAIN/TEST DATAFRAMES
splits <- msplit(dft, size = 0.8)
splits$train$split <- "train"
splits$test$split <- "test"
df <- rbind(splits$train, splits$test)
r <- h2o_automl(df, "Survived", max_models = 1, train_test = "split")

## End(Not run)
```

---

h2o_explainer *DALEX Explainer for H2O*

Description

DALEX helper function to create an `explainer` object using a `h2o` trained model.

Usage

```
h2o_explainer(df, model, y = "tag", ignore = NULL, ...)
```

Arguments

df	Dataframe. Must contain all columns and predictions
model	Model object (H2O)
y	Character or Variable name. Variable's column name.
ignore	Character vector. Which columns should be ignored?
...	Additional parameters to pass to h2o_predict_model or h2o_predict_MOJO.

Value

List; explainer. Containing the model, data, y, predict_function, y_hat, residuals, class, label, model_info, residual_function, and weights.

See Also

Other Interpretability: dalex_local(), dalex_residuals(), dalex_variable()

Examples

```
# You must have "DALEX" library to use this auxiliary function:
## Not run:
data(dft) # Titanic dataset

# TRAIN A SIMPLE MODEL
dfm <- h2o_automl(dft,
  y = "Survived",
  ignore = c("Ticket", "PassengerId", "Cabin"),
  max_models = 1
)

# EXPLAINER
explainer <- h2o_explainer(df = dfm$datasets$test, model = dfm$model, y = "Survived")
explainer$data <- na.omit(explainer$data)

# CATEGORICAL EXAMPLE
class <- dalex_variable(explainer, vars = c("Pclass", "Sex"))
class$plot

# NUMERICAL EXAMPLE
num <- dalex_variable(explainer, vars = c("Fare", "Age"))
num$plot

# LOCAL EXAMPLE
local <- dalex_local(explainer, row = 1)
# OR YOU COULD MANUALLY INPUT THE OBSERVATION
local <- dalex_local(explainer, observation = explainer$data[1, ])
local$plot

# xai2shiny's UI (needs to be installed from ModelOriented/xai2shiny)
xai2shiny(explainer, run = TRUE)

## End(Not run)
```

h2o_predict_API *H2O Predict using API Service*

Description

This function lets the user get the score from an API service

Usage

```
h2o_predict_API(df, api, exclude = "tag")
```

Arguments

df	Dataframe/Vector. Data to insert into the model.
api	Character. API URL.
exclude	Character. Name of the variables to exclude.

Value

vector with predicted results.

See Also

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_MOJO(), h2o_predict_binary(), h2o_predict_model(), h2o_selectmodel(), impute(), iter_seeds(), lasso_vars(), model_metrics(), model_preprocess(), msplit()

Other H2O: h2o_predict_MOJO(), h2o_predict_binary(), h2o_predict_model()

h2o_predict_binary *H2O Predict using Binary file*

Description

This function lets the user predict using the h2o binary file. Note that it works with the files generated when using the function export_results(). Recommendation: use the h2o_predict_MOJO() function when possible - it let's you change h2o's version without problem.

Usage

```
h2o_predict_binary(df, model_path, sample = NA)
```

Arguments

df	Dataframe. Data to insert into the model.
model_path	Character. Relative model path directory or zip file.
sample	Integer. How many rows should the function predict?

Value

vector with predicted results.

See Also

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_model(), h2o_selectmodel(), impute(), iter_seeds(), lasso_vars(), model_metrics(), model_preprocess(), msplit()

Other H2O: h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_model()

---

h2o_predict_model *H2O Predict using H2O Model Object*

Description

This function lets the user get scores from a H2O Model Object.

Usage

```
h2o_predict_model(df, model)
```

Arguments

df	Dataframe/Vector. Data to insert into the model.
model	h2o model Object

Value

data.frame with predicted results.

See Also

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_binary(), h2o_selectmodel(), impute(), iter_seeds(), lasso_vars(), model_metrics(), model_preprocess(), msplit()

Other H2O: h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_binary()

h2o_predict_MOJO *H2O Predict using MOJO file*

Description

This function lets the user predict using the h2o .zip file containing the MOJO files. Note that it works with the files generated when using the function export_results()

Usage

```
h2o_predict_MOJO(df, model_path, method = "mojo", batch = 300)
```

Arguments

df	Dataframe. Data to pass to the model.
model_path	Character. Relative path of directory where your zip model file is. If multiple zip files are found, first one found will be used.
method	Character. One of "mojo" or "json".
batch	Integer. Run n batches at a time for "json" method.

Value

data.frame with predicted results.

See Also

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_binary(), h2o_predict_model(), h2o_selectmodel(), impute(), iter_seeds(), lasso_vars(), model_metrics(), model_preprocess(), msplit()

Other H2O: h2o_predict_API(), h2o_predict_binary(), h2o_predict_model()

h2o_results *Automated H2O's AutoML Results*

Description

This is an auxiliary function to calculate predictions and results when using the h2o_automl() function.

Usage

```
h2o_results(
  h2o_object,
  test,
  train,
  y = "tag",
  which = 1,
  model_type,
  target = "auto",
  split = 0.7,
  ignore = NULL,
  quiet = FALSE,
  project = "ML Project",
  seed = 0,
  leaderboard = list(),
  plots = TRUE,
  ...
)
```

Arguments

h2o_object	H2O Leaderboard (H2OFrame/H2OAutoML) or Model (h2o)
test, train	Dataframe. Must have the same columns
y	Variable or Character. Name of the independent variable.
which	Integer. Which model to select from leaderboard
model_type	Character. Select "Classification" or "Regression"
target	Value. Which is your target positive value? If set to 'auto', the target with largest mean(score) will be selected. Change the value to overwrite. Only used when binary categorical model.
split	Numeric. Value between 0 and 1 to split as train/test datasets. Value is for training set. Set value to 1 to train with all available data and test with same data (cross-validation will still be used when training). If train_test is set, value will be overwritten with its real split rate.
ignore	Character vector. Columns too ignore
quiet	Boolean. Quiet all messages, warnings, recommendations?
project	Character. Your project's name
seed	Integer. Set a seed for reproducibility. AutoML can only guarantee reproducibility if max_models is used because max_time is resource limited.
leaderboard	H2O's Leaderboard. Passed when using h2o_selectmodel as it contains plain model and no leader board.
plots	Boolean. Create plots objects?
...	Additional parameters on h2o::h2o.automl

Value

List. Trained model, predicted scores and datasets used, performance metrics, parameters, importance data.frame, seed, and plots when `plots=TRUE`.

`h2o_selectmodel` *Select Model from h2o_automl's Leaderboard*

Description

Select wich model from the h2o_automl function to use

Usage

```
h2o_selectmodel(results, which_model = 1, quiet = FALSE, ...)
```

Arguments

`results`	`h2o_automl()` object.
`which_model`	Integer. Which model from the leaderboard you wish to use?
`quiet`	Boolean. Quiet all messages, warnings, recommendations?
`...`	Additional parameters on `h2o::h2o.automl`

Value

H2O processed model

See Also

Other Machine Learning: `ROC()`, `conf_mat()`, `export_results()`, `gain_lift()`, `h2o_automl()`, `h2o_predict_API()`, `h2o_predict_MOJO()`, `h2o_predict_binary()`, `h2o_predict_model()`, `impute()`, `iter_seeds()`, `lasso_vars()`, `model_metrics()`, `model_preprocess()`, `msplit()`

Other Tools: `autoline()`, `bind_files()`, `bring_api()`, `db_download()`, `db_upload()`, `export_plot()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

h2o_shap *SHAP values for H2O Models*

Description

SHAP (SHapley Additive exPlanations) by Lundberg and Lee (2016) is a method to explain individual predictions. SHAP is based on the game theoretically optimal Shapley Values. Calculate SHAP values for h2o models in which each row is an observation and each column a feature. Use `plot` method to visualize features importance and distributions.

Usage

```
h2o_shap(model, test = "auto", scores = "auto", y = "y", ...)

## S3 method for class 'h2o_shap'
plot(x, relevant = TRUE, top = 15, quiet = FALSE, ...)
```

Arguments

`model`	`h2o_automl` object or `h2o` model.
`test`	String or Dataframe. Leave "auto" to use `h2o_automl`'s test dataset or pass a valid dataframe.
`scores`	Numeric vector. If test != "auto", you must provide predicted values
`y`	Character. If test != "auto", you must provide y variable's name
`...`	Additional argument for `predict_contributions.H2OModel`
`x`	h2o_shap object
`relevant`	Boolean. Keep only relevant non-trivial (>0) features
`top`	Integer. Plot only top n values (as in importance)
`quiet`	Boolean. Print messages?

Value

H2OFrame with shap values for every observation and feature.

See Also

Other SHAP: `shap_var()`

Examples

```
## Not run:
# Train a h2o_automl model
model <- h2o_automl(dft, Survived,
  max_models = 1, target = TRUE,
  ignore = c("Ticket", "Cabin", "PassengerId"),
  quiet = TRUE
```

```
)

# Calculate SHAP values
SHAP_values <- h2o_shap(model)
# Equivalent to:
# SHAP_values <- h2o_shap(
#  model = model$model,
#  test = model$datasets$test,
#  scores = model$scores_test$scores)

# Check SHAP results
head(SHAP_values)

# You must have "ggbeeswarm" library to use this auxiliary function:
# Plot SHAP values (feature importance)
plot(SHAP_values)

# Plot some of the variables (categorical)
shap_var(SHAP_values, Pclass)

# Plot some of the variables (numerical)
shap_var(SHAP_values, Fare)

## End(Not run)
```

`haveInternet` *Internet Connection Check*

Description

This function checks if your R session currently have Wifi or Internet connection.

Usage

```
haveInternet(thresh = 3, url = "http://www.google.com")
```

Arguments

`thresh`	Numeric. How many seconds to consider a slow connection?
`url`	Character. URL to test the readLines 1 command

Value

Boolean. Result of checking if device has internet connection.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

---

holidays *Holidays in your Country*

Description

This function lets the user automatically scrap holiday dates from any country and year within +- 5 years. Thanks to timeanddate.com!

Usage

```
holidays(countries = "Venezuela", years = year(Sys.Date()))
```

Arguments

countries	Character or vector. For which country(ies) should the holidays be imported?
years	Character or vector. For which year(s) do you wish to import holiday dates?

Value

data.frame with holidays data for given countries and years.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Feature Engineering: date_feats(), ohse()

Other Scrapper: filesGD(), get_mp3(), gtrends_related(), ip_data(), readGS(), splot_etf(), stocks_quote()

Other One Hot Encoding: date_feats(), ohe_commas(), ohse()

Examples

```
holidays(countries = "Argentina")
holidays(countries = c("Argentina", "Venezuela"), years = c(2019, 2020))
```

image_metadata *Get Meta Data from Image Files*

Description

This function lets the user get meta data from image files or directory.

Usage

```
image_metadata(files)
```

Arguments

files	Character vector. Files or directory which contains files.

Value

data.frame with meta-data for each image file.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

importxlsx *Import Excel File with All Its Tabs*

Description

This function lets the user import an Excel file's tabs into a list

Usage

```
importxlsx(file)
```

Arguments

file	String. Local Excel file name

Value

List or data.frame. If single tab is found, a data.frame; if multiple tabs are found on file, a list of data.frames.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

`impute` *Impute Missing Values (using MICE)*

Description

This function uses the MICE methodology to impute missing values.

Usage

```
impute(df, m = 5, iters = 5, seed = 0, quiet = FALSE)
```

Arguments

`df`	Dataframe. Dataframe to transform.
`m`	Integer. Number of multiple imputations.
`iters`	Integer. Number of iterations.
`seed`	Integer. Set a seed for reproducibility.
`quiet`	Boolean. Keep quiet? (or print replacements).

Value

data.frame with imputed values.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_binary(), h2o_predict_model(), h2o_selectmodel(), iter_seeds(), lasso_vars(), model_metrics(), model_preprocess(), msplit()

Other Missing Values: missingness()

install_recommended *Install/Update Additional Recommended Libraries*

Description

All needed libraries to use (most) lares are already a dependency. There are some functions that many people won't event know exist that will require other additional libraries. Also, this may be used as a Docker way of installing useful libraries on an new instance.

Usage

```
install_recommended(progress = TRUE, all = FALSE)
```

Arguments

progress	Boolean. Show status bar?
all	Boolean. All packages? If not, only the ones not installed yet.

ip_data *Scrap data based on IP address*

Description

This function lets the user scrap https://db-ip.com/ given IP address(es) to get their associated address type, ASN, ISP, organization, country, state or region, county, city, ZIP postal code, weather station, coordinates, Timezone, local time, languages, and currency.

Usage

```
ip_data(ip = myip(), quiet = FALSE)
```

Arguments

ip	Vector. Vector with all IP's we wish to search.
quiet	Boolean. Do not show the loading statusbar?

Value

data.frame. Each row is an unique ip address, and columns will bee created for all the additional information found.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Other Scrapper: filesGD(), get_mp3(), gtrends_related(), holidays(), readGS(), splot_etf(), stocks_quote()

Examples

```
ip_data("163.114.132.0")
ip_data(ip = c(myip(), "201.244.197.199"), quiet = TRUE)
```

---

is_url *Check if input is_* or are_**

Description

Check whether a value or vector is or is not following a set of rules. For example: is an URL, is an ID vector, are non-variant or constant values, are binary values... Notice that `is_` will return the result for each observation and `are_` for the whole vector.

Usage

```
is_url(x, ...)

is_ip(x, ...)

are_id(x)

are_constant(x)

are_binary(x)
```

Arguments

x	Vector
...	Additional parameters passed to `grepl()`

Value

is_url. Boolean. Result of checking if x is a valid URL string.

is_ip. Boolean. Result of checking if x is a valid IP string.

are_id. Boolean. Result of checking if x is a potential ID vector

are_constant. Boolean. Result of checking if x is a constant vector

are_binary. Boolean. Result of checking if x is a binary vector

Examples

```
is_url(c("google.com", "http://google.com"))

is_ip(c("163.114.132.0", "7.114.132", "0.0.0.0", "1.1.1.1."))

are_id(1:10)
are_id(LETTERS[1:10])

are_constant(rep(1, 10))
are_constant(1:10)

are_binary(c("A", "B", "A"))
```

---

iter_seeds *Iterate Seeds on AutoML*

Description

This functions lets the user iterate and search for best seed. Note that if the results change a lot, you are having a high variance in your data.

Usage

```
iter_seeds(df, y, tries = 10, ...)
```

Arguments

df	Dataframe. Dataframe containing all your data, including the independent variable labeled as 'tag'. If you want to define which variable should be used instead, use the y parameter.
y	Variable or Character. Name of the independent variable.
tries	Integer. Number of iterations
...	Additional arguments passed to h2o_automl

Value

data.frame with performance results by seed tried on every row.

See Also

Other Machine Learning: `ROC()`, `conf_mat()`, `export_results()`, `gain_lift()`, `h2o_automl()`, `h2o_predict_API()`, `h2o_predict_MOJO()`, `h2o_predict_binary()`, `h2o_predict_model()`, `h2o_selectmodel()`, `impute()`, `lasso_vars()`, `model_metrics()`, `model_preprocess()`, `msplit()`

---

json2vector *Convert Python JSON string to R vector (data.frame with 1 row)*

Description

This function lets the user transform a JSON string into vector (data.frame with 1 row). You can also pass a Python's dictionary. For any other JSON transformation, `jsonlite` is recommended.

Usage

```
json2vector(json)
```

Arguments

`json`	Character. JSON string.

Value

List, data.frame, or vector. Depends on the `json` string.

See Also

Other Tools: `autoline()`, `bind_files()`, `bring_api()`, `db_download()`, `db_upload()`, `export_plot()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

Examples

```
json2vector('{"id": 1, "nodata": null, "gender": "M"}')
```

lares *Analytics, Data Mining & Machine Learning Sidekick*

Description

R library for better/faster analytics, visualization, data mining, and machine learning tasks.

Author(s)

Bernardo Lares (laresbernardo@gmail.com)

See Also

Useful links:

- https://github.com/laresbernardo/lares
- https://laresbernardo.github.io/lares/
- Report bugs at https://github.com/laresbernardo/lares/issues

lares-exports *Pipe operator*

Description

Pipe operator

lares_logo *Print lares R library Logo*

Description

Used "ASCII Art Generator" from manytools.org to convert logo to ASCII. Visit.

Usage

```
lares_logo(version = TRUE)
```

Arguments

version	Boolean. Include R and lares version?

Examples

```
lares_logo()
```

`lares_pal` *Personal Colours Palette*

Description

Fetch customizable palettes for the library's usage. The package has its own default colour-blind friendly colours but can be customized using R internal options (i.e. `options("lares.palette" = c("#FF8303" = "#000","#40A4D8" = "#FFF", ...))`. There are 3 options you can use to customize all colour palletes: "lares.palette" (vector, will be used in the same order as passed, and must have a counter colour defined), "lares.colours" (vector, simple colour names and their HEX codes), and "lares.colours.custom" (data.frame, containing "values" to use dynamically, "fill" for main colour, and "colour" (not obligatory) for counter colour).

Usage

```
lares_pal(return = "list")
```

Arguments

`return`	Character. Get only what you need. Select any of: "all" or "list" (list), "colors" or "colours" (vector), "pal" or "palette" (named vector), "simple" (named vector), "custom" or "personal" (data.frame)

Value

Depending on the `return` input, we get a:

- `vector` with `palette` results vector
- `vector` with `palette` results vector's names
- `list` with `palette` results vector, `labels` results data.frame, and `simple` results named vector

See Also

Other Themes: `gg_fill_customs()`, `plot_palette()`, `theme_lares()`

Examples

```
# Simple colour-named palette
lares_pal("simple")

# Raw colours and counter-colours
# OR simply: lares_pal("palette")
nice_palette <- lares_pal("colours")
nice_palette_ctr <- as.vector(lares_pal()$palette)
lapply(list(nice_palette, nice_palette_ctr), head)

# Personal colours by name
df <- lares_pal("custom")
df[sample(nrow(df), 5), ]
```

lasso_vars *Most Relevant Features Using Lasso Regression*

Description

Use Lasso regression to identify the most relevant variables that can predict/identify another variable. You might want to compare with `corr_var()` and/or `x2y()` results to compliment the analysis No need to standardize, center or scale your data. Tidyverse friendly.

Usage

```
lasso_vars(
  df,
  variable,
  ignore = NULL,
  nlambdas = 100,
  nfolds = 10,
  top = 20,
  quiet = FALSE,
  seed = 123,
  ...
)
```

Arguments

`df`	Dataframe. Any dataframe is valid as `ohse` will be applied to process categorical values, and values will be standardize automatically.
`variable`	Variable. Independent variable.
`ignore`	Character vector. Variables to exclude from study.
`nlambdas`	Integer. Number of lambdas to be used in a search.
`nfolds`	Integer. Number of folds for K-fold cross-validation (>= 2).
`top`	Integer. Plot top n results only.
`quiet`	Boolean. Keep quiet? Else, show messages.
`seed`	Numeric.
`...`	Additional parameters passed to `ohse()`.

Value

List. Contains lasso model coefficients, performance metrics, the actual model fitted and a plot.

See Also

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_binary(), h2o_predict_model(), h2o_selectmodel(), impute(), iter_seeds(), model_metrics(), model_preprocess(), msplit()

Other Exploratory: corr_cross(), corr_var(), crosstab(), df_str(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), missingness(), plot_cats(), plot_df(), plot_nums(), tree_var()

Examples

```
## Not run:
# CRAN
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset

m <- lasso_vars(dft, Survived, ignore = c("Cabin"))
print(m$coef)
print(m$metrics)
plot(m$plot)

## End(Not run)
```

---

left *Left or Right N characters of a string*

Description

This functions lets the user extract the first or last n characters of a string or vector of strings.

Usage

```
left(string, n = 1)

right(string, n = 1)
```

Arguments

string	String or Vector.
n	Integer. How many characters starting on right/left?

Value

Character. Trimmed strings.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
left("Bernardo", 3)
right(c("Bernardo", "Lares", "V"), 3)
```

---

listfiles *List files in a directory*

Description

This function lets the user list all files on a given directory. It also lets filter files which contains a string.

Usage

```
listfiles(folder = getwd(), recursive = TRUE, regex = NA, images = FALSE)
```

Arguments

folder	Character. Directory which contains files
recursive	Boolean. Should the listing recurse into directories?
regex	Character. String to use for filtering files
images	Boolean. Bring only image files?

Value

data.frame with relevant data for each file on folder directory.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
# All files in current directory (without recursive files)
df <- listfiles(recursive = TRUE)
head(df, 3)

# All files in current directory (with recursive files)
df <- listfiles(recursive = TRUE)
tail(df, 3)

# Check R files using regex
df <- listfiles(regex = "\\.R$")
```

`list_cats` *List categorical values for data.frame*

Description

Make a list with all categorical values and

Usage

```
list_cats(df, ..., abc = TRUE)
```

Arguments

df	data.frame
...	Variables to segment counters
abc	Boolean. Sort alphabetically?

Value

List. Length same as number of categorical columns, each with a frequency data.frame using `freqs()`.

See Also

Other Tools: `autoline()`, `bind_files()`, `bring_api()`, `db_download()`, `db_upload()`, `export_plot()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `updateLares()`, `warnifnot()`, `what_size()`

Examples

```
data(dft) # Titanic dataset
df <- dft[, 1:5]
head(df)
list_cats(df)
```

li_auth *OAuth Linkedin*

Description

This function authenticates and creates a token for LinkedIn's API REST

Usage

```
li_auth(app_name = NA, client_id = NA, client_secret = NA)
```

Arguments

app_name	Character. Your App's given name.
client_id	Character. Your App's client ID.
client_secret	Character. Your App's client secret.

Value

Character. String with token requested.

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token(), gpt_ask(), li_profile(), queryGA(), slackSend()

Other LinkedIn: li_profile()

li_profile *Get My Personal LinkedIn Data*

Description

This function brings a list with your personal LinkedIn data

Usage

```
li_profile(token = NA)
```

Arguments

token	Object. OAuth Authentication: li_auth()'s output.

Value

List. Results of your own profile data given the token.

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token(), gpt_ask(), li_auth(), queryGA(), slackSend()

Other LinkedIn: li_auth()

---

loglossBinary *Logarithmic Loss Function for Binary Models*

Description

This function calculates log loss/cross-entropy loss for binary models. NOTE: when result is 0.69315, the classification is neutral; it assigns equal probability to both classes.

Usage

```
loglossBinary(tag, score, eps = 0.001)
```

Arguments

tag	Vector. Real known label
score	Vector. Predicted value or model's result
eps	Numeric. Epsilon value

See Also

Other Model metrics: ROC(), conf_mat(), errors(), gain_lift(), model_metrics()

---

mail_send *Send Emails with Attachments (POST)*

Description

This function lets the user send Emails with Attachments using MailGun's API service.

Usage

```
mail_send(
  from = "RMail <laresbernardo@gmail.com>",
  to = "laresbernardo@gmail.com",
  cc = NULL,
  bcc = NULL,
  subject = "Mail from R",
  text = " \n",
  html = NULL,
```

```
  attachment = NULL,
  service = "mailgun",
  creds = NULL,
  quiet = FALSE,
  ...
)
```

Arguments

from, to, cc, bcc	Character. Emails
subject	Character. Subject for the email.
text, html	Character. Text or HTML to send in the body.
attachment	Character, plot or data.frame. Will send the file, plot as PNG or data.frame as CSV, respectively.
service	Character. Service platform to search on creds.
creds	Character. Credential's user (see get_creds()). Must contain: url (POST address), api (API key).
quiet	Boolean. Keep quite or display messages?
...	Additional parameters.

Value

No return value, called for side effects.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Other Credentials: db_download(), db_upload(), get_credentials(), get_tweets(), queryDB(), queryGA(), slackSend(), stocks_file(), stocks_report()

Examples

```
## Not run:
myPlot <- noPlot("My plot")
mail_send(
  from = "BLV <myuser@mail.com>",
  to = "youruser@mail.com",
  cc = "myuser@mail.com",
  subject = paste("Daily report:", Sys.Date()),
  attachment = myPlot
)

## End(Not run)
```

markdown2df *Convert markdown string tables to data.frame*

Description

Convert markdown string tables to data.frame

Usage

```
markdown2df(text)
```

Arguments

text	Character. Markdown text representing a table.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
txt <- "| Item | Value |\n|------|-------|\n| 50C  | 122F  |\n| 300K | 80.33F |"
markdown2df(txt)
```

missingness *Calculate and Visualize Missingness*

Description

This function lets the user calculate the percentage of NAs or missingness in a data.frame. It also plots the results if needed.

Usage

```
missingness(df, plot = FALSE, full = FALSE, subtitle = NA, summary = TRUE)
```

Arguments

df	Dataframe. Dataframe to study
plot	Boolean. Do you wish to plot results?
full	Boolean. Return all variables (or only with missings)?
subtitle	Character. Subtitle to show in plot
summary	Boolean. Show numerical summary text?

Value

data.frame with each variable, number of missing values and percentage. If `plot=TRUE`, a plot with the same information reflected.

See Also

Other Exploratory: `corr_cross()`, `corr_var()`, `crosstab()`, `df_str()`, `distr()`, `freqs_df()`, `freqs_list()`, `freqs_plot()`, `freqs()`, `lasso_vars()`, `plot_cats()`, `plot_df()`, `plot_nums()`, `tree_var()`

Other Missing Values: `impute()`

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal

# Dummy data
df <- data.frame(
  A = c(1:5),
  B = c(NA, NA, 1, 1, 1),
  C = rep(NA, 5),
  D = c(NA, LETTERS[1:4])
)

# Missing values summary
missingness(df)

# Visual results

missingness(df, plot = TRUE)

# Show all variables (including those with no missing values)
missingness(df, plot = TRUE, full = TRUE)
```

model_metrics *Model Metrics and Performance*

Description

This function lets the user get a confusion matrix and accuracy, and for for binary classification models: AUC, Precision, Sensitivity, and Specificity, given the expected (tags) values and predicted values (scores).

Usage

```
model_metrics(
  tag,
  score,
  multis = NA,
  abc = TRUE,
  thresh = 10,
  auto_n = TRUE,
  thresh_cm = 0.5,
  target = "auto",
  type = "test",
  model_name = NA,
  plots = TRUE,
  quiet = FALSE,
  subtitle = NA
)
```

Arguments

tag	Vector. Real known label
score	Vector. Predicted value or model's result
multis	Data.frame. Containing columns with each category score (only used when more than 2 categories coexist)
abc	Boolean. Arrange columns and rows alphabetically when categorical values?
thresh	Integer. Threshold for selecting binary or regression models: this number is the threshold of unique values we should have in 'tag' (more than: regression; less than: classification)
auto_n	Add n_ before digits when it's categorical and not numerical, even though seems numerical?
thresh_cm	Numeric. Value to splits the results for the confusion matrix. Range of values: (0-1)
target	Value. Which is your target positive value? If set to 'auto', the target with largest mean(score) will be selected. Change the value to overwrite. Only used when binary categorical model.
type	Character. One of: "train", "test".

model_name	Character. Model's name for reference.
plots	Boolean. Create plots objects?
quiet	Boolean. Quiet all messages, warnings, recommendations?
subtitle	Character. Subtitle for plots

Value

List. Multiple performance metrics that vary depending on the type of model (classification or regression). If plot=TRUE, multiple plots are also returned.

See Also

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_binary(), h2o_predict_model(), h2o_selectmodel(), impute(), iter_seeds(), lasso_vars(), model_preprocess(), msplit()

Other Model metrics: ROC(), conf_mat(), errors(), gain_lift(), loglossBinary()

Other Calculus: corr(), dist2d(), quants()

Examples

```
data(dfr) # Results for AutoML Predictions
lapply(dfr, head)

# Metrics for Binomial Model
met1 <- model_metrics(dfr$class2$tag, dfr$class2$scores,
  model_name = "Titanic Survived Model",
  plots = FALSE
)
print(met1)

# Metrics for Multi-Categorical Model
met2 <- model_metrics(dfr$class3$tag, dfr$class3$score,
  multis = subset(dfr$class3, select = -c(tag, score)),
  model_name = "Titanic Class Model",
  plots = FALSE
)
print(met2)

# Metrics for Regression Model
met3 <- model_metrics(dfr$regr$tag, dfr$regr$score,
  model_name = "Titanic Fare Model",
  plots = FALSE
)
print(met3)
```

model_preprocess *Automate Data Preprocess for Modeling*

Description

Pre-process your data before training a model. This is the prior step on the h2o_automl() function's pipeline. Enabling for other use cases when wanting too use any other framework, library, or custom algorithm.

Usage

```
model_preprocess(
  df,
  y = "tag",
  ignore = NULL,
  train_test = NA,
  split = 0.7,
  weight = NULL,
  target = "auto",
  balance = FALSE,
  impute = FALSE,
  no_outliers = TRUE,
  unique_train = TRUE,
  center = FALSE,
  scale = FALSE,
  thresh = 10,
  seed = 0,
  quiet = FALSE
)
```

Arguments

df	Dataframe. Dataframe containing all your data, including the independent variable labeled as 'tag'. If you want to define which variable should be used instead, use the y parameter.
y	Character. Column name for independent variable.
ignore	Character vector. Force columns for the model to ignore
train_test	Character. If needed, df's column name with 'test' and 'train' values to split
split	Numeric. Value between 0 and 1 to split as train/test datasets. Value is for training set. Set value to 1 to train with all available data and test with same data (cross-validation will still be used when training). If train_test is set, value will be overwritten with its real split rate.
weight	Column with observation weights. Giving some observation a weight of zero is equivalent to excluding it from the dataset; giving an observation a relative weight of 2 is equivalent to repeating that row twice. Negative weights are not allowed.

target	Value. Which is your target positive value? If set to 'auto', the target with largest mean(score) will be selected. Change the value to overwrite. Only used when binary categorical model.
balance	Boolean. Auto-balance train dataset with under-sampling?
impute	Boolean. Fill NA values with MICE?
no_outliers	Boolean/Numeric. Remove y's outliers from the dataset? Will remove those values that are farther than n standard deviations from the independent variable's mean (Z-score). Set to TRUE for default (3) or numeric to set a different multiplier.
unique_train	Boolean. Keep only unique row observations for training data?
center, scale	Boolean. Using the base function scale, do you wish to center and/or scale all numerical values?
thresh	Integer. Threshold for selecting binary or regression models: this number is the threshold of unique values we should have in 'tag' (more than: regression; less than: classification)
seed	Integer. Set a seed for reproducibility. AutoML can only guarantee reproducibility if max_models is used because max_time is resource limited.
quiet	Boolean. Quiet all messages, warnings, recommendations?

Value

List. Contains original data.frame df, an index to identify which observations with be part of the train dataset train_index, and which model type should be model_type.

See Also

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_binary(), h2o_predict_model(), h2o_selectmodel(), impute(), iter_seeds(), lasso_vars(), model_metrics(), msplit()

Examples

```
data(dft) # Titanic dataset

model_preprocess(dft, "Survived", balance = TRUE)

model_preprocess(dft, "Fare", split = 0.5, scale = TRUE)

model_preprocess(dft, "Pclass", ignore = c("Fare", "Cabin"))

model_preprocess(dft, "Pclass", quiet = TRUE)
```

move_files *Move files from A to B*

Description

Move one or more files from a directory to another using R.

Usage

```
move_files(from, to)
```

Arguments

from	Character. File names and directories. All files will be moved recursively.
to	Character. File names for each from file or directory. If directory does not exist, it will be created.

Value

No return value, called for side effects.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

mplot_conf *Confussion Matrix Plot*

Description

This function plots a confussion matrix.

Usage

```
mplot_conf(
  tag,
  score,
  thresh = 0.5,
  abc = TRUE,
  squared = FALSE,
```

```
  diagonal = TRUE,
  top = 20,
  subtitle = NA,
  model_name = NULL,
  save = FALSE,
  subdir = NA,
  file_name = "viz_conf_mat.png"
)
```

Arguments

tag	Vector. Real known label.
score	Vector. Predicted value or model's result.
thresh	Integer. Threshold for selecting binary or regression models: this number is the threshold of unique values we should have in 'tag' (more than: regression; less than: classification)
abc	Boolean. Arrange columns and rows alphabetically?
squared	Boolean. Force plot to be squared?
diagonal	Boolean. FALSE to convert diagonal numbers to zeroes. Ideal to detect must confusing categories.
top	Integer. Plot only the most n frequent variables. Set to NA to plot all.
subtitle	Character. Subtitle to show in plot
model_name	Character. Model's name
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot

Details

You may use conf_mat() to get calculate values.

Value

Plot with confusion matrix results.

See Also

Other ML Visualization: mplot_cuts_error(), mplot_cuts(), mplot_density(), mplot_full(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
lapply(dfr, head)

# Plot for Binomial Model
mplot_conf(dfr$class2$tag, dfr$class2$scores,
  model_name = "Titanic Survived Model"
)

# Plot for Multi-Categorical Model
mplot_conf(dfr$class3$tag, dfr$class3$score,
  model_name = "Titanic Class Model"
)
```

`mplot_cuts` *Cuts by quantiles for score plot*

Description

This function cuts by quantiles any score or prediction.

Usage

```
mplot_cuts(
  score,
  splits = 10,
  model_name = NA,
  subtitle = NA,
  table = FALSE,
  save = FALSE,
  subdir = NA,
  file_name = "viz_ncuts.png"
)
```

Arguments

`score`	Vector. Predicted value or model's result.
`splits`	Integer. Numer of separations to plot
`model_name`	Character. Model's name
`subtitle`	Character. Subtitle to show in plot
`table`	Boolean. Do you wish to return a table with results?
`save`	Boolean. Save output plot into working directory
`subdir`	Character. Sub directory on which you wish to save the plot
`file_name`	Character. File name as you wish to save the plot

Value

Plot with performance results by cuts.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_density(), mplot_full(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
head(dfr$class2)

# Data
mplot_cuts(dfr$class2$scores, splits = 5, table = TRUE)

# Plot
mplot_cuts(dfr$class2$scores, model_name = "Titanic Survived Model")
```

---

mplot_cuts_error *Cuts by quantiles on absolute and percentual errors plot*

Description

This function cuts by quantiles on absolute and percentual errors

Usage

```
mplot_cuts_error(
  tag,
  score,
  splits = 10,
  title = NA,
  model_name = NA,
  save = FALSE,
  subdir = NA,
  file_name = "viz_ncuts_error.png"
)
```

Arguments

tag	Vector. Real known label.
score	Vector. Predicted value or model's result.
splits	Integer. Number of separations to plot
title	Character. Title to show in plot

model_name	Character. Model's name
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot

Value

Plot with error results by cuts.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts(), mplot_density(), mplot_full(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
head(dfr$regr)
mplot_cuts_error(dfr$regr$tag, dfr$regr$score,
  model_name = "Titanic Fare Model"
)
```

---

mplot_density *Density plot for discrete and continuous values*

Description

This function plots discrete and continuous values results

Usage

```
mplot_density(
  tag,
  score,
  thresh = 6,
  model_name = NA,
  subtitle = NA,
  save = FALSE,
  subdir = NA,
  file_name = "viz_distribution.png"
)
```

Arguments

tag	Vector. Real known label
score	Vector. Predicted value or model's result
thresh	Integer. Threshold for selecting binary or regression models: this number is the threshold of unique values we should have in 'tag' (more than: regression; less than: classification)
model_name	Character. Model's name
subtitle	Character. Subtitle to show in plot
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot

Value

Plot with distribution and performance results.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_cuts(), mplot_full(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
lapply(dfr[c(1, 3)], head)

# Plot for binomial results
mplot_density(dfr$class2$tag, dfr$class2$scores, subtitle = "Titanic Survived Model")

# Plot for regression results
mplot_density(dfr$regr$tag, dfr$regr$score, model_name = "Titanic Fare Model")
```

---

mplot_full *MPLOTS Score Full Report Plots*

Description

This function plots a whole dashboard with a model's results. It will automatically detect if it's a categorical or regression's model by checking how many different unique values the independent variable (tag) has.

Usage

```
mplot_full(
  tag,
  score,
  multis = NA,
  splits = 8,
  thresh = 6,
  subtitle = NA,
  model_name = NA,
  plot = TRUE,
  save = FALSE,
  subdir = NA,
  file_name = "viz_full.png"
)
```

Arguments

tag	Vector. Real known label.
score	Vector. Predicted value or model's result.
multis	Data.frame. Containing columns with each category probability or score (only used when more than 2 categories coexist).
splits	Integer. Number of separations to plot
thresh	Integer. Threshold for selecting binary or regression models: this number is the threshold of unique values we should have in 'tag' (more than: regression; less than: classification)
subtitle	Character. Subtitle to show in plot
model_name	Character. Model's name
plot	Boolean. Plot results? If not, plot grid object returned
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot

Value

Multiple plots gathered into one, showing tag vs score performance results.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_cuts(), mplot_density(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
lapply(dfr, head)

# Dasboard for Binomial Model
mplot_full(dfr$class2$tag, dfr$class2$scores,
  model_name = "Titanic Survived Model"
)

# Dasboard for Multi-Categorical Model
mplot_full(dfr$class3$tag, dfr$class3$score,
  multis = subset(dfr$class3, select = -c(tag, score)),
  model_name = "Titanic Class Model"
)

# Dasboard for Regression Model
mplot_full(dfr$regr$tag, dfr$regr$score,
  model_name = "Titanic Fare Model"
)
```

---

mplot_gain *Cumulative Gain Plot*

Description

The cumulative gains plot, often named 'gains plot', helps us answer the question: When we apply the model and select the best X deciles, what expect to target? The cumulative gains chart shows the percentage of the overall number of cases in a given category "gained" by targeting a percentage of the total number of cases.

Usage

```
mplot_gain(
  tag,
  score,
  multis = NA,
  target = "auto",
  splits = 10,
  highlight = "auto",
  caption = NA,
  save = FALSE,
  subdir = NA,
  file_name = "viz_gain.png",
  quiet = FALSE
)
```

Arguments

tag	Vector. Real known label.
score	Vector. Predicted value or model's result.
multis	Data.frame. Containing columns with each category probability or score (only used when more than 2 categories coexist).
target	Value. Which is your target positive value? If set to 'auto', the target with largest mean(score) will be selected. Change the value to overwrite. Only works for binary classes
splits	Integer. Numer of quantiles to split the data
highlight	Character or Integer. Which split should be used for the automatic conclussion in the plot? Set to "auto" for best value, "none" to turn off or the number of split.
caption	Character. Caption to show in plot
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot
quiet	Boolean. Do not show message for auto target?

Value

Plot with gain and performance results by cuts.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_cuts(), mplot_density(), mplot_full(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
lapply(dfr, head)

# Plot for Binomial Model
mplot_gain(dfr$class2$tag, dfr$class2$scores,
  caption = "Titanic Survived Model",
  target = "FALSE"
)
mplot_gain(dfr$class2$tag, dfr$class2$scores,
  caption = "Titanic Survived Model",
  target = "TRUE"
)

# Plot for Multi-Categorical Model
mplot_gain(dfr$class3$tag, dfr$class3$score,
  multis = subset(dfr$class3, select = -c(tag, score)),
  caption = "Titanic Class Model"
)
```

`mplot_importance` *Variables Importances Plot*

Description

This function plots Variable Importances

Usage

```
mplot_importance(
  var,
  imp,
  colours = NA,
  limit = 15,
  model_name = NA,
  subtitle = NA,
  save = FALSE,
  subdir = NA,
  file_name = "viz_importance.png"
)
```

Arguments

`var`	Vector. Variable or column's names
`imp`	Vector. Importance of said variables. Must have same length as var
`colours`	If positive and negative contribution is known
`limit`	Integer. Limit how many variables you wish to plot
`model_name`	Character. Model's name
`subtitle`	Character. Subtitle to show in plot
`save`	Boolean. Save output plot into working directory
`subdir`	Character. Sub directory on which you wish to save the plot
`file_name`	Character. File name as you wish to save the plot

Value

Plot with ranked importance variables results.

See Also

Other ML Visualization: `mplot_conf()`, `mplot_cuts_error()`, `mplot_cuts()`, `mplot_density()`, `mplot_full()`, `mplot_gain()`, `mplot_lineal()`, `mplot_metrics()`, `mplot_response()`, `mplot_roc()`, `mplot_splits()`, `mplot_topcats()`

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
df <- data.frame(
  variable = LETTERS[1:6],
  importance = c(4, 6, 6.7, 3, 4.8, 6.2) / 100,
  positive = c(TRUE, TRUE, FALSE, TRUE, FALSE, FALSE)
)
head(df)

mplot_importance(
  var = df$variable,
  imp = df$importance,
  model_name = "Random values model"
)

# Add a colour for categories
mplot_importance(
  var = df$variable,
  imp = df$importance,
  colours = df$positive,
  limit = 4
)
```

---

mplot_lineal *Linear Regression Results Plot*

Description

This function plots a Linear Regression Result

Usage

```
mplot_lineal(
  tag,
  score,
  subtitle = NA,
  model_name = NA,
  save = FALSE,
  subdir = NA,
  file_name = "viz_lineal.png"
)
```

Arguments

tag	Vector. Real known label.
score	Vector. Predicted value or model's result.
subtitle	Character. Subtitle to show in plot
model_name	Character. Model's name

save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot

Value

Plot with linear distribution and performance results.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_cuts(), mplot_density(), mplot_full(), mplot_gain(), mplot_importance(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
lapply(dfr, head)
mplot_lineal(dfr$regr$tag, dfr$regr$score, model_name = "Titanic Fare Model")
```

---

mplot_metrics *Model Metrics and Performance Plots*

Description

This function generates plots of the metrics of a predictive model. This is an auxiliary function used in model_metrics() when the parameter plot is set to TRUE.

Usage

```
mplot_metrics(
  results,
  subtitle = NA,
  model_name = NA,
  save = FALSE,
  subdir = NA,
  file_name = "viz_metrics.png"
)
```

Arguments

results	Object. Results object from h2o_automl function
subtitle	Character. Subtitle to show in plot
model_name	Character. Model's name
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot

Value

Plot with `results` performance.

See Also

Other ML Visualization: `mplot_conf()`, `mplot_cuts_error()`, `mplot_cuts()`, `mplot_density()`, `mplot_full()`, `mplot_gain()`, `mplot_importance()`, `mplot_lineal()`, `mplot_response()`, `mplot_roc()`, `mplot_splits()`, `mplot_topcats()`

`mplot_response` *Cumulative Response Plot*

Description

The response gains plot helps us answer the question: When we apply the model and select up until ntile X, what is the expected

Usage

```
mplot_response(
  tag,
  score,
  multis = NA,
  target = "auto",
  splits = 10,
  highlight = "auto",
  caption = NA,
  save = FALSE,
  subdir = NA,
  file_name = "viz_response.png",
  quiet = FALSE
)
```

Arguments

`tag`	Vector. Real known label.
`score`	Vector. Predicted value or model's result.
`multis`	Data.frame. Containing columns with each category probability or score (only used when more than 2 categories coexist).
`target`	Value. Which is your target positive value? If set to 'auto', the target with largest mean(score) will be selected. Change the value to overwrite. Only works for binary classes
`splits`	Integer. Numer of quantiles to split the data
`highlight`	Character or Integer. Which split should be used for the automatic conclussion in the plot? Set to "auto" for best value, "none" to turn off or the number of split.

caption	Character. Caption to show in plot
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot
quiet	Boolean. Do not show message for auto target?

Value

Plot with cumulative response and performance results by cuts.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_cuts(), mplot_density(), mplot_full(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_roc(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
lapply(dfr, head)

# Plot for Binomial Model
mplot_response(dfr$class2$tag, dfr$class2$scores,
  caption = "Titanic Survived Model",
  target = "TRUE"
)
mplot_response(dfr$class2$tag, dfr$class2$scores,
  caption = "Titanic Survived Model",
  target = "FALSE"
)

# Plot for Multi-Categorical Model
mplot_response(dfr$class3$tag, dfr$class3$score,
  multis = subset(dfr$class3, select = -c(tag, score)),
  caption = "Titanic Class Model"
)
```

---

mplot_roc *ROC Curve Plot*

Description

This function plots ROC Curves with AUC values with 95% confidence range. It also works for multi-categorical models.

Usage

```
mplot_roc(
  tag,
  score,
  multis = NA,
  sample = 1000,
  model_name = NA,
  subtitle = NA,
  interval = 0.2,
  squared = TRUE,
  plotly = FALSE,
  save = FALSE,
  subdir = NA,
  file_name = "viz_roc.png"
)
```

Arguments

tag	Vector. Real known label.
score	Vector. Predicted value or model's result.
multis	Data.frame. Containing columns with each category probability or score (only used when more than 2 categories coexist).
sample	Integer. Number of samples to use for rendering plot.
model_name	Character. Model's name
subtitle	Character. Subtitle to show in plot
interval	Numeric. Interval for breaks in plot
squared	Boolean. Keep proportions?
plotly	Boolean. Use plotly for plot's output for an interactive plot
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot

Value

Plot with ROC curve and AUC performance results.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_cuts(), mplot_density(), mplot_full(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_splits(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
lapply(dfr[c(1, 2)], head)

# ROC Curve for Binomial Model
mplot_roc(dfr$class2$tag, dfr$class2$scores,
  model_name = "Titanic Survived Model"
)

# ROC Curves for Multi-Categorical Model
mplot_roc(dfr$class3$tag, dfr$class3$score,
  multis = subset(dfr$class3, select = -c(tag, score)),
  squared = FALSE,
  model_name = "Titanic Class Model"
)
```

---

mplot_splits *Split and compare quantiles plot*

Description

This function lets us split and compare quantiles on a given prediction to compare different categorical values vs scores grouped by equal sized buckets.

Usage

```
mplot_splits(
  tag,
  score,
  splits = 5,
  subtitle = NA,
  model_name = NA,
  save = FALSE,
  subdir = NA,
  file_name = "viz_splits.png"
)
```

Arguments

tag	Vector. Real known label.
score	Vector. Predicted value or model's result.
splits	Integer. Number of separations to plot
subtitle	Character. Subtitle to show in plot
model_name	Character. Model's name
save	Boolean. Save output plot into working directory
subdir	Character. Sub directory on which you wish to save the plot
file_name	Character. File name as you wish to save the plot

Value

Plot with distribution and performance results by splits.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_cuts(), mplot_density(), mplot_full(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_topcats()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
lapply(dfr, head)

# For categorical (binary) values
mplot_splits(dfr$class2$tag, dfr$class2$scores,
  splits = 4,
  model_name = "Titanic Survived Model"
)

# For categorical (+2) values
mplot_splits(dfr$class3$tag, dfr$class2$scores,
  model_name = "Titanic Class Model"
)

# For continuous values
mplot_splits(dfr$regr$tag, dfr$regr$score,
  splits = 4,
  model_name = "Titanic Fare Model"
)
```

---

mplot_topcats *Top Hit Ratios for Multi-Classification Models*

Description

Calculate and plot a multi-class model's predictions accuracy based on top N predictions and distribution of probabilities.

Usage

```
mplot_topcats(tag, score, multis, model_name = NA)
```

Arguments

tag	Vector. Real known label.
score	Vector. Predicted value or model's result.
multis	Data.frame. Containing columns with each category probability or score (only used when more than 2 categories coexist).
model_name	Character. Model's name

Value

Plot with performance results over most frequent categories.

See Also

Other ML Visualization: mplot_conf(), mplot_cuts_error(), mplot_cuts(), mplot_density(), mplot_full(), mplot_gain(), mplot_importance(), mplot_lineal(), mplot_metrics(), mplot_response(), mplot_roc(), mplot_splits()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dfr) # Results for AutoML Predictions
mplot_topcats(dfr$class3$tag, dfr$class3$score,
  multis = subset(dfr$class3, select = -c(tag, score)),
  model_name = "Titanic Class Model"
)
```

---

msplit *Split a dataframe for training and testing sets*

Description

This function splits automatically a dataframe into train and test datasets. You can define a seed to get the same results every time, but has a default value. You can prevent it from printing the split counter result.

Usage

```
msplit(df, size = 0.7, seed = 0, print = TRUE)
```

Arguments

df	Dataframe
size	Numeric. Split rate value, between 0 and 1. If set to 1, the train and test set will be the same.
seed	Integer. Seed for random split
print	Boolean. Print summary results?

Value

List with both datasets, summary, and split rate.

See Also

Other Machine Learning: ROC(), conf_mat(), export_results(), gain_lift(), h2o_automl(), h2o_predict_API(), h2o_predict_MOJO(), h2o_predict_binary(), h2o_predict_model(), h2o_selectmodel(), impute(), iter_seeds(), lasso_vars(), model_metrics(), model_preprocess()

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
data(dft) # Titanic dataset
splits <- msplit(dft, size = 0.7, seed = 123)
names(splits)
```

---

myip *What's my IP?*

Description

Reveal your current IP address.

Usage

```
myip()
```

Value

Character. Result of your IP address based on ipify.org

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
myip()
```

---

ngrams *Build N-grams and keep most frequent*

Description

Build out n-grams for multiple text inputs and keep the n most frequent combinations.

Usage

```
ngrams(text, ngram = c(2, 3), top = 10, stop_words = NULL, ...)
```

Arguments

text	Character vector
ngram	Integer vector. Number of continuous n items in text.
top	Integer. Keep n most frequent ngrams only.
stop_words	Character vector. Words to exclude from text. Example: if you want to exclude "a", whenever that word appears it will be excluded, but when the letter "a" appears in a word, it will remain.
...	Additional parameters passed to remove_stopwords.

Value

data.frame with ngrams and counters, sorted by frequency.

See Also

Other Text Mining: cleanText(), remove_stopwords(), replaceall(), sentimentBreakdown(), textCloud(), textFeats(), textTokenizer(), topics_rake()

Examples

```
# You must have "tidytext" library to use this auxiliary function:
## Not run:
women <- read.csv("https://bit.ly/3mXJOOi")
x <- women$description
ngrams(x, ngram = c(2, 3), top = 3)
ngrams(x, ngram = 2, top = 6, stop_words = c("a", "is", "of", "the"))

## End(Not run)
```

noPlot *Plot Result with Nothing to Plot*

Description

This function lets the user print a plot without plot, with a customizable message. It is quite useful for Shiny renderPlot when using filters and no data is returned.

Usage

```
noPlot(message = "Nothing to show here!", size = 4.5, ...)
```

Arguments

message	Character. What message do you wish to show?
size	Numeric. Font size for message input.
...	Additional parameters passed to theme_lares().

Value

Empty ggplot2 object (with a message if set).

See Also

Other Visualization: distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), plot_chord(), plot_survey(), plot_timeline(), tree_var()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
noPlot(message = "No plot to show!")
noPlot(background = "#FF5500", size = 7)
```

normalize *Normalize Vector*

Description

This function lets the user normalize numerical values into the 0 to 1 range

Usage

```
normalize(x)
```

Arguments

x	Numeric Vector. Numbers to be transformed into normalized vector

Value

Vector with normalized x values

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
x <- c(0, 1, 4, 7.5, 10)
normalize(x)
```

---

num_abbr *Abbreviate numbers*

Description

This function converts a numeric vector's values into their abbreviated character equivalent, i.e. 100,000,000 into 100M.

Usage

```
num_abbr(x, n = 3)
```

Arguments

x	Numeric vector
n	Integer. Single numeric value, specifying number of significant figures to show. Range 1 to 6.

Value

Vector of character values that contain converted values

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
num_abbr(rnorm(10) * 1e6)
num_abbr(rnorm(10) * 1e6, n = 1)
```

ohe_commas *One Hot Encoding for a Vector with Comma Separated Values*

Description

This function lets the user do one hot encoding on a variable with comma separated values

Usage

```
ohe_commas(df, ..., sep = ",", noval = "NoVal", remove = FALSE)
```

Arguments

df	Dataframe. May contain one or more columns with comma separated values which will be separated as one hot encoding
...	Variables. Which variables to split into new columns?
sep	Character. Which regular expression separates the elements?
noval	Character. No value text
remove	Boolean. Remove original variables?

Value

data.frame on which all features are numerical by nature or transformed with one hot encoding.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other One Hot Encoding: date_feats(), holidays(), ohse()

Examples

```
df <- data.frame(
  id = c(1:5),
  x = c("AA, D", "AA,B", "B,  D", "A,D,B", NA),
  z = c("AA+BB+AA", "AA", "BB,  AA", NA, "BB+AA")
)
ohe_commas(df, x, remove = TRUE)
ohe_commas(df, z, sep = "\\+")
ohe_commas(df, x, z)
```

ohse *One Hot Smart Encoding (Dummy Variables)*

Description

This function lets the user automatically transform a dataframe with categorical columns into numerical by one hot encoding technic.

Usage

```
ohse(
  df,
  redundant = FALSE,
  drop = TRUE,
  ignore = NULL,
  dates = FALSE,
  holidays = FALSE,
  country = "Venezuela",
  currency_pair = NA,
  trim = 0,
  limit = 10,
  variance = 0.9,
  other_label = "OTHER",
  sep = "_",
  quiet = FALSE,
  ...
)
```

Arguments

df	Dataframe
redundant	Boolean. Should we keep redundant columns? i.e. If the column only has two different values, should we keep both new columns? Is set to NULL, only binary variables will dump redundant columns.
drop	Boolean. Drop automatically some useless features?
ignore	Vector or character. Which column should be ignored?
dates	Boolean. Do you want the function to create more features out of the date/time columns?
holidays	Boolean. Include holidays as new columns?
country	Character or vector. For which countries should the holidays be included?
currency_pair	Character. Which currency exchange do you wish to get the history from? i.e, USD/COP, EUR/USD...
trim	Integer. Trim names until the nth character
limit	Integer. Limit one hot encoding to the n most frequent values of each column. Set to NA to ignore argument.

variance	Numeric. Drop columns with more than n variance. Range: 0-1. For example: if a variable contains 91 unique different values out of 100 observations, this column will be suppressed if value is set to 0.9
other_label	Character. With which text do you wish to replace the filtered values with?
sep	Character. Separator's string
quiet	Boolean. Quiet all messages and summaries?
...	Additional parameters

Value

data.frame on which all features are numerical by nature or transformed with one hot encoding.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Feature Engineering: date_feats(), holidays()

Other One Hot Encoding: date_feats(), holidays(), ohe_commas()

Examples

```
data(dft)
dft <- dft[, c(2, 3, 5, 9, 11)]

ohse(dft, limit = 3) %>% head(3)
ohse(dft, limit = 3, redundant = NULL) %>% head(3)

# Getting rid of columns with no (or too much) variance
dft$no_variance1 <- 0
dft$no_variance2 <- c("A", rep("B", nrow(dft) - 1))
dft$no_variance3 <- as.character(rnorm(nrow(dft)))
dft$no_variance4 <- c(rep("A", 20), round(rnorm(nrow(dft) - 20), 4))
ohse(dft, limit = 3) %>% head(3)
```

---

outlier_turkey *Outliers: Tukey's fences*

Description

Tukey's fences is a technique used in box plots. The non-outlier range is defined with [Q1-k(Q3-Q1), Q3+k(Q3-Q1)], where Q1 and Q3 are the lower and upper quartiles respectively, k - some non-negative constant (popular choice is 1.5). A value is an outlier based on Tukey's fences when its value does not lie in non-outlier range.

Usage

```
outlier_turkey(x, k = 1.5)
```

Arguments

x	Numeric. Distribution
k	Positive Numeric. K-multiplier.

Value

Boolean vector detecting outliers.

See Also

Other Outliers: outlier_zscore_plot(), outlier_zscore(), winsorize()

---

outlier_zscore *Outliers: Z-score method*

Description

Z-score, also called a standard score, of an observation is a distance from the population center measured in number of normalization units. The default choice for center is sample mean and for normalization unit is standard deviation. Values are considered outliers based on z-score if its absolute value of default z-score is higher then the threshold (popular choice is 3).

Usage

```
outlier_zscore(x, thresh = 3, mad = FALSE)
```

Arguments

x	Numeric. Distribution
thresh	Numeric. Z-Score threshold for n standard deviations.
mad	Boolean. Use median absolute deviation instead?

Value

data.frame. Each row is an x observation with its respective std/mean or mad/med calculations depending on mad input.

See Also

Other Outliers: outlier_turkey(), outlier_zscore_plot(), winsorize()

`outlier_zscore_plot` *Outliers: Z-score method plot*

Description

Test several Z-score thresholds to visualize outliers. Tidyverse friendly.

Usage

```
outlier_zscore_plot(df, var, group = NULL, thresh = c(2, 3, 5), top = 5)
```

Arguments

`df`	Dataframe.
`var`	Numeric variable.
`group`	Categorical variable. Grouping variable.
`thresh`	Numeric vector. Z-Score threshold for n standard deviations.
`top`	Integer. Show only n most frequent categorical values when using the `group` argument.

Value

ggplot2 object

See Also

Other Outliers: `outlier_turkey()`, `outlier_zscore()`, `winsorize()`

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset
outlier_zscore_plot(dft, Fare)
p <- outlier_zscore_plot(dft, Fare, Pclass, thresh = c(3, 5))
plot(p)
attr(p, "z_values")
head(attr(p, "labels"))
```

plot_cats *Plot All Categorical Features (Frequencies)*

Description

This function filters categorical columns and plots the frequency for each value on every feature.

Usage

```
plot_cats(df)
```

Arguments

df	Dataframe

Value

Plot. Result of df categorical features.

See Also

Other Exploratory: corr_cross(), corr_var(), crosstab(), df_str(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_df(), plot_nums(), tree_var()

plot_chord *Chords Plot*

Description

This auxiliary function plots discrete and continuous values results

Usage

```
plot_chord(
  origin,
  dest,
  weight = 1,
  mg = 3,
  title = "Chord Diagram",
  subtitle = "",
  pal = NA
)
```

Arguments

origin, dest	Vectors. Origin and destination vectors
weight	Vector. Weight for each chord.
mg	Numeric. Margin adjust for plot in case of need
title	Character. Title for the plot
subtitle	Character. Subtitle for the plot
pal	Vector. Colour pallete. Order matters.

Value

chordDiagram object

See Also

Other Visualization: distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), noPlot(), plot_survey(), plot_timeline(), tree_var()

Examples

```
# You must have "circlize" library to use this auxiliary function:
## Not run:
df <- data.frame(from = c(1, 1, 2, 3, 4, 1, 6), to = c(4, 4, 4, 2, 2, NA, NA))
plot_chord(df$from, df$to)

## End(Not run)
```

---

plot_df *Plot Summary of Numerical and Categorical Features*

Description

This function plots all columns frequencies and boxplots, for categorical and numerical respectively.

Usage

```
plot_df(df)
```

Arguments

df	Dataframe

Value

Plot. Result of df categorical and numerical features.

See Also

Other Exploratory: corr_cross(), corr_var(), crosstab(), df_str(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_nums(), tree_var()

---

plot_nums *Plot All Numerical Features (Boxplots)*

Description

This function filters numerical columns and plots boxplots.

Usage

```
plot_nums(df)
```

Arguments

df	Dataframe

Value

Plot. Result of df numerical features.

See Also

Other Exploratory: corr_cross(), corr_var(), crosstab(), df_str(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), tree_var()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data(dft) # Titanic dataset
plot_nums(dft)
```

plot_palette *Plot Palette Colours*

Description

This function plots a list of colours

Usage

```
plot_palette(fill, colour = "black", id = NA, limit = 12)
```

Arguments

fill	Vector. List of colours for fills.
colour	Vector. List of colours for colours.
id	Vector. ID for each color.
limit	Integer. Show only first n values.

Value

Plot with `fill` colours and `colour` counter-colours if provided.

See Also

Other Themes: `gg_fill_customs()`, `lares_pal()`, `theme_lares()`

Examples

```
# Simply pass a vector
pal <- lares_pal("simple")
plot_palette(pal)
# Or fill + color named vector
pal <- lares_pal("pal")
plot_palette(fill = names(pal), colour = as.vector(pal))
```

plot_survey *Visualize Survey Results*

Description

This function lets the user plot a survey's result.

Usage

```
plot_survey(answers, ignore = 1, title = NA, subtitle = NA)
```

Arguments

answers	Dataframe. Answers. Each row a different person. Each column a different answer.
ignore	Numeric Vector. Which columns are NOT answers?
title	Character. Title for your plot
subtitle	Character. Subtitle for your plot.

Value

ggplot2 object

See Also

Other Visualization: distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), noPlot(), plot_chord(), plot_timeline(), tree_var()

---

plot_timeline *Plot timeline as Gantt Plot*

Description

This function plots groups of observartions with timelines in a Gantt Plot way. Only works if start and end are date format values.

Usage

```
plot_timeline(
  event,
  start,
  end = start + 1,
  label = NA,
  group = NA,
  title = "Curriculum Vitae Timeline",
  subtitle = "Bernardo Lares",
  interactive = FALSE,
  save = FALSE,
  subdir = NA
)
```

Arguments

event	Vector. Event, role, label, or row.
start	Vector. Start date.
end	Vector. End date. Only one day be default if not defined
label	Vector. Place, institution, or label.

group	Vector. Academic, Work, Extracurricular... Pass as factor to keep a specific order
title	Character. Title for the plot
subtitle	Character. Subtitle for the plot
interactive	Boolean. Run with plotly?
save	Boolean. Save the output plot in our working directory
subdir	Character. Into which subdirectory do you wish to save the plot to?

Value

ggplot2 object

See Also

Other Visualization: distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), noPlot(), plot_chord(), plot_survey(), tree_var()

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
cols <- c("Role", "Place", "Type", "Start", "End")
today <- as.character(Sys.Date())
cv <- data.frame(rbind(
  c("Marketing Science Partner", "Facebook", "Work Experience", "2019-12-09", today),
  c("Data Scientist Consultant", "MatrixDS", "Work Experience", "2018-09-01", today),
  c("R Community Contributor", "lares library", "Extra", "2018-07-18", today),
  c("Lead Data Scientist", "MEG", "Work Experience", "2019-01-15", "2019-12-09"),
 c("Head of Analytics", "Comparamejor/R5", "Work Experience", "2016-08-01", "2019-01-15"),
  c("Big Data & Data Science Programme", "UdC", "Academic", "2017-09-01", "2018-02-28"),
  c("Project Engineer", "Polytex", "Work Experience", "2016-05-15", "2016-09-01"),
  c("Big Data Analyst", "MEG", "Work Experience", "2016-01-01", "2016-04-30"),
  c("Advanced Excel Instructor", "ARTS", "Work Experience", "2015-11-01", "2016-04-30"),
 c("Continuous Improvement Intern", "PAVCO", "Work Experience", "2015-04-01", "2015-08-30"),
 c("Mechanical Design Intern", "SIGALCA", "Work Experience", "2013-07-01", "2013-09-30"),
 c("DJs Online Community Owner", "LaresDJ.com / SoloParaDJs", "Extra", "2010-01-05", "2020-05-20"),
  c("Mechanical Engineer Degree", "USB", "Academic", "2009-09-15", "2015-11-20"),
  c("DJ and Composer/Producer", "Legacy Discplay", "Extra", "2009-05-01", "2015-04-30")
))
colnames(cv) <- cols
plot_timeline(
  event = cv$Role,
  start = cv$Start,
  end = cv$End,
  label = cv$Place,
  # Simple trick to re-arrange the grids
  group = factor(cv$Type, levels = c("Work Experience", "Academic", "Extra"))
)
```

prophesize *Facebook's Prophet Forecast*

Description

Prophet is Facebook's procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

Usage

```
prophesize(
  df,
  n_future = 60,
  country = NULL,
  trend.param = 0.05,
  logged = FALSE,
  pout = 0.03,
  project = "Prophet Forecast"
)
```

Arguments

df	Data frame. Must contain date/time column and values column, in that order.
n_future	Integer. How many steps do you wish to forecast?
country	Character. Country code for holidays.
trend.param	Numeric. Flexibility of trend component. Default is 0.05, and as this value becomes larger, the trend component will be more flexible.
logged	Boolean. Convert values into logs?
pout	Numeric. Get rid of pout % of outliers.
project	Character. Name of your forecast project for plot title

Details

Official documentation: https://github.com/facebook/prophet

Value

List. Containing the forecast results, the prophet model, and a plot.

See Also

Other Forecast: forecast_arima()

quants *Calculate cuts by quantiles*

Description

This function lets the user quickly calculate cuts for quantiles and discretize numerical values into categorical values.

Usage

```
quants(values, splits = 10, return = "labels", n = 2)
```

Arguments

values	Vector. Values to calculate quantile cuts
splits	Integer. How many cuts should split the values?
return	Character. Return "summary" or "labels"
n	Integer. Determines the number of digits used in formatting the break numbers.

Value

Factor vector or data.frame. Depending on `return` input:

- `labels` a factor ordered vector with each observation's quantile
- `summary` a data.frame with information on each quantile cut

See Also

Other Data Wrangling: `balance_data()`, `categ_reducer()`, `cleanText()`, `date_cuts()`, `date_feats()`, `file_name()`, `formatHTML()`, `holidays()`, `impute()`, `left()`, `normalize()`, `num_abbr()`, `ohe_commas()`, `ohse()`, `removenacols()`, `replaceall()`, `replacefactor()`, `textFeats()`, `textTokenizer()`, `vector2text()`, `year_month()`, `zerovar()`

Other Calculus: `corr()`, `dist2d()`, `model_metrics()`

Examples

```
data(dft) # Titanic dataset
quants(dft$Age, splits = 5, "summary")
quants(dft$Age, splits = 5, "labels")[1:10]
```

queryDB *PostgreSQL Queries on Database (Read)*

Description

This function lets the user query a PostgreSQL database. Previously was called queryDummy but was replaced and deprecated for a more general function by using the from parameter.

Usage

```
queryDB(query, from, creds = NA)
```

Arguments

query	Character. SQL Query
from	Character. Credential's user (see get_creds())
creds	Character. Credential's directory (see get_creds())

Value

data.frame. Result of fetching the query data.

See Also

Other Credentials: db_download(), db_upload(), get_credentials(), get_tweets(), mail_send(), queryGA(), slackSend(), stocks_file(), stocks_report()

queryGA *Queries on Google Analytics*

Description

This function lets the user query Google Analytics with its API. More about the documentation and parameters in googleAnalyticsR::google_analytics() or Google Analytics' API

Usage

```
queryGA(
  account,
  creds = NA,
  token_dir = NA,
  metrics = "sessions",
  dimensions = "date",
  met_filters = NULL,
  dim_filters = NULL,
  start = lubridate::floor_date(Sys.Date(), "month"),
  end = Sys.Date()
)
```

Arguments

account	Character. Personal named accounts
creds	Character. Credential's user (see get_creds())
token_dir	Character. Credential's directory (see get_creds())
metrics	Character. Which metrics we wish to bring
dimensions	Character. Which dimensions we wish to bring
met_filters, dim_filters	A filter_clause_ga4 for filtering metrics/dimensions. Check googleAnalyticsR::google_analytic
start	Date. Start date for the report
end	Date. End date for the report

Value

data.frame with the API GET request tabulated results.

See Also

Other Credentials: db_download(), db_upload(), get_credentials(), get_tweets(), mail_send(), queryDB(), slackSend(), stocks_file(), stocks_report()

Other Google: filesGD(), gtrends_related(), readGS()

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token(), gpt_ask(), li_auth(), li_profile(), slackSend()

---

quiet *Quiet prints and verbose noise*

Description

This function silences (verbose) output prints. Thanks to Hadley Wickham for bringing the idea.

Usage

```
quiet(fx, quiet = TRUE)
```

Arguments

fx	Function to quiet
quiet	Quiet outputs? If not, skip quietness.

Value

Same as fx but with no messages or prints.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

---

read.file *Read Files Quickly (Auto-detected)*

Description

This function lets the user import csv, xlsx, xls, sav files.

Usage

```
read.file(filename, current_wd = TRUE, sheet = 1, quiet = FALSE)
```

Arguments

filename	Character. File name to import.
current_wd	Boolean. Use current working directory before the file's name? Use this param to NOT get absolute root directory.
sheet	Character. Name or index of the sheet to read data from if file is xlsx or xls.
quiet	Boolean. Quiet summary message?

Value

List or data.frame, depending on filename's data.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), statusbar(), tic(), try_require(), updateLares(), warnifnot(), what_size()

readGS *Google Sheets Reading and Writing (API v4)*

Description

Read and write data from Google Sheets knowing the file's title. You may use a single value from a cell or a data.frame from a cell range.

Usage

```
readGS(
  title,
  sheet = "Hoja 1",
  range = NULL,
  drop_nas = TRUE,
  json = NULL,
  email = NULL,
  api_key = NULL,
  server = FALSE,
  ...
)

writeGS(
  data,
  title,
  sheet = "Hoja 1",
  range = "A1",
  reformat = FALSE,
  append = FALSE,
  json = NULL,
  email = NULL,
  api_key = NULL,
  server = FALSE,
  ...
)
```

Arguments

title	Character. Title of Google Drive file. Uses regular expressions so you may fetch with patterns instead of names.
sheet	Character. Working sheet to import
range	Character. A cell range to read from
drop_nas	Boolean. Remove columns and rows that contain only NAs?
json	Character. JSON filename with service auth
email, api_key	Character. If you have multiple pre-authorized accounts in your machine, you may non-interactively select which one you wish to use by email and/or api_key.

server	Boolean. Force interacting auth process?
...	Additional parameters passed to read_sheet().
data	Object (value, vector, data.frame, list).
reformat	Boolean. Reformat the affected cells?
append	Boolean.

Value

For reading, data.frame with the results of your Google Sheets file based on its title, specificially the sheet and range requested. For writing, no return value.

See Also

Other Scrapper: filesGD(), get_mp3(), gtrends_related(), holidays(), ip_data(), splot_etf(), stocks_quote()

Other Google: filesGD(), gtrends_related(), queryGA()

---

reduce_pca *Reduce Dimensionality with PCA*

Description

Principal component analysis or (PCA) is a method we can use to reduce high-dimensional data to a low-dimensional space. In other words, we cannot accurately visualize high-dimensional datasets because we cannot visualize anything above 3 features. The main purpose behind PCA is to transform datasets with more than 3 features (high-dimensional) into typically a 2/3 column dataset. Despite the reduction into a lower-dimensional space we still can retain most of the variance or information from our original dataset.

Usage

```
reduce_pca(
  df,
  n = NULL,
  ignore = NULL,
  comb = c(1, 2),
  quiet = FALSE,
  plot = TRUE,
  ...
)
```

Arguments

df	Dataframe
n	Integer. Number of dimensions to reduce to.
ignore	Character vector. Names of columns to ignore.
comb	Vector. Which columns do you wish to plot? Select which two variables by name or column position.
quiet	Boolean. Keep quiet? If not, print messages.
plot	Boolean. Create plots?
...	Additional parameters passed to `stats::prcomp`

Value

List with reduced dataframe and possible plots.

See Also

Other Dimensionality: `reduce_tsne()`

Other Clusters: `clusterKmeans()`, `clusterOptimalK()`, `clusterVisualK()`, `reduce_tsne()`

Examples

```
Sys.unsetenv("LARES_FONT") # Temporal
data("iris")
df <- subset(iris, select = c(-Species))
df$id <- 1:nrow(df)
reduce_pca(df, n = 3, ignore = "id")
```

---

reduce_tsne *Reduce Dimensionality with t-SNE*

Description

t-SNE takes high-dimensional data and reduces it to a low-dimensional graph (1-3 dimensions). Unlike PCA, t-SNE can reduce dimensions with non-linear relationships. PCA attempts to draw the best fitting line through the distribution. T-SNE calculates a similarity measure based on the distance between points instead of trying to maximize variance.

Usage

```
reduce_tsne(df, n = 2, ignore = NULL, quiet = FALSE, plot = TRUE, ...)
```

Arguments

df	Dataframe
n	Integer. Number of dimensions to reduce to.
ignore	Character vector. Names of columns to ignore.
quiet	Boolean. Keep quiet? If not, print messages.
plot	Boolean. Create plots?
...	Additional parameters passed to Rtsne::Rtsne

Value

List with reduced dataframe and possible plots.

See Also

Other Dimensionality: reduce_pca()

Other Clusters: clusterKmeans(), clusterOptimalK(), clusterVisualK(), reduce_pca()

Examples

```
## Not run:
data("iris")
df <- subset(iris, select = c(-Species))
df$id <- 1:nrow(df)
reduce_tsne(df, ignore = "id", max_iter = 800, perplexity = 20)

## End(Not run)
```

---

removenacols *Remove/Drop Columns in which ALL or SOME values are NAs*

Description

This function lets the user remove all columns that have some or all values as NAs

This function lets the user remove all rows that have some or all values as NAs

Usage

```
removenacols(df, all = TRUE, ignore = NULL)

removenarows(df, all = TRUE)

numericalonly(df, dropnacols = TRUE, logs = FALSE, natransform = NA)
```

Arguments

df	Data.frame
all	Boolean. Remove rows which contains ONLY NA values. If set to FALSE, rows which contains at least one NA will be removed
ignore	Character vector. Column names to ignore validation.
dropnacols	Boolean. Drop columns with only NA values?
logs	Boolean. Calculate log(x)+1 for numerical columns?
natransform	String. "mean" or 0 to impute NA values. If set to NA no calculation will run.

Value

data.frame with removed columns.

data.frame with removed rows.

data.frame with all numerical columns selected.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), replaceall(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
data(dft) # Titanic dataset
str(dft)
numericalonly(dft) %>% head()
numericalonly(dft, natransform = "mean") %>% head()
```

---

remove_stopwords *Remove stop-words and patterns from character vector*

Description

Remove all stop-words and specific patterns from a character vector

Usage

```
remove_stopwords(text, stop_words, exclude = NULL, sep = " ")
```

Arguments

text	Character vector
stop_words	Character vector. Words to exclude from text. Example: if you want to exclude "a", whenever that word appears it will be excluded, but when the letter "a" appears in a word, it will remain.
exclude	Character. Pattern to exclude using regex.
sep	Character. String that separate the terms.

Value

Character vector with removed texts.

See Also

Other Text Mining: cleanText(), ngrams(), replaceall(), sentimentBreakdown(), textCloud(), textFeats(), textTokenizer(), topics_rake()

Examples

```
x <- c("A brown fox jumps over a dog.", "Another brown dog.")
remove_stopwords(x, stop_words = c("dog", "brown", "a"), exclude = "\\.")
```

---

replaceall *Replace Values With*

Description

This function lets the user replace all specific values in a vector or data.frame into another value. If replacing more than one value, order matters so they will be replaced in the same order that you pass them to the function. Factors will be refactored.

Usage

```
replaceall(df, original, change, which = "all", fixclass = TRUE, quiet = TRUE)
```

Arguments

df	Data.frame or Vector
original	String or Vector. Original text you wish to replace
change	String or Vector. Values you wish to replace the originals with
which	Character vector. Name of columns to use. Leave "all" for everything
fixclass	Boolean. Try to detect logical classes after transformations (or leave as default classes as character)?
quiet	Boolean. Keep quiet? (or print replacements)

Value

data.frame with replaced values based on inputs.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replacefactor(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Other Text Mining: cleanText(), ngrams(), remove_stopwords(), sentimentBreakdown(), textCloud(), textFeats(), textTokenizer(), topics_rake()

Examples

```
df <- data.frame(
  one = c(1:4, NA),
  two = LETTERS[1:5],
  three = rep("A", 5),
  four = c(NA, "Aaa", 123, "B", "C")
)
print(df)

replaceall(df, "A", NA)

replaceall(df, "A", "a")

replaceall(df, 1, "*")

replaceall(df, NA, "NotNA")

replaceall(df, NA, 0)

replaceall(df, c("A", "B"), c("'A'", "'B'"))

replaceall(df, "a", "*", which = "four")
```

---

replacefactor *Replace Factor Values*

Description

This function lets the user replace levels on a factor vector.

Usage

```
replacefactor(x, original, change)
```

Arguments

x	Factor (or Character) Vector
original	String or Vector. Original text you wish to replace
change	String or Vector. Values you wish to replace the originals with

Value

Factor vector with transformed levels.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), textFeats(), textTokenizer(), vector2text(), year_month(), zerovar()

Examples

```
library(dplyr)
data(dft)
# Replace a single value
dft <- mutate(dft, Pclass = replacefactor(Pclass, original = "1", change = "First"))
levels(dft$Pclass)
# Replace multiple values
dft <- mutate(dft, Pclass = replacefactor(Pclass, c("2", "3"), c("Second", "Third")))
levels(dft$Pclass)
```

ROC *AUC and ROC Curves Data*

Description

This function calculates ROC Curves and AUC values with 95% confidence range. It also works for multi-categorical models.

Usage

```
ROC(tag, score, multis = NA)
```

Arguments

tag	Vector. Real known label
score	Vector. Predicted value or model's result
multis	Data.frame. Containing columns with each category score (only used when more than 2 categories coexist)

Value

List with ROC's results, area under the curve (AUC) and their CI.

Plot Results

To plot results, use the `mplot_roc()` function.

See Also

Other Machine Learning: `conf_mat()`, `export_results()`, `gain_lift()`, `h2o_automl()`, `h2o_predict_API()`, `h2o_predict_MOJO()`, `h2o_predict_binary()`, `h2o_predict_model()`, `h2o_selectmodel()`, `impute()`, `iter_seeds()`, `lasso_vars()`, `model_metrics()`, `model_preprocess()`, `msplit()`

Other Model metrics: `conf_mat()`, `errors()`, `gain_lift()`, `loglossBinary()`, `model_metrics()`

Examples

```
data(dfr) # Results for AutoML Predictions
lapply(dfr[c(1, 2)], head)

# ROC Data for Binomial Model
roc1 <- ROC(dfr$class2$tag, dfr$class2$scores)
lapply(roc1, head)

# ROC Data for Multi-Categorical Model
roc2 <- ROC(dfr$class3$tag, dfr$class3$score,
  multis = subset(dfr$class3, select = -c(tag, score))
)
lapply(roc2, head)
```

---

rtistry_sphere *Generative Art: Sphere XmodY*

Description

Generative Art: Sphere XmodY

Usage

```
rtistry_sphere(eye = c(100, 0, 0), pal = "auto", var = 3)
```

Arguments

`eye, pal, var`	Parameters to change aesthetics and calculations

Value

ggplot object

scale_x_comma *Axis scales format*

Description

The _comma ones set comma format for axis text, the _percent ones set percent format for axis text, _dollar for collar currency, and _abbr for abbreviated format. Lastly, use _formatNum to further customize your numerical scales with formatNum.

Usage

```
scale_x_comma(...)

scale_y_comma(...)

scale_x_percent(...)

scale_y_percent(...)

scale_x_dollar(...)

scale_y_dollar(...)

scale_x_abbr(...)

scale_y_abbr(...)

scale_x_formatNum(
  ...,
  decimals = 2,
  signif = NULL,
  type = Sys.getenv("LARES_NUMFORMAT"),
  pre = "",
  pos = "",
  sign = FALSE,
  abbr = FALSE
)

scale_y_formatNum(
  ...,
  decimals = 2,
  signif = NULL,
  type = Sys.getenv("LARES_NUMFORMAT"),
  pre = "",
  pos = "",
  sign = FALSE,
  abbr = FALSE
```

```
)
```

Arguments

...	Arguments passed to ggplot2::continuous_scale or formatNum depending on the function.
decimals	Integer. Amount of decimals to display. If set to NULL, then getOption("digits") will be used.
signif	Integer. Rounds the values in its first argument to the specified number of significant digits.
type	Integer. 1 for International standards. 2 for American Standards. Use Sys.setenv("LARES_NUMFORMAT" = 2) to set this parameter globally.
pre, pos	Character. Add string before or after number.
sign	Boolean. Add + sign to positive values.
abbr	Boolean. Abbreviate using num_abbr()? You can use the 'decimals' parameter to set abbr's n(-1) parameter.

Value

Reformatted scales on ggplot2 object

Examples

```
library(ggplot2)
df <- ggplot2::txhousing %>% removenarows(all = FALSE)

ggplot(df, aes(x = sales, y = volume)) +
  geom_point() +
  scale_x_dollar() +
  scale_y_abbr()

# Use any argument from scale_x/y_continuous
ggplot(df, aes(x = listings, y = log(inventory))) +
  geom_point() +
  scale_x_comma() +
  scale_y_percent(limits = c(0, 3))

# Use any argument from scale_x/y_continuous AND formatNum
ggplot(df, aes(x = median, y = inventory)) +
  geom_point() +
  scale_x_formatNum(n.breaks = 3, pre = "@", abbr = TRUE) +
  scale_y_formatNum(position = "right", decimals = 0, pos = " X")
```

`scrabble_dictionary` *Scrabble: Dictionaries*

Description

Download words from 4 different languages: English, Spanish, German, and French. Words will be save into the `temp` directory. This is an auxiliary function. You may want to use `scrabble_words` directly if you are searching for the highest score words!

Get score for any word or list of words. You may set manually depending on the rules and languages you are playing with. Check the examples for Spanish and English values when I played Words With Friends.

Dataframe for every letter and points given a language.

Find highest score words given a set of letters, rules, and language to win at Scrabble! You just have to find the best place to post your tiles.

Usage

```
scrabble_dictionary(lang_dic, quiet = FALSE)

scrabble_score(words, scores.df)

scrabble_points(lang)

scrabble_words(
  tiles = "",
  free = 0,
  force_start = "",
  force_end = "",
  force_str = "",
  force_exclude = "",
  exclude_here = "",
  force_n = 0,
  force_max = 0,
  language = Sys.getenv("LARES_LANG"),
  scores = language,
  words = NULL,
  quiet = FALSE,
  print = TRUE
)
```

Arguments

`lang_dic`	Character. Any of "en","es","de","fr". Set to NULL if you wish to skip this step (and use `words` parameter in `scrabble_words` instead).
`quiet`	Boolean. Do not print words as they are being searched.

`words` Character vector. Use if you wish to manually add words.

`scores.df` Dataframe. Must contain two columns: "tiles" with every letter of the alphabet and "scores" for each letter's score.

`lang` Character. Any of "en","es". Set to NULL if you wish to skip this step (and use `words` parameter in `scrabble_words()` instead).

`tiles` Character. The letters you wish to consider.

`free` Integer. How many free blank tiles you have?

`force_start, force_end`
Character. Force words to start or end with a pattern of letters and position. Examples: "S" or "SO" or "__S_O"... If the string contains tiles that were not specified in `tiles`, they will automatically be included.

`force_str` Character vector. Force words to contain strings. If the string contains tiles that were not specified in `tiles`, they will automatically be included.

`force_exclude, exclude_here`
Character vector. Exclude words containing these tiles (and positions). Not very relevant on Scrabble but for Wordle.

`force_n, force_max`
Integer. Force words to be n or max n characters long. Leave 0 to ignore parameter.

`scores, language`
Character. Any of "en","es","de","fr". If scores is not any of those languages, must be a data.frame that contains two columns: "tiles" with every letter of the alphabet and "scores" for each letter's score. If you wish to overwrite or complement this dictionaries other words you can set to `"none"` and/or use the `words` parameter. You might also want to set this parameter globally with `Sys.setenv("LARES_LANG" = "en")` and forget about it!

`print` Boolean. Print how many words are left by step.

Value

data.frame with words and language columns.

data.frame with word, scores, and length values for each `word`.

data.frame with tiles and scores for each alphabet letter.

data.frame with matching words found, sorted by higher points.

Examples

```
# For Spanish words
dictionary <- scrabble_dictionary("es")


# For Spanish words (default)
es_scores <- scrabble_points("es")
# Custom scores for each letter
cu_scores <- data.frame(
```

```
  tiles = tolower(LETTERS),
  scores = c(1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 5, 2, 4, 2, 1, 4, 10, 1, 1, 1, 2, 5, 4, 8, 3, 10)
)

# Score values for each set of rules
words <- c("Bernardo", "Whiskey", "R is great")
scrabble_score(words, es_scores)
scrabble_score(words, cu_scores)

scrabble_points("es")
scrabble_points("en")
# Not yet available
scrabble_points("fr")

# Automatic use of languages and scores
Sys.setenv("LARES_LANG" = "es")
scrabble_words(
  tiles = "hola",
  free = 2,
  force_start = "h",
  force_n = 4,
  force_str = "_o_a",
  exclude_here = "__z|j"
)

wordle <- c("board", "tempo", "shoes", "hoard")
scrabble_words(
  language = NULL,
  words = wordle,
  force_n = 5,
  force_str = "O_R"
)

# Words considered for a language (you can custom it too!)
es_words <- scrabble_dictionary("es")
```

`sentimentBreakdown` *Sentiment Breakdown on Text*

Description

This function searches for relevant words in a given text and adds sentiments labels (joy, anticipation, surprise, positive, trust, anger, sadness, fear, negative, disgust) for each of them, using NRC. Then, makes a summary for all words and plot results.

Usage

```
sentimentBreakdown(
  text,
```

```
  lang = "spanish",
  exclude = c("maduro", "que"),
  append_file = NA,
  append_words = NA,
  plot = TRUE,
  subtitle = NA
)
```

Arguments

text	Character vector
lang	Character. Language in text (used for stop words)
exclude	Character vector. Which word do you wish to exclude?
append_file	Character. Add a dictionary to append. This file must contain at least two columns, first with words and second with the sentiment (consider sentiments on description).
append_words	Dataframe. Same as append_file but appending data frame with word and sentiment directly
plot	Boolean. Plot results summary?
subtitle	Character. Add subtitle to the plot

Value

List. Contains data.frame with words and sentiments, summary and plot.

See Also

Other Text Mining: cleanText(), ngrams(), remove_stopwords(), replaceall(), textCloud(), textFeats(), textTokenizer(), topics_rake()

---

shap_var *SHAP-based dependence plots for categorical/numerical features (PDP)*

Description

Having a h2o_shap object, plot a dependence plot for any categorical or numerical feature.

Usage

```
shap_var(x, var, keep_outliers = FALSE)
```

Arguments

x	h2o_shap object
var	Variable name
keep_outliers	Boolean. Outliers detected with z-score and 3sd may be suppress or kept in your plot. Keep them?

Value

ggplot2 objct with shap values plotted

See Also

Other SHAP: h2o_shap()

Examples

```
## Not run:
# Train a h2o_automl model
model <- h2o_automl(dft, Survived,
  max_models = 1, target = TRUE,
  ignore = c("Ticket", "Cabin", "PassengerId"),
  quiet = TRUE
)

# Calculate SHAP values
SHAP_values <- h2o_shap(model)
# Equivalent to:
# SHAP_values <- h2o_shap(
#  model = model$model,
#  test = model$datasets$test,
#  scores = model$scores_test$scores)

# Check SHAP results
head(SHAP_values)

# You must have "ggbeeswarm" library to use this auxiliary function:
# Plot SHAP values (feature importance)
plot(SHAP_values)

# Plot some of the variables (categorical)
shap_var(SHAP_values, Pclass)

# Plot some of the variables (numerical)
shap_var(SHAP_values, Fare)

## End(Not run)
```

slackSend *Send Slack Message (Webhook)*

Description

This function send a Slack message using its Webhooks.

Usage

```
slackSend(text, title = "", pretext = "", hook = NA, creds = NA)
```

Arguments

text, title, pretext	Character. Content on you Slack message.
hook	Character. Web hook URL. Ths value will be overwritten by creds if correctly used.
creds	Character. Credential's dir (see get_creds()). Set hook URL into the "slack" list in your YML file. Will use first value.

Details

For more help, you can follow the Sending messages using Incoming Webhooks original documentarion.

Value

Invisible POST response

See Also

Other API: bring_api(), fb_accounts(), fb_ads(), fb_creatives(), fb_insights(), fb_process(), fb_report_check(), fb_rf(), fb_token(), gpt_ask(), li_auth(), li_profile(), queryGA()

Other Credentials: db_download(), db_upload(), get_credentials(), get_tweets(), mail_send(), queryDB(), queryGA(), stocks_file(), stocks_report()

Examples

```
## Not run:
slackSend(text = "This is a message", title = "TEST", pretext = Sys.info()["user"])

## End(Not run)
```

splot_change *Portfolio Plots: Daily Change*

Description

This function plots each stock's change through history, since inception, with weighted attributions or absolute values.

Usage

```
splot_change(
  p,
  s,
  rel = TRUE,
  group = FALSE,
  n_days = 365,
  keep_old = FALSE,
  save = FALSE
)
```

Arguments

p	Dataframe. Result from daily_portfolio()
s	Dataframe. Result from daily_stocks()
rel	Boolean. Relative delta values (weighted with portfolio)? If not, absolute monetary delta values.
group	Boolean. Group stocks by stocks type?
n_days	Integer. How many days back you want to see? sold entirely?
keep_old	Boolean. Include sold tickers even though not currently in portfolio?
save	Boolean. Save plot into a local file?

Value

ggplot object

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

Other Investment Plots: splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types()

splot_divs *Portfolio Plots: Dividends per Year and Quarter*

Description

This function plots a portfolio's historical dividends incomes grouped by quarter an year.

Usage

```
splot_divs(p, type = 1)
```

Arguments

p	Dataframe. Result from daily_portfolio()
type	Integer. Typo of plot. 1 for incomes.

Value

ggplot object

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

Other Investment Plots: splot_change(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types()

splot_etf *Portfolio's Sector Distribution (ETFs)*

Description

This function lets the user plot his portfolio's distribution, specifically ETF's sectors

Usage

```
splot_etf(s, keep_all = FALSE, cache = TRUE, save = FALSE)
```

Arguments

s	Dataframe. Result from daily_stocks()
keep_all	Boolean. Keep "Not Known / Not ETF"?
cache	Boolean. Use daily cache if available?
save	Boolean. Save plot into a local file?

Value

ggplot2 object

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

Other Investment Plots: splot_change(), splot_divs(), splot_growth(), splot_roi(), splot_summary(), splot_types()

Other Scrapper: filesGD(), get_mp3(), gtrends_related(), holidays(), ip_data(), readGS(), stocks_quote()

---

splot_growth *Portfolio Plots: Growth (Cash + Invested)*

Description

This function plots your portfolio's growth, in cash and investment, since inception.

Usage

```
splot_growth(p, save = FALSE)
```

Arguments

p	Dataframe. Result from daily_portfolio()
save	Boolean. Save plot into a local file?

Value

ggplot object

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

Other Investment Plots: splot_change(), splot_divs(), splot_etf(), splot_roi(), splot_summary(), splot_types()

splot_roi *Portfolio Plots: Daily ROI*

Description

This function plots a portfolio's historical ROI since inception or since last n days, with 2 moving average lines.

Usage

```
splot_roi(p, n_days = 365, historical = TRUE, ma = c(12, 50), save = FALSE)
```

Arguments

p	Dataframe. Result from daily_portfolio()
n_days	Integer. How many days back you want to see?
historical	Boolean. Historical ROI metric? If not, ROI will be calculated locally for n_days parameter
ma	Numeric Vector. Select 2 values for moving averages. Set to NA to turn this metric off
save	Boolean. Save plot into a local file?

Value

ggplot object

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

Other Investment Plots: splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_summary(), splot_types()

---

splot_summary *Portfolio Plots: Total Summary*

Description

This function plots a summary for the whole portfolio, showing how much have you invested, how much has each ticker changed, etc.

Usage

```
splot_summary(p, s, save = FALSE)
```

Arguments

p	Dataframe. Result from daily_portfolio()
s	Dataframe. Result from daily_stocks()
save	Boolean. Save plot into a local file?

Value

ggplot object

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_types(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

Other Investment Plots: splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_types()

---

splot_types *Portfolio Plots: Types of Stocks*

Description

This function lets the user plot types or categories of tickers.

Usage

```
splot_types(s, save = FALSE)
```

Arguments

s	Dataframe. Result from daily_stocks()
save	Boolean. Save plot into a local file?

Value

ggplot object

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), stocks_file(), stocks_obj(), stocks_quote(), stocks_report()

Other Investment Plots: splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary()

spread_list *Spread list column into new columns*

Description

Spread an existing list column into new columns on a data.frame. Note that every element on every observation must have a name for the function to do its work. Original column will be automatically suppressed but you can set the `replace` argument to avoid it.

Usage

```
spread_list(df, col, str = NULL, replace = TRUE)
```

Arguments

`df`	Dataframe
`col`	Variable name.
`str`	Character. Start column names with. If set to `NULL`, original name of column will be used.
`replace`	Boolean. Replace original values (delete column)

Value

data.frame. Result of un-nesting named or un-named list columns.

Examples

```
df <- dplyr::starwars
# Un-named list columns
spread_list(df, films, replace = FALSE) %>%
  dplyr::select(name, dplyr::starts_with("films")) %>%
  head(8)
# Named (and un-nammed) list columns
df <- dplyr::tibble(id = 1:3, platform = list(
  list("fb" = 1, "ig" = 2),
  list("fb" = 3),
  list()
))
spread_list(df, platform, str = "ptf_")
```

statusbar *Progressive Status Bar (Loading)*

Description

This function lets the user view a progressbar for a 'for' loop.

Usage

```
statusbar(
  run = 1,
  max.run = 100,
  label = run,
  msg = "",
  type = Sys.getenv("LARES_STATUSBAR"),
  start_time = NA,
  multiples = 1,
  alarm = FALSE
)
```

Arguments

run	Iterator. for loop or an integer with the current loop number. Start with 1 preferibly
max.run	Number. Maximum number of loops
label	String. With additionaly information to be printed at the end of the line. The default is run.
msg	Character. Finish message.
type	Character. Loading type style: equal, domino, sword, filled.
start_time	POSIXct. Start time to consider. If NA, then when first iteration starts will be set as start time. Useful for when first iteration is showed as done but started a few seconds/minutes ago.
multiples	Integer. Only print when multiples of N (to avoid) wasting resources on fast and lots of iterations.
alarm	Boolean. Ping (sound) when done. Requires beepr.

Value

No return value, called for side effects.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(),

importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), tic(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
for (i in 1:9) {
  statusbar(i, 9, multiples = 2)
  Sys.sleep(0.3)
}
```

---

stocks_file *Get Personal Portfolio's Data*

Description

This function lets the user download his personal Excel with his Portfolio's data, locally or from Dropbox.

Usage

```
stocks_file(
  file = NA,
  creds = NA,
  auto = TRUE,
  sheets = c("Portafolio", "Fondos", "Transacciones"),
  keep_old = TRUE,
  cache = TRUE,
  quiet = FALSE
)
```

Arguments

file	Character. Import an Excel file, local or from URL.
creds	Character. Dropbox's credentials (see get_creds())
auto	Boolean. Automatically use my local personal file? You might want to set in into your .Renviron LARES_PORTFOLIO=~/dir/to/your/file.xlsx so you can leave all other parameters as NA and use it every time.
sheets	Character Vector. Names of each sheet containing Portfolio summary, Cash, and Transactions information
keep_old	Boolean. Include sold tickers even though not currently in portfolio?
cache	Boolean. Use daily cache if available?
quiet	Boolean. Keep quiet? If not, informative messages will be printed.

Value

List with portfolio, transactions, and cash data.frames.

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_obj(), stocks_quote(), stocks_report()

Other Credentials: db_download(), db_upload(), get_credentials(), get_tweets(), mail_send(), queryDB(), queryGA(), slackSend(), stocks_report()

Examples

```
## Not run:
# Load lares dummy portfolio XLSX
file <- system.file("inst/docs", "dummyPortfolio.xlsx", package = "lares")
df <- stocks_file(
  file = file,
  sheets = c("Portafolio", "Fondos", "Transacciones"),
  keep_old = FALSE
)

## End(Not run)
```

---

stocks_obj *Portfolio's Calculations and Plots*

Description

This function lets the user create his portfolio's calculations and plots for further study.

Usage

```
stocks_obj(
  data = stocks_file(),
  cash_fix = 0,
  tax = 30,
  sectors = FALSE,
  parg = FALSE,
  window = c("1M", "YTD", "1Y", "MAX"),
  cache = TRUE,
  quiet = FALSE
)
```

Arguments

data	List. Containing the following dataframes: portfolio, transactions, cash. They have to follow the original xlsx format
cash_fix	Numeric. If, for some reason, you need to fix your cash amount for all reports, set the amount here
tax	Numeric. How much [0-99] of your dividends are gone with taxes?

sectors	Boolean. Return sectors segmentation for ETFs?
parg	Boolean. Personal argument. Used to personalize stuff, in this case, taxes changed from A to B in given date (hard-coded)
window	Character. Choose any of: "1W", "1M", "6M", "1Y", "YTD", "5Y", "MAX"
cache	Boolean. Use daily cache if available?
quiet	Boolean. Keep quiet? If not, informative messages will be printed.

Value

List. Aggregated results and plots.

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_quote(), stocks_report()

---

stocks_quote *Download Stocks Historical and Current Values*

Description

This function lets the user download stocks live data.

This function lets the user download stocks historical data.

Usage

```
stocks_quote(symbols)

stocks_hist(
  symbols = c("VTI", "TSLA"),
  from = Sys.Date() - 365,
  to = Sys.Date(),
  today = TRUE,
  tax = 15,
  parg = FALSE,
  cache = TRUE,
  quiet = FALSE,
  ...
)

## S3 method for class 'stocks_hist'
plot(x, type = 1, ...)
```

Arguments

symbols	Character Vector. List of symbols to download historical data
from, to	Date. Dates for range. If not set, 1 year will be downloaded. Do use more than 4 days or will be over-written.
today	Boolean. Do you wish to add today's live quote? This will happen only if to value is the same as today's date
tax	Numeric. How much [0-99] of your dividends are gone with taxes?
parg	Boolean. Personal argument. Used to personalize stuff, in this case, taxes changed from A to B in given date (hard-coded)
cache	Boolean. Use daily cache if available?
quiet	Boolean. Keep quiet? If not, informative messages will be printed.
...	Additional parameters
x	stocks_hist object
type	Integer. Select type of plot.

Value

data.frame with Symbol, Type of stock, Quote time, current value, Daily Change, Market, and Symbol Name.

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_report()

Other Scrapper: filesGD(), get_mp3(), gtrends_related(), holidays(), ip_data(), readGS(), splot_etf()

Examples

```
# Multiple quotes at the same time
stocks_quote(c("VTI", "VOO", "TSLA"))

## Not run:
# CRAN
df <- stocks_hist(symbols = c("VTI", "FB", "FIW"), from = Sys.Date() - 180)
print(head(df))
plot(df)

## End(Not run)
```

stocks_report *Portfolio's Full Report and Email*

Description

This function lets the user create his portfolio's full report with plots and send it to an email with the HTML report attached

Usage

```
stocks_report(
  data = NA,
  keep_old = TRUE,
  dir = NA,
  mail = FALSE,
  attachment = TRUE,
  to = "laresbernardo@gmail.com",
  sectors = FALSE,
  keep = FALSE,
  creds = NA,
  cache = TRUE
)
```

Arguments

data	Character. `stocks_obj()` output. If NA, automatic parameters and `stocks_file()` defaults will be used.
keep_old	Boolean. Include sold tickers even though not currently in portfolio?
dir	Character. Directory for HTML report output. If set to NA, current working directory will be used. If mail sent, file will be erased
mail	Boolean. Do you want to send an email with the report attached? If not, an HTML file will be created in dir
attachment	Boolean. Create and add report as attachment if `mail=TRUE`? If not, no report will be rendered and only tabulated summaries will be included on email's body.
to	Character. Email to send the report to
sectors	Boolean. Return sectors segmentation for ETFs?
keep	Boolean. Keep HTML file when sent by email?
creds	Character. Credential's user (see `get_creds()`) for sending mail and Dropbox interaction.
cache	Boolean. Use daily cache if available?

Value

Invisible list. Aggregated results and plots.

See Also

Other Investment: daily_portfolio(), daily_stocks(), etf_sector(), splot_change(), splot_divs(), splot_etf(), splot_growth(), splot_roi(), splot_summary(), splot_types(), stocks_file(), stocks_obj(), stocks_quote()

Other Credentials: db_download(), db_upload(), get_credentials(), get_tweets(), mail_send(), queryDB(), queryGA(), slackSend(), stocks_file()

Examples

```
## Not run:
list <- stocks_obj()
stocks_report(list, dir = "~/Desktop")

## End(Not run)
```

---

sudoku_solver *Solve Sudoku Puzzles*

Description

Solve a Sudoku puzzle, where empty values are represented by 0s into a matrix object.

Usage

```
sudoku_solver(board, needed_cells = NULL, index = 1, quiet = FALSE)
```

Arguments

board	Matrix. 9x9 matrix or vector length 81, with only digits from 0 to 9.
needed_cells, index	Auxiliary parameters to auto-iterate using this same fx.
quiet	Boolean. Keep quiet? If not, plot results.

Value

Logical output answering of the input board can be solved. The actual solved solution will be created as an object named solved in your .GlobalEnv.

Examples

```
# board <- c(0,0,0,0,0,6,000,
#            0,9,5,7,0,0,3,0,0,
#            4,0,0,0,9,2,0,0,5,
#            7,6,4,0,0,0,0,0,3,
#            0,0,0,0,0,0,0,0,0,
#            2,0,0,0,0,0,9,7,1,
#            5,0,0,2,1,0,0,0,9,
```

```
#            0,0,7,0,0,5,4,8,0,
#            0,0,0,8,0,0,0,0,0)
# sudoku_solver(board)

# Trivial input (everything)
trivial <- matrix(rep(0, 81), byrow = TRUE, ncol = 9)
trivial
sudoku_solver(trivial)

# Wrong / Impossible to solve input
imp <- matrix(c(rep(1, 72), rep(0, 9)), byrow = TRUE, ncol = 9)
imp
sudoku_solver(imp)
```

`target_set` *Set Target Value in Target Variable*

Description

This function detects or forces the target value when predicting a categorical binary model. This is an auxiliary function.

Usage

```
target_set(tag, score, target = "auto", quiet = FALSE)
```

Arguments

`tag`	Vector. Real known label
`score`	Vector. Predicted value or model's result
`target`	Value. Which is your target positive value? If set to 'auto', the target with largest mean(score) will be selected. Change the value to overwrite. Only used when binary categorical model.
`quiet`	Boolean. Do not show message for auto target?

Value

List. Contains original data.frame `df` and `which` with the target variable.

textCloud *Wordcloud Plot*

Description

Study the distribution of a target variable vs another variable. This function is quite similar to the funModeling's corrplot function.

Usage

```
textCloud(
  text,
  lang = "english",
  exclude = NULL,
  seed = 0,
  keep_spaces = FALSE,
  min = 2,
  pal = NA,
  print = TRUE
)
```

Arguments

text	Character vector
lang	Character. Language in text (used for stop words)
exclude	Character vector. Which word do you wish to exclude?
seed	Numeric. Seed for re-producible plots
keep_spaces	Boolean. If you wish to keep spaces in each line to keep unique compount words, separated with spaces, set to TRUE. For example, 'LA ALAMEDA' will be set as 'LA_ALAMEDA' and treated as a single word.
min	Integer. Words with less frequency will not be plotted
pal	Character vector. Which colours do you wish to use
print	Boolean. Plot results as textcloud?

Value

wordcloud plot object

See Also

Other Text Mining: cleanText(), ngrams(), remove_stopwords(), replaceall(), sentimentBreakdown(), textFeats(), textTokenizer(), topics_rake()

`textFeats` *Create features out of text*

Description

This function creates a data.frame with features based on a text vector

Usage

```
textFeats(text, auto = TRUE, contains = NA, prc = FALSE)
```

Arguments

`text`	Character vector
`auto`	Boolean. Auto create some useful parameters?
`contains`	Character vector. Which columns do you wish to add with a contains (counter) string validator?
`prc`	Boolean. Also add percentage of each column compared with length?

Value

data.frame with additional features based on `text`.

See Also

Other Data Wrangling: `balance_data()`, `categ_reducer()`, `cleanText()`, `date_cuts()`, `date_feats()`, `file_name()`, `formatHTML()`, `holidays()`, `impute()`, `left()`, `normalize()`, `num_abbr()`, `ohe_commas()`, `ohse()`, `quants()`, `removenacols()`, `replaceall()`, `replacefactor()`, `textTokenizer()`, `vector2text()`, `year_month()`, `zerovar()`

Other Text Mining: `cleanText()`, `ngrams()`, `remove_stopwords()`, `replaceall()`, `sentimentBreakdown()`, `textCloud()`, `textTokenizer()`, `topics_rake()`

Examples

```
textFeats("Bernardo Lares")
textFeats("Bernardo Lares 123!", prc = TRUE)
textFeats("I'm 100% Lares...", contains = c("Lares", "lares"))
textFeats(c("GREAT library!!", "Have you tried this 2?", "Happy faces :D :-)"))
```

`textTokenizer` *Tokenize Vectors into Words*

Description

This function transforms texts into words, calculate frequencies, supress stop words in a given language.

Usage

```
textTokenizer(
  text,
  exclude = NULL,
  lang = NULL,
  min_word_freq = 5,
  min_word_len = 2,
  keep_spaces = FALSE,
  lowercase = TRUE,
  remove_numbers = TRUE,
  remove_punct = TRUE,
  remove_lettt = TRUE,
  laughs = TRUE,
  utf = TRUE,
  df = FALSE,
  h2o = FALSE,
  quiet = FALSE
)
```

Arguments

`text`	Character vector. Sentences or texts you wish to tokenize.
`exclude`	Character vector. Which words do you wish to exclude?
`lang`	Character. Language in text (used for stop words). Example: "spanish" or "english". Set to `NA` to ignore.
`min_word_freq`	Integer. This will discard words that appear less than <int> times. Defaults to 2. Set to `NA` to ignore.
`min_word_len`	Integer. This will discard words that have less than <int> characters. Defaults to 5. Set to `NA` to ignore.
`keep_spaces`	Boolean. If you wish to keep spaces in each line to keep unique compound words, separated with spaces, set to TRUE. For example, 'one two' will be set as 'one_two' and treated as a single word.
`lowercase, remove_numbers, remove_punct`	Boolean.
`remove_lettt`	Boolean. Repeated letters (more than 3 consecutive).
`laughs`	Boolean. Try to unify all laughs texts.

utf	Boolean. Transform all characters to UTF (no accents and crazy symbols)
df	Boolean. Return a dataframe with a one-hot-encoding kind of results? Each word is a column and returns if word is contained.
h2o	Boolean. Return H2OFrame?
quiet	Boolean. Keep quiet? If not, print messages

Value

data.frame. Tokenized words with counters.

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), vector2text(), year_month(), zerovar()

Other Text Mining: cleanText(), ngrams(), remove_stopwords(), replaceall(), sentimentBreakdown(), textCloud(), textFeats(), topics_rake()

---

theme_lares *Theme for ggplot2 (lares)*

Description

Based on hrbrthemes' theme_ipsum and customized for lares usage. With this team you can custom the colour and fill palettes, global colour parameters, major and minor grids, legend, font and font size.

Usage

```
theme_lares(
  font = Sys.getenv("LARES_FONT"),
  size = 12,
  main_colour = "darkorange3",
  hard_colour = "black",
  soft_colour = "grey30",
  plot_colour = "transparent",
  panel_colour = "transparent",
  background = "transparent",
  no_facets = FALSE,
  legend = NULL,
  grid = TRUE,
  axis = TRUE,
  clean = FALSE,
  mg = 9,
  pal = 0,
```

```
  palette = NULL,
  which = "fc",
  ...
)
```

Arguments

font, size	Character and numeric. Base font family and base size for texts. Arial Narrow is set by default when the library is loaded; you may change it with Sys.getenv("LARES_FONT" = "X") or by using this parameter manually.
main_colour, hard_colour, soft_colour, plot_colour, panel_colour	Character. Main colours for your theme.
background	Character. Main colour for your background. Overwrites plot_colour and panel_colour.
no_facets	Boolean. Suppress facet labels?
legend	Character. Legend position: "top", "right", "bottom", or "left" You can also set to FALSE or "none" to suppress legend.
grid	Character or Boolean. Use TRUE/FALSE or a combination of X, x, Y, y to enable/disable minor and major grids.
axis	Character or Boolean. Use TRUE/FALSE, x or Y to enable X and/or Y axis lines.
clean	Boolean. Suppress grids and axis? Overwrites both parameters.
mg	Numeric. External margins reference.
pal	Integer. 1 for fill and colour palette, 2 for only colour palette, 3 for only fill palette, 4 for personal labels-colour palette. 0 for nothing.
palette	Character vector. Pass a vector with HEX colour codes to use a custom palette. If you pass a named vector, the name values will be used as fill and the values will be used as colour.
which	Character. When pal = 3, select which colours should be added with the custom colours palette: fill, colour, text (fct) - first letters.
...	Additional parameters passed

Value

Themed ggplot2 object

Why Arial Narrow?

First and foremost, Arial Narrow is generally installed by default or readily available on any modern system, so it's "free"-ish; plus, it is a condensed font with solid default kerning pairs and geometric numbers.

See Also

Other Themes: gg_fill_customs(), lares_pal(), plot_palette()

Examples

```
data(dft)
library(ggplot2)
p <- ggplot(dft, aes(x = Pclass, y = sum(Fare), fill = Pclass)) +
  geom_col()
p + theme_lares()
p + theme_lares(pal = 1)
p + theme_lares(background = "#999999", mg = 25)
p + theme_lares(legend = "top", grid = "Yy")
p + theme_lares(clean = TRUE)
```

---

tic *Stopwatch to measure timings in R*

Description

Start a stopwatch.

Stop a stopwatch.

Usage

```
tic(id = 1, start = proc.time()["elapsed"], quiet = TRUE)

toc(
  id = 1,
  msg = "Elapsed time:",
  type = "units",
  signif = 3,
  quiet = FALSE,
  ...
)
```

Arguments

id	Define ID if multiple tic & toc are being used.
start	Start time. Now is default.
quiet	Boolean. Quiet messages?
msg	Character. Custom message shown
type	Character. Output format for time list element. Choose any of: units, clock, seconds.
signif	Integer. Significant digits.
...	Additional parameters.

Value

Invisible list. Contains tic (start time), toc (stop time), elapsed time and message printed.

toc returns an (invisible) list containing the time-stamps tic and toc, time in seconds and the message msg.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), try_require(), updateLares(), warnifnot(), what_size()

Examples

```
# Basic use (global stopwatch)
tic()
Sys.sleep(0.1)
toc()

# Multiple tic tocs
tic(id = "two", quiet = FALSE)
Sys.sleep(0.2)
toc(id = "two")

# Global is still working (id = 1)
toc(msg = "The function finished its work in")
```

---

topics_rake *Keyword/Topic identification using RAKE*

Description

RAKE is a basic algorithm which tries to identify keywords in text. Based on udpipe library, model models, and keywords_rake function.

Usage

```
topics_rake(text, file = "english-ewt-ud-2.4-190531.udpipe", lang = "english")
```

Arguments

text	Character vector
file	Character. Name of udpipe model previously downloaded for a specific language
lang	Character. If file does not exist, this language will be downloaded from udpipe's models.

Value

data.frame with topics for each `text` input.

See Also

Other Text Mining: `cleanText()`, `ngrams()`, `remove_stopwords()`, `replaceall()`, `sentimentBreakdown()`, `textCloud()`, `textFeats()`, `textTokenizer()`

---

tree_var *Recursive Partitioning and Regression Trees*

Description

Fit and plot a `rpart` model for exploratory purposes using `rpart` and `rpart.plot` libraries.

Usage

```
tree_var(
  df,
  y,
  type = 2,
  max = 3,
  min = 20,
  cp = 0,
  ohse = TRUE,
  plot = TRUE,
  explain = TRUE,
  title = NA,
  subtitle = NULL,
  ...
)
```

Arguments

`df`	Data frame
`y`	Variable or Character. Name of the independent variable.
`type`	Type of plot. Possible values: **0** Draw a split label at each split and a node label at each leaf. **1** Label all nodes, not just leaves. Similar to `text.rpart`'s `all=TRUE`. **2** Default. Like `1` but draw the split labels below the node labels. Similar to the plots in the CART book. **3** Draw separate split labels for the left and right directions. **4** Like `3` but label all nodes, not just leaves. Similar to `text.rpart`'s `fancy=TRUE`. See also `clip.right.labs`. **5** Show the split variable name in the interior nodes.

max	Integer. Maximal depth of the tree.
min	Integer. The minimum number of observations that must exist in a node in order for a split to be attempted.
cp	complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. For instance, with anova splitting, this means that the overall R-squared must increase by cp at each step. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. Essentially,the user informs the program that any split which does not improve the fit by cp will likely be pruned off by cross-validation, and that hence the program need not pursue it.
ohse	Boolean. Auto generate One Hot Smart Encoding?
plot	Boolean. Return a plot? If not, rpart object.
explain	Boolean. Include a brief explanation on the bottom part of the plot.
title, subtitle	Character. Title and subtitle to include in plot. Set to NULL to ignore.
...	Additional parameters passed to rpart.plot().

Details

This differs from the tree function in S mainly in its handling of surrogate variables. In most details it follows Breiman *et. al* (1984) quite closely. R package **tree** provides a re-implementation of tree.

Value

(Invisible) list type 'tree_var' with plot (function), model, predictions, performance metrics, and interpret auxiliary text.

Author(s)

Stephen Milborrow, borrowing heavily from the rpart package by Terry M. Therneau and Beth Atkinson, and the R port of that package by Brian Ripley.

References

Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) *Classification and Regression Trees.* Wadsworth.

See Also

Other Exploratory: corr_cross(), corr_var(), crosstab(), df_str(), distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), lasso_vars(), missingness(), plot_cats(), plot_df(), plot_nums()

Other Visualization: distr(), freqs_df(), freqs_list(), freqs_plot(), freqs(), noPlot(), plot_chord(), plot_survey(), plot_timeline()

Examples

```
data(dft)
# Regression Tree
tree <- tree_var(dft, Fare, subtitle = "Titanic dataset")
tree$plot() # tree plot
tree$model # rpart model object
tree$performance # metrics
# Binary Tree
tree_var(dft, Survived_TRUE, explain = FALSE, cex = 0.8)$plot()
# Multiclass tree
tree_var(dft[, c("Pclass", "Fare", "Age")], Pclass, ohse = FALSE)$plot()
```

---

trim_mp3 *Trim MP3 Audio File*

Description

This function trims MP3 files given a start and/or end numeric timestamp. Requires `ffmpeg` installed in your machine.

Usage

```
trim_mp3(
  file,
  start_time = 0,
  end_time = NA,
  overwrite = FALSE,
  ext = "mp3",
  quiet = FALSE
)
```

Arguments

`file`	Character. File name to trim.
`start_time, end_time`	Numeric. Start and end time to trim the audio output in seconds.
`overwrite`	Boolean. Overwrite original file?
`ext`	Character. File extension/type.
`quiet`	Boolean. Keep quiet? If not, print messages.

See Also

Other Audio: `get_mp3()`

try_require *Check if Specific Package is Installed*

Description

This function checks library dependencies

Usage

```
try_require(package, stop = TRUE, load = TRUE, lib.loc = NULL, ...)
```

Arguments

package	Character. Name of the library
stop	Boolean. Stop if not installed. If FALSE and library is not available, warning will be shown.
load	Boolean. Load library?
lib.loc	Character vector. Location of R library trees to search through, or NULL. The default value of NULL corresponds to all libraries currently known to .libPaths(). Non-existent library trees are silently ignored.
...	Pass additional parameters.

Value

No return value, called for side effects.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), updateLares(), warnifnot(), what_size()

Examples

```
# Check if library base is installed. If not, stop and show error
try_require("base", stop = TRUE)
# Check if library xxx is installed. If not, show warning
try_require("xxx", stop = FALSE)
```

updateLares *Update the library (dev or CRAN version)*

Description

This auxiliary function lets the user update `lares` to latest CRAN or developer version.

Usage

```
updateLares(force = FALSE, dev = TRUE, all = FALSE, local = FALSE, fb = FALSE)
```

Arguments

`force`	Boolean. Force install.
`dev`	Boolean. Developer version (Github)? If not, CRAN version.
`all`	Boolean. Install other recommended libraries? Kinda Docker install!
`local`	Boolean. Install package with local files? (or Github repo).
`fb`	Boolean. From FB instance? Personal internal use.

Value

No return value, called for side effects.

See Also

Other Tools: `autoline()`, `bind_files()`, `bring_api()`, `db_download()`, `db_upload()`, `export_plot()`, `export_results()`, `files_functions()`, `font_exists()`, `formatColoured()`, `formatHTML()`, `get_credentials()`, `glued()`, `grepm()`, `h2o_selectmodel()`, `haveInternet()`, `image_metadata()`, `importxlsx()`, `ip_data()`, `json2vector()`, `list_cats()`, `listfiles()`, `mail_send()`, `markdown2df()`, `move_files()`, `msplit()`, `myip()`, `quiet()`, `read.file()`, `statusbar()`, `tic()`, `try_require()`, `warnifnot()`, `what_size()`

vector2text *Convert a vector into a comma separated text*

Description

Convert a vector into a comma separated text

Usage

```
vector2text(vector, sep = ", ", quotes = TRUE, force_single = FALSE, and = "")

v2t(vector, sep = ", ", quotes = TRUE, force_single = FALSE, and = "")
```

Arguments

vector	Vector. Vector with more than 1 observation.
sep	Character. String text wished to insert between values.
quotes	Boolean. Bring simple quotes for each observation.
force_single	Boolean. Force single quotes by replacing \".
and	Character. Add 'and' or something before last observation. Not boolean variable so it can be used on other languages. Note that the last comma will be suppressed if Sys.getenv("LARES_NUMFORMAT") is set to 1 and you have less than 3 values.

Value

Vector pasting vector values into a single string

See Also

Other Data Wrangling: balance_data(), categ_reducer(), cleanText(), date_cuts(), date_feats(), file_name(), formatHTML(), holidays(), impute(), left(), normalize(), num_abbr(), ohe_commas(), ohse(), quants(), removenacols(), replaceall(), replacefactor(), textFeats(), textTokenizer(), year_month(), zerovar()

Examples

```
vector2text(LETTERS[1:5])
vector2text(c(1:5), quotes = FALSE)
vector2text(c(1:5), quotes = FALSE, sep = "-")
vector2text(c(1:5), and = "and also")
vector2text(c("Text", "R's"), force_single = TRUE)
# Shorter function with same purpose
v2t(LETTERS[1:5])
```

---

warnifnot *Test the Truth of R Expressions and Warn*

Description

If the expression in ... is not TRUE, warning is called, producing a warning message indicating the expression which was not true.

Usage

```
warnifnot(...)
```

Arguments

...	any R expression, which should evaluate to TRUE

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), what_size()

Examples

```
warnifnot(TRUE)
warnifnot(FALSE)
warnifnot(1 + 1 == 3)
```

---

what_size *Calculate the size of any R object*

Description

Calculate the size of any R object

Usage

```
what_size(x, units = "Mb", ...)
```

Arguments

x	Object
units	Character. Specify which unit to use, i.e. "Gb", "Mb", "Kb".
...	further arguments passed to or from other methods.

See Also

Other Tools: autoline(), bind_files(), bring_api(), db_download(), db_upload(), export_plot(), export_results(), files_functions(), font_exists(), formatColoured(), formatHTML(), get_credentials(), glued(), grepm(), h2o_selectmodel(), haveInternet(), image_metadata(), importxlsx(), ip_data(), json2vector(), list_cats(), listfiles(), mail_send(), markdown2df(), move_files(), msplit(), myip(), quiet(), read.file(), statusbar(), tic(), try_require(), updateLares(), warnifnot()

Examples

```
what_size(seq(1:1e3), "Kb")
what_size(seq(1:1e6))
what_size(as.character(seq(1:1e6)))
```

winsorize *Outliers: Winsorize*

Description

Winsorizing a vector means that a predefined quantum of the smallest and/or the largest values are replaced by less extreme values. Thereby the substitute values are the most extreme retained values.

Usage

```
winsorize(x, thresh = c(0.05, 0.95), na.rm = FALSE)
```

Arguments

x	Numeric vector. Distribution to be winsorized.
thresh	Numeric vector. Lower and upper quantiles thresholds. Set values within [0,1].
na.rm	Boolean. Should NA be omitted to calculate the quantiles? Note that NA in x are preserved and left unchanged anyway.

Value

Numeric vector transformed.

See Also

Other Outliers: outlier_turkey(), outlier_zscore_plot(), outlier_zscore()

wordle_check *Wordle Game Validation*

Description

Given and input and a word, validate each letter based on Wordle's rules: correct letter in correct placement (green), correct letter in wrong placement (yellow), letter is not present (red).

Usage

```
wordle_check(
  input,
  word,
  dictionary = NULL,
  lang_dic = "en",
  method = 3,
  print = TRUE
)
```

```
## S3 method for class 'wordle_check'
print(x, print = TRUE, ...)

wordle_dictionary(lang_dic = "en", method = 3, quiet = TRUE)

wordle_simulation(input, word, seed = NULL, quiet = FALSE, ...)

## S3 method for class 'wordle_simulation'
print(x, type = 1, ...)
```

Arguments

input	Character. Word to validate (5-letters)
word	Character. Word actually answer (5-letters).
dictionary	Character vector. List of valid words. If set to NULL then will use modified scrabble_dictionary() to fetch 5 letter words. Use lang_dic param to set language.
lang_dic	Character. Any of: "en", "es". Only used when dictionary parameter is NULL. Requires internet connection the first time. Uses cache.
method	Integer. 1 for scrabble_dictionary(), 3 for scrapping the words taken straight from the game's source code.
print	Boolean. Print validation results?
x	Object to print
...	Additional parameters to pass.
quiet	Boolean. Do not print words as they are being searched.
seed	Numeric. For reproducibility. Accepts more than one: will run as many seeds there are.
type	Integer. 1 for summary and 2 for coloured results.

Value

Invisible vector with results by letter.

Examples

```
word <- "ABBEY"
# Or pick a random one:
# word <- sample(wordle_dictionary("en"), 1)
wordle_check("OPENS", word)
wordle_check("BABES", word)
wordle_check("KEBAB", word, print = FALSE)
wordle_check("ABYSS", word)
wordle_check("ABBEY", word)
# Feel free to use scrabble_words() for hints

x <- wordle_simulation(input = "SAINT", word = "ABBEY", seed = 1:3)
```

```
print(x)
# hist(sapply(x, function(x) x$iters))
```

---

x2y *Ranked Predictive Power of Cross-Features (x2y)*

Description

The relative reduction in error when we go from a baseline model (average for continuous and most frequent for categorical features) to a predictive model, can measure the strength of the relationship between two features. In other words, x2y measures the ability of x to predict y. We use CART (Classification And Regression Trees) models to be able to 1) compare numerical and non-numerical features, 2) detect non-linear relationships, and 3) because they are easy/quick to train.

Usage

```
x2y(
  df,
  target = NULL,
  symmetric = FALSE,
  target_x = FALSE,
  target_y = FALSE,
  plot = FALSE,
  top = 20,
  quiet = "auto",
  ohse = FALSE,
  corr = FALSE,
  ...
)

x2y_metric(x, y, confidence = FALSE, bootstraps = 20, max_cat = 20)

## S3 method for class 'x2y_preds'
plot(x, corr = FALSE, ...)

## S3 method for class 'x2y'
plot(x, type = 1, ...)

x2y_preds(x, y, max_cat = 10)
```

Arguments

df	data.frame. Note that variables with no variance will be ignored.
target	Character vector. If you are only interested in the x2y values between particular variable(s) in df, set name(s) of the variable(s) you are interested in. Keep NULL to calculate for every variable (column). Check target_x and target_y parameters as well.

symmetric	Boolean. x2y metric is not symmetric with respect to x and y. The extent to which x can predict y can be different from the extent to which y can predict x. Set symmetric=TRUE if you wish to average both numbers.
target_x, target_y	Boolean. Force target features to be part of x OR y?
plot	Boolean. Return a plot? If not, only a data.frame with calculated results will be returned.
top	Integer. Show/plot only top N predictive cross-features. Set to NULL to return all.
quiet	Boolean. Keep quiet? If not, show progress bar.
ohse	Boolean. Use lares::ohse() to pre-process the data?
corr	Boolean. Add correlation and pvalue data to compare with? For more custom studies, use lares::corr_cross() directly.
...	Additional parameters passed to x2y_metric()
x, y	Vectors. Categorical or numerical vectors of same length.
confidence	Boolean. Calculate 95% confidence intervals estimated with N bootstraps.
bootstraps	Integer. If confidence=TRUE, how many bootstraps? The more iterations we run the more precise the confidence internal will be.
max_cat	Integer. Maximum number of unique x or y values when categorical. Will select then most frequent values and the rest will be passed as "".
type	Integer. Plot type: 1 for tile plot, 2 for ranked bar plot.

Details

This x2y metric is based on Rama Ramakrishnan's post: An Alternative to the Correlation Coefficient That Works For Numeric and Categorical Variables. This analysis complements our lares::corr_cross() output.

Value

Depending on plot input, a plot or a data.frame with x2y results.

Examples

```
data(dft) # Titanic dataset
x2y_results <- x2y(dft, quiet = TRUE, max_cat = 10, top = NULL)
head(x2y_results, 10)
plot(x2y_results, type = 2)

# Confidence intervals with 10 bootstrap iterations
x2y(dft,
  target = c("Survived", "Age"),
  confidence = TRUE, bootstraps = 10, top = 8
)

# Compare with mean absolute correlations
```

```
x2y(dft, "Fare", corr = TRUE, top = 6, target_x = TRUE)

# Plot (symmetric) results
symm <- x2y(dft, target = "Survived", symmetric = TRUE)
plot(symm, type = 1)

# Symmetry: x2y vs y2x
on.exit(set.seed(42))
x <- seq(-1, 1, 0.01)
y <- sqrt(1 - x^2) + rnorm(length(x), mean = 0, sd = 0.05)

# Knowing x reduces the uncertainty about the value of y a lot more than
# knowing y reduces the uncertainty about the value of x. Note correlation.
plot(x2y_preds(x, y), corr = TRUE)
plot(x2y_preds(y, x), corr = TRUE)
```

`year_month` *Convert Date into Year-Month, Year-Quarter or Year-Week Format*

Description

This function lets the user convert a date into YYYY-MM, YYYY-QX, or YYYY-WW format easily.

Usage

```
year_month(date)

year_quarter(date)

year_week(date)
```

Arguments

`date`	Date vector. Date to transform format.

Value

Vector with dates reformatted

See Also

Other Data Wrangling: `balance_data()`, `categ_reducer()`, `cleanText()`, `date_cuts()`, `date_feats()`, `file_name()`, `formatHTML()`, `holidays()`, `impute()`, `left()`, `normalize()`, `num_abbr()`, `ohe_commas()`, `ohse()`, `quants()`, `removenacols()`, `replaceall()`, `replacefactor()`, `textFeats()`, `textTokenizer()`, `vector2text()`, `zerovar()`

Examples

```
year_month(Sys.Date())
year_quarter(Sys.Date())
year_week(Sys.Date())
```

`zerovar` *Zero Variance Columns*

Description

This function detects which columns have the same value (whichever) for each column.

Usage

```
zerovar(df)
```

Arguments

`df`	Dataframe

Value

Character vector with column names on which its values have no variance.

See Also

Other Data Wrangling: `balance_data()`, `categ_reducer()`, `cleanText()`, `date_cuts()`, `date_feats()`, `file_name()`, `formatHTML()`, `holidays()`, `impute()`, `left()`, `normalize()`, `num_abbr()`, `ohe_commas()`, `ohse()`, `quants()`, `removenacols()`, `replaceall()`, `replacefactor()`, `textFeats()`, `textTokenizer()`, `vector2text()`, `year_month()`

Examples

```
df <- data.frame(a = c(1, NA, 3), b = rep(NA, 3), c = rep(5, 3))
print(df)
zerovar(df)
```

Index