Package ‘theft’

November 17, 2022

Type Package

Title Tools for Handling Extraction of Features from Time Series

Version 0.4.1.1

Date 2022-11-15

Maintainer Trent Henderson <then6675@uni.sydney.edu.au>


BugReports https://github.com/hendersontrent/theft/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports rlang, stats, dplyr, ggplot2, tidyr, reshape2, scales, tibble, purrr, broom, tsibble, fabletools, tsfeatures, feasts, Rcatch22, reticulate, Rtsne, R.matlab, plotly, caret, janitor

Suggests lifecycle, cachem, bslib, knitr, markdown, rmarkdown, pkgdown, testthat

RoxygenNote 7.2.2
R topics documented:

calculate_features ...................................................... 3
check_vector_quality ..................................................... 4
compute_top_features .................................................... 4
demo_multi_outputs ...................................................... 7
demo_outputs ............................................................. 7
feature_list .............................................................. 8
fit_multi_feature_classifier ........................................... 8
fit_single_feature_classifier ......................................... 10
init_theft ................................................................. 12
minmax_scaler ............................................................. 13
normalise_feature_frame ................................................. 13
normalise_feature_vector ............................................... 14
normalize_feature_frame ............................................... 15
normalize_feature_vector .............................................. 16
plot_all_features ........................................................ 17
plot_feature_correlations .............................................. 18
plot_feature_matrix ..................................................... 20
plot_low_dimension ..................................................... 21
plot_quality_matrix .................................................... 23
plot_ts_correlations .................................................... 24
process_hctsa_file ..................................................... 25
robustsigmoid_scaler ................................................... 26
sigmoid_scaler ........................................................... 26
simData ................................................................. 27
theft ................................................................. 27
zscore_scaler ............................................................ 28

Index 29
**calculate_features**  
*Compute features on an input time series dataset*

**Description**
Compute features on an input time series dataset

**Usage**
```r
calculate_features(
  data,
  id_var = NULL,
  time_var = NULL,
  values_var = NULL,
  group_var = NULL,
  feature_set = c("catch22", "feasts", "tsfeatures", "Kats", "tsfresh", "TSFEL"),
  catch24 = FALSE,
  tsfresh_cleanup = FALSE,
  seed = 123
)
```

**Arguments**
- `data` a dataframe with at least 4 columns: id variable, group variable, time variable, value variable
- `id_var` a string specifying the ID variable to identify each time series. Defaults to NULL
- `time_var` a string specifying the time index variable. Defaults to NULL
- `values_var` a string specifying the values variable. Defaults to NULL
- `group_var` a string specifying the grouping variable that each unique series sits under (if one exists). Defaults to NULL
- `feature_set` the set of time-series features to calculate. Defaults to catch22
- `catch24` a Boolean specifying whether to compute catch24 in addition to catch22 if catch22 is one of the feature sets selected. Defaults to FALSE
- `tsfresh_cleanup` a Boolean specifying whether to use the in-built tsfresh relevant feature filter or not. Defaults to FALSE
- `seed` fixed number for R’s random number generator to ensure reproducibility

**Value**
object of class dataframe that contains the summary statistics for each feature

**Author(s)**
- Trent Henderson
Examples

```r
featMat <- calculate_features(data = simData,
   id_var = "id",
   time_var = "timepoint",
   values_var = "values",
   group_var = "process",
   feature_set = "catch22",
   seed = 123)
```

---

**check_vector_quality**  
*Check data quality of a vector*

**Description**

Check data quality of a vector.

**Usage**

```r
check_vector_quality(x)
```

**Arguments**

- `x`  
  input data vector

**Value**

Boolean of whether the data is good to extract features on or not.

**Examples**

```r
x <- stats::rnorm(10)
check_vector_quality(x)
```

---

**compute_top_features**  
*Return an object containing results from top-performing features on a classification task*

**Description**

Return an object containing results from top-performing features on a classification task.
compute_top_features

Usage

```r
compute_top_features(
  data,
  id_var = "id",
  group_var = "group",
  num_features = 40,
  normalise_violin_plots = FALSE,
  method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax"),
  cor_method = c("pearson", "spearman"),
  test_method = "gaussprRadial",
  clust_method = c("average", "ward.D", "ward.D2", "single", "complete", "mcquitty", "median", "centroid"),
  use_balanced_accuracy = FALSE,
  use_k_fold = FALSE,
  num_folds = 10,
  use_empirical_null = FALSE,
  null_testing_method = c("ModelFreeShuffles", "NullModelFits"),
  p_value_method = c("empirical", "gaussian"),
  num_permutations = 50,
  pool_empirical_null = FALSE,
  seed = 123
)
```

Arguments

- **data**: the dataframe containing the raw feature matrix
- **id_var**: a string specifying the ID variable to group data on (if one exists). Defaults to "id"
- **group_var**: a string specifying the grouping variable that the data aggregates to. Defaults to "group"
- **num_features**: the number of top features to retain and explore. Defaults to 40
- **normalise_violin_plots**: a Boolean of whether to normalise features before plotting. Defaults to FALSE
- **method**: a rescaling/normalising method to apply to violin plots. Defaults to "RobustSigmoid"
- **cor_method**: the correlation method to use. Defaults to "pearson"
- **test_method**: the algorithm to use for quantifying class separation. Defaults to "gaussprRadial"
- **clust_method**: the hierarchical clustering method to use for the pairwise correlation plot. Defaults to "average"
- **use_balanced_accuracy**: a Boolean specifying whether to use balanced accuracy as the summary metric for caret model training. Defaults to FALSE
- **use_k_fold**: a Boolean specifying whether to use k-fold procedures for generating a distribution of classification accuracy estimates if a caret model is specified for test_method. Defaults to FALSE
- **num_folds**: an integer specifying the number of k-folds to perform if use_k_fold is set to TRUE. Defaults to 10
use_empirical_null
   a Boolean specifying whether to use empirical null procedures to compute p-
   values if a caret model is specified for test_method. Defaults to FALSE
null_testing_method
   a string specifying the type of statistical method to use to calculate p-values.
   Defaults to model free shuffles
p_value_method a string specifying the method of calculating p-values. Defaults to "empirical"
num_permutations
   an integer specifying the number of class label shuffles to perform if use_empirical_null
   is TRUE. Defaults to 50
pool_empirical_null
   a Boolean specifying whether to use the pooled empirical null distribution of all
   features or each features’ individual empirical null distribution if a caret model
   is specified for test_method use_empirical_null is TRUE. Defaults to FALSE
seed
   fixed number for R’s random number generator to ensure reproducibility

Value
   an object of class list containing a dataframe of results, a feature x feature matrix plot, and a violin
   plot

Author(s)
   Trent Henderson

Examples

featMat <- calculate_features(data = simData,
id_var = "id",
time_var = "timepoint",
values_var = "values",
group_var = "process",
feature_set = "catch22",
seed = 123)

compute_top_features(featMat,
id_var = "id",
group_var = "group",
um_features = 10,
normalise_violin_plots = FALSE,
method = "RobustSigmoid",
cor_method = "pearson",
test_method = "gaussprRadial",
clust_method = "average",
use_balanced_accuracy = FALSE,
use_k_fold = FALSE,
um_folds = 10,
use_empirical_null = TRUE,
null_testing_method = "ModelFreeShuffles",
demo_multi_outputs

p_value_method = "gaussian",
num_permutations = 100,
pool_empirical_null = FALSE,
seed = 123)

demo_multi_outputs  Computed values for multi-feature classification results for use in vignette

Description
Format is:

Usage
demo_multi_outputs

Format
A list object

FeatureSetResultsPlot  ggplot comparing feature set classification accuracy
TestStatistics  data.frame of test statistics
RawClassificationResults  data.frame of raw classification accuracy

demo_outputs  Computed values for top features results for use in vignette

Description
Format is:

Usage
demo_outputs

Format
A list object

ResultsTable  data.frame of results for top features
FeatureFeatureCorrelationPlot  ggplot heatmap of feature-feature correlation matrix
ViolinPlots  ggplot of distributions rendered as violins for top features
feature_list  
*All features available in theft in tidy format*

Description

The variables include:

Usage

`feature_list`

Format

A tidy dataframe with 2 variables:

- **feature_set** Name of the set the feature is from
- **feature** Name of the feature

---

fit_multi_feature_classifier

*Fit a classifier to feature matrix using all features or all features by set*

Description

Fit a classifier to feature matrix using all features or all features by set

Usage

```r
fit_multi_feature_classifier(
  data,
  id_var = "id",
  group_var = "group",
  by_set = FALSE,
  test_method = "gaussprRadial",
  use_balanced_accuracy = FALSE,
  use_k_fold = TRUE,
  num_folds = 10,
  use_empirical_null = FALSE,
  null_testing_method = c("ModelFreeShuffles", "NullModelFits"),
  p_value_method = c("empirical", "gaussian"),
  num_permutations = 100,
  seed = 123
)
```
Arguments

- **data**: the dataframe containing the raw feature data as calculated by `theft::calculate_features`.
- **id_var**: a string specifying the ID variable to group data on (if one exists). Defaults to "id".
- **group_var**: a string specifying the grouping variable that the data aggregates to. Defaults to "group".
- **by_set**: Boolean specifying whether to compute classifiers for each feature set. Defaults to FALSE.
- **test_method**: the algorithm to use for quantifying class separation. Defaults to "gaussprRadial".
- **use_balanced_accuracy**: a Boolean specifying whether to use balanced accuracy as the summary metric for caret model training. Defaults to FALSE.
- **use_k_fold**: a Boolean specifying whether to use k-fold procedures for generating a distribution of classification accuracy estimates. Defaults to TRUE.
- **num_folds**: an integer specifying the number of folds (train-test splits) to perform if `use_k_fold` is set to TRUE. Defaults to 10.
- **use_empirical_null**: a Boolean specifying whether to use empirical null procedures to compute p-values. Defaults to FALSE.
- **null_testing_method**: a string specifying the type of statistical method to use to calculate p-values. Defaults to model free shuffles.
- **p_value_method**: a string specifying the method of calculating p-values. Defaults to "empirical".
- **num_permutations**: an integer specifying the number of class label shuffles to perform if `use_empirical_null` is TRUE. Defaults to 100.
- **seed**: fixed number for R’s random number generator to ensure reproducibility.

Value

an object of class list containing dataframe summaries of the classification models and a ggplot object if `by_set` is TRUE.

Author(s)

Trent Henderson

Examples

```r
featMat <- calculate_features(data = simData,
   id_var = "id",
   time_var = "timepoint",
   values_var = "values",
   group_var = "process",
   feature_set = "catch22",
   by_set = TRUE,
   test_method = "gaussprRadial",
   use_k_fold = TRUE,
   num_folds = 10,
   use_balanced_accuracy = TRUE,
   use_empirical_null = TRUE,
   num_permutations = 100,
   seed = 123)
```
fit_single_feature_classifier

Fit a classifier to feature matrix to extract top performers

Description

Fit a classifier to feature matrix to extract top performers

Usage

```r
fit_single_feature_classifier(
  data,
  id_var = "id",
  group_var = "group",
  test_method = "gaussprRadial",
  use_balanced_accuracy = FALSE,
  use_k_fold = FALSE,
  num_folds = 10,
  use_empirical_null = FALSE,
  null_testing_method = c("ModelFreeShuffles", "NullModelFits"),
  p_value_method = c("empirical", "gaussian"),
  num_permutations = 50,
  seed = 123
)
```

Arguments

- `data`: the dataframe containing the raw feature matrix
- `id_var`: a string specifying the ID variable to group data on (if one exists). Defaults to "id"
fit_single_feature_classifier

- **group_var**: a string specifying the grouping variable that the data aggregates to. Defaults to "group"
- **test_method**: the algorithm to use for quantifying class separation. Defaults to "gaussprRadial". Should be either "t-test", "wilcox", or "binomial logistic" for two-class problems to obtain exact statistics, or a valid caret classification model for everything else
- **use_balanced_accuracy**: a Boolean specifying whether to use balanced accuracy as the summary metric for caret model training. Defaults to FALSE
- **use_k_fold**: a Boolean specifying whether to use k-fold procedures for generating a distribution of classification accuracy estimates if a caret model is specified for test_method. Defaults to FALSE
- **num_folds**: an integer specifying the number of k-folds to perform if use_k_fold is set to TRUE. Defaults to 10
- **use_empirical_null**: a Boolean specifying whether to use empirical null procedures to compute p-values if a caret model is specified for test_method. Defaults to FALSE
- **null_testing_method**: a string specifying the type of statistical method to use to calculate p-values. Defaults to model free shuffles
- **p_value_method**: a string specifying the method of calculating p-values. Defaults to "empirical"
- **num_permutations**: an integer specifying the number of class label shuffles to perform if use_empirical_null is TRUE. Defaults to 50
- **pool_empirical_null**: a Boolean specifying whether to use the pooled empirical null distribution of all features or each features' individual empirical null distribution if a caret model is specified for test_method use_empirical_null is TRUE. Defaults to FALSE
- **seed**: fixed number for R’s random number generator to ensure reproducibility

**Value**

- an object of class dataframe containing results

**Author(s)**

Trent Henderson

**Examples**

```r
featMat <- calculate_features(data = simData,
   id_var = "id",
   time_var = "timepoint",
   values_var = "values",
   group_var = "process",
   feature_set = "catch22",
   seed = 123)
```
# Mimic machinery of theft::compute_top_features
# which calls fit_single_feature_classifier and
# does these operations prior

featMat$group <- make.names(featMat$group)
featMat$group <- as.factor(featMat$group)
featMat$values <- as.numeric(featMat$values)

fit_single_feature_classifier(featMat,
    id_var = "id",
    group_var = "group",
    test_method = "gaussprRadial",
    use_balanced_accuracy = FALSE,
    use_k_fold = TRUE,
    num_folds = 10,
    use_empirical_null = TRUE,
    null_testing_method = "ModelFreeShuffles",
    p_value_method = "gaussian",
    num_permutations = 50,
    pool_empirical_null = FALSE,
    seed = 123)

---

**init_theft**

Communicate to R the correct Python version containing the relevant libraries for calculating features

**Description**

Communicate to R the correct Python version containing the relevant libraries for calculating features

**Usage**

`init_theft(path_to_python)`

**Arguments**

- `path_to_python` a string specifying the filepath to the version of Python containing the relevant libraries for calculating features

**Value**

no return value; called for side effects

**Author(s)**

Trent Henderson
**minmax_scaler**

This function rescales a vector of numerical values into the unit interval [0,1]

**Usage**

```
minmax_scaler(x)
```

**Arguments**

- `x`: a numeric vector, preferably of feature values computed by other `theft` package functions

**Value**

- `x`: a numeric vector, rescaled into the [0,1] unit interval

**Author(s)**

Trent Henderson

**Examples**

```
minmax_scaler(stats::rnorm(10))
```

**normalise_feature_frame**

Scale each feature vector into a user-specified range for visualisation and modelling

**Description**

Scale each feature vector into a user-specified range for visualisation and modelling

**Usage**

```
normalise_feature_frame(
  data,
  names_var = "names",
  values_var = "values",
  method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax")
)
```
normalise_feature_vector

Arguments

data a dataframe with at least 2 columns: names variable (feature names) and value variable

names_var a string denoting the name of the variable/column that holds the feature names. Defaults to "names"

values_var a string denoting the name of the variable/column that holds the numerical feature values. Defaults to "values"

method a rescaling/normalising method to apply. Defaults to "RobustSigmoid"

Value

a dataframe with the value column rescaled into the specified range

Author(s)

Trent Henderson

Examples

```r
teMat <- calculate_features(data = simData,
  id_var = "id",
  time_var = "timepoint",
  values_var = "values",
  group_var = "process",
  feature_set = "catch22",
  seed = 123)

normed <- normalise_feature_frame(featMat,
  names_var = "names",
  values_var = "values",
  method = "RobustSigmoid")
```

normalise_feature_vector

Scale each value into a user-specified range for visualisation and analysis

Description

Scale each value into a user-specified range for visualisation and analysis

Usage

```r
normalise_feature_vector(
  x,
  method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax")
)
```
normalize_feature_frame

Arguments

x a vector of scalar values
method a rescaling/normalising method to apply. Defaults to "RobustSigmoid"

Value

a vector of scalar values normalised into the selected range

Author(s)

Trent Henderson

Examples

featMat <- calculate_features(data = simData,
    id_var = "id",
    time_var = "timepoint",
    values_var = "values",
    group_var = "process",
    feature_set = "catch22",
    seed = 123)

x <- featMat[featMat$names == "DN_HistogramMode_5", ]
xnormed <- normalise_feature_vector(x$values, method = "RobustSigmoid")

normalize_feature_frame

Scale each feature vector into a user-specified range for visualisation and modelling

Description

Scale each feature vector into a user-specified range for visualisation and modelling

Usage

normalize_feature_frame(  
data,  
    names_var = "names",  
    values_var = "values",  
    method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax")
)
normalize_feature_vector

Scale each value into a user-specified range for visualisation and analysis

Description

Scale each value into a user-specified range for visualisation and analysis

Usage

normalize_feature_vector(
  x,
  method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax")
)
plot_all_features

Arguments

x a vector of scalar values
method a rescaling/normalising method to apply. Defaults to "RobustSigmoid"

Value

a vector of scalar values normalised into the selected range

Author(s)

Trent Henderson

Examples

featMat <- calculate_features(data = simData,
id_var = "id",
time_var = "timepoint",
values_var = "values",
group_var = "process",
feature_set = "catch22",
seed = 123)
x <- featMat[featMat$names == "DN_HistogramMode_5", ]
xnormed <- normalise_feature_vector(x$values, method = "RobustSigmoid")

plot_all_features

Produce a heatmap matrix of the calculated feature value vectors and each unique time series with automatic hierarchical clustering.

Description

Produce a heatmap matrix of the calculated feature value vectors and each unique time series with automatic hierarchical clustering.

Usage

plot_all_features(
data,
is_normalised = FALSE,
id_var = "id",
method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax"),
clust_method = c("average", "ward.D", "ward.D2", "single", "complete", "mcquitty", "median", "centroid"),
interactive = FALSE
)
Arguments

- `data` a dataframe with at least 2 columns called "names" and "values"
- `is_normalised` a Boolean as to whether the input feature values have already been scaled. Defaults to FALSE
- `id_var` a string specifying the ID variable to identify each time series. Defaults to "id"
- `method` a rescaling/normalising method to apply. Defaults to "RobustSigmoid"
- `clust_method` the hierarchical clustering method to use for the pairwise correlation plot. Defaults to "average"
- `interactive` a Boolean as to whether to plot an interactive plotly graphic. Defaults to FALSE

Value

an object of class ggplot that contains the heatmap graphic

Author(s)

Trent Henderson

Examples

```r
featMat <- calculate_features(data = simData,
   id_var = "id",
   time_var = "timepoint",
   values_var = "values",
   group_var = "process",
   feature_set = "catch22",
   seed = 123)

plot_all_features(featMat,
   is_normalised = FALSE,
   id_var = "id",
   method = "RobustSigmoid",
   clust_method = "average",
   interactive = FALSE)
```

Description

Produce a correlation matrix plot showing pairwise correlations of feature vectors by unique id with automatic hierarchical clustering.
Usage

plot_feature_correlations(
  data,
  is_normalised = NULL,
  id_var = "id",
  names_var = "names",
  values_var = "values",
  method = NULL,
  cor_method = c("pearson", "spearman"),
  clust_method = c("average", "ward.D", "ward.D2", "single", "complete", "mcquitty",
                   "median", "centroid"),
  interactive = FALSE
)

Arguments

data a dataframe with at least 3 columns for 'id', 'names' and 'values'

is_normalised deprecated as of 0.4.0; do not use

id_var a string specifying the ID variable to compute pairwise correlations between. Defaults to "id"

names_var a string denoting the name of the variable/column that holds the feature names. Defaults to "names"

values_var a string denoting the name of the variable/column that holds the numerical feature values. Defaults to "values"

method deprecated as of 0.4.0; do not use

cor_method the correlation method to use. Defaults to "pearson"

clust_method the hierarchical clustering method to use for the pairwise correlation plot. Defaults to "average"

interactive a Boolean as to whether to plot an interactive plotly graphic. Defaults to FALSE

Value

an object of class ggplot that contains the correlation matrix graphic

Author(s)

Trent Henderson

Examples

featMat <- calculate_features(data = simData,
  id_var = "id",
  time_var = "timepoint",
  values_var = "values",
  group_var = "process",
  feature_set = "catch22",
  seed = 123)
plot_feature_matrix

Produce a heatmap matrix of the calculated feature value vectors and each unique time series with automatic hierarchical clustering.

Description

Produce a heatmap matrix of the calculated feature value vectors and each unique time series with automatic hierarchical clustering.

Usage

plot_feature_matrix(
data,  
is_normalised = FALSE, 
id_var = "id",  
method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax"),  
clust_method = c("average", "ward.D", "ward.D2", "single", "complete", "mcquitty", "median", "centroid"),  
interactive = FALSE
)

Arguments

data a dataframe with at least 2 columns called "names" and "values"
is_normalised a Boolean as to whether the input feature values have already been scaled. Defaults to FALSE
id_var a string specifying the ID variable to identify each time series. Defaults to "id"
method a rescaling/normalising method to apply. Defaults to "RobustSigmoid"
clust_method the hierarchical clustering method to use for the pairwise correlation plot. Defaults to "average"
interactive a Boolean as to whether to plot an interactive plotly graphic. Defaults to FALSE

Value

an object of class ggplot that contains the heatmap graphic
Author(s)

Trent Henderson

Examples

```r
featMat <- calculate_features(data = simData,
   id_var = "id",
   time_var = "timepoint",
   values_var = "values",
   group_var = "process",
   feature_set = "catch22",
   seed = 123)

plot_feature_matrix(featMat,
   is_normalised = FALSE,
   id_var = "id",
   method = "RobustSigmoid",
   clust_method = "average",
   interactive = FALSE)
```

Description

Produce a principal components analysis (PCA) on normalised feature values and render a bivariate plot to visualise it

Usage

```r
plot_low_dimension(
   data,
   is_normalised = FALSE,
   id_var = "id",
   group_var = NULL,
   method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax"),
   low_dim_method = c("PCA", "t-SNE"),
   perplexity = 30,
   plot = TRUE,
   show_covariance = FALSE,
   seed = 123
)
```
Arguments

- **data**: a dataframe with at least 2 columns called "names" and "values"
- **is_normalised**: a Boolean as to whether the input feature values have already been scaled. Defaults to FALSE
- **id_var**: a string specifying the ID variable to uniquely identify each time series. Defaults to "id"
- **group_var**: a string specifying the grouping variable that the data aggregates to (if one exists). Defaults to NULL
- **method**: a rescaling/normalising method to apply. Defaults to "z-score"
- **low_dim_method**: the low dimensional embedding method to use. Defaults to "PCA"
- **perplexity**: the perplexity hyperparameter to use if t-SNE algorithm is selected. Defaults to 30
- **plot**: a Boolean as to whether a plot or model fit information should be returned. Defaults to TRUE
- **show_covariance**: a Boolean as to whether covariance ellipses should be shown on the plot. Defaults to FALSE
- **seed**: fixed number for R’s random number generator to ensure reproducibility

Value

If `plot = TRUE`, returns an object of class `ggplot`, if `plot = FALSE` returns an object of class dataframe with PCA results

Author(s)

Trent Henderson

Examples

```r
featMat <- calculate_features(data = simData,
   id_var = "id",
   time_var = "timepoint",
   values_var = "values",
   group_var = "process",
   feature_set = "catch22",
   seed = 123)

plot_low_dimension(featMat,
   is_normalised = FALSE,
   id_var = "id",
   group_var = "group",
   method = "RobustSigmoid",
   low_dim_method = "PCA",
   plot = TRUE,
   show_covariance = TRUE,
   seed = 123)
```
plot_quality_matrix

Produce a matrix visualisation of data types computed by feature calculation function.

Description

Produce a matrix visualisation of data types computed by feature calculation function.

Usage

plot_quality_matrix(data, ignore_good_features = FALSE)

Arguments

data  a dataframe with at least 2 columns called "names" and "values"
ignore_good_features

   Boolean whether to remove "good" features (i.e., successful numeric values)
   from the plot. Defaults to FALSE

Value

an object of class ggplot

Author(s)

Trent Henderson

Examples

featMat <- calculate_features(data = simData,
   id_var = "id",
   time_var = "timepoint",
   values_var = "values",
   group_var = "process",
   feature_set = "catch22",
   seed = 123)

plot_quality_matrix(data = featMat,
   ignore_good_features = FALSE)
**plot_ts_correlations**

Produce a correlation matrix plot showing pairwise correlations of time series with automatic hierarchical clustering

**Description**

Produce a correlation matrix plot showing pairwise correlations of time series with automatic hierarchical clustering

**Usage**

```r
plot_ts_correlations(
  data,  
  is_normalised = NULL,  
  id_var = "id",  
  time_var = "timepoint",  
  values_var = "values",  
  method = NULL,  
  clust_method = c("average", "ward.D", "ward.D2", "single", "complete", "mcquitty", "median", "centroid"),  
  cor_method = c("pearson", "spearman"),  
  interactive = FALSE  
)
```

**Arguments**

- **data** a data frame with at least 2 columns for "id" and "values" variables
- **is_normalised** deprecated as of 0.4.0; do not use
- **id_var** a string specifying the ID variable to compute pairwise correlations between. Defaults to "id"
- **time_var** a string specifying the time index variable. Defaults to NULL
- **values_var** a string denoting the name of the variable/column that holds the numerical feature values. Defaults to "values"
- **method** deprecated as of 0.4.0; do not use
- **clust_method** the hierarchical clustering method to use for the pairwise correlation plot. Defaults to "average"
- **cor_method** the correlation method to use. Defaults to "pearson"
- **interactive** a Boolean as to whether to plot an interactive plotly graphic. Defaults to FALSE

**Value**

an object of class ggplot

**Author(s)**

Trent Henderson
process_hctsa_file

Examples

```r
plot_ts_correlations(data = simData,
                     id_var = "id",
                     time_var = "timepoint",
                     values_var = "values",
                     method = "RobustSigmoid",
                     cor_method = "pearson",
                     clust_method = "average",
                     interactive = FALSE)
```

Description

Load in hctska formatted MATLAB files of time series data into a tidy format ready for feature extraction

Usage

```r
process_hctsa_file(data)
```

Arguments

data a string specifying the filepath to the MATLAB file to parse

Value

an object of class dataframe in tidy format

Author(s)

Trent Henderson

Examples

```r
myfile <- process_hctsa_file(
  "https://cloudstor.aarnet.edu.au/plus/s/6sRD6IPMjyZLNn/download"
)
```
### Description

This function rescales a vector of numerical values with an outlier-robust Sigmoidal transformation.

### Usage

```r
robustsigmoid_scaler(x, unitInt = TRUE)
```

### Arguments

- `x`: a numeric vector, preferably of feature values computed by other theft package functions.
- `unitInt`: Boolean whether to rescale Sigmoidal outputs into unit interval \([0, 1]\). Defaults to TRUE.

### Value

`x` a numeric rescaled vector.

### Author(s)

Trent Henderson

### Examples

```r
robustsigmoid_scaler(stats::rnorm(10))
```

---

### Description

This function rescales a vector of numerical values with a Sigmoidal transformation.

### Usage

```r
sigmoid_scaler(x, unitInt = TRUE)
```

### Value

`x` a numeric rescaled vector.
Arguments

- **x**: a numeric vector, preferably of feature values computed by other `theft` package functions.
- **unitInt**: Boolean whether to rescale Sigmoidal outputs into unit interval \([0, 1]\). Defaults to `TRUE`.

Value

- **x**: a numeric rescaled vector.

Author(s)

Trent Henderson

Examples

```r
sigmoid_scaler(stats::rnorm(10))
```

---

**simData**

*Sample of randomly-generated time series to produce function tests and vignettes*

---

**Description**

The variables include:

**Usage**

`simData`

**Format**

A tidy dataframe with 4 variables:

- **id**: Unique identifier for the time series
- **timepoint**: Time index
- **values**: Value
- **process**: Group label for the type of time series

---

**theft**

*Tools for Handling Extraction of Features from Time-series*

---

**Description**

Tools for Handling Extraction of Features from Time-series
zscore scaler

This function rescales a vector of numerical values into z-scores

Description

This function rescales a vector of numerical values into z-scores

Usage

zscore_scaler(x, unitInt = TRUE)

Arguments

x a numeric vector, preferably of feature values computed by other theft package functions

unitInt Boolean whether to rescale outputs into unit interval [0,1]. Defaults to TRUE

Value

x a numeric vector, rescaled into z-scores

Author(s)

Trent Henderson

Examples

zscore_scaler(stats::rnorm(10))
Index

* datasets
  demo_multi_outputs, 7
  demo_outputs, 7
  feature_list, 8
  simData, 27

  calculate_features, 3
  check_vector_quality, 4
  compute_top_features, 4

  demo_multi_outputs, 7
  demo_outputs, 7

  feature_list, 8
  fit_multi_feature_classifier, 8
  fit_single_feature_classifier, 10

  init_theft, 12

  minmax_scaler, 13

  normalise_feature_frame, 13
  normalise_feature_vector, 14
  normalize_feature_frame, 15
  normalize_feature_vector, 16

  plot_all_features, 17
  plot_feature_correlations, 18
  plot_feature_matrix, 20
  plot_low_dimension, 21
  plot_quality_matrix, 23
  plot_ts_correlations, 24
  process_hctsa_file, 25

  robustsigmoid_scaler, 26

  sigmoid_scaler, 26
  simData, 27

  theft, 27

  zscore_scaler, 28