Package ‘theft’

October 6, 2023

**Type** Package

**Title** Tools for Handling Extraction of Features from Time Series

**Version** 0.5.4.1

**Date** 2023-10-05

**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, tibble, fabletools, tsfeatures, feasts, Rcatch22, reticulate, Rtsne, R.matlab, e1071, janitor

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

**RoxygenNote** 7.2.2
VignetteBuilder  knitr

**URL**  https://hendersontrent.github.io/theft/

**NeedsCompilation**  no

**Author**  Trent Henderson [cre, aut],
            Annie Bryant [ctb] (Balanced classification accuracy)

**Repository**  CRAN

**Date/Publication**  2023-10-06 21:10:02 UTC

### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculate_features</td>
<td>3</td>
</tr>
<tr>
<td>calculate_interval</td>
<td>4</td>
</tr>
<tr>
<td>check_vector_quality</td>
<td>5</td>
</tr>
<tr>
<td>compare_features</td>
<td>6</td>
</tr>
<tr>
<td>feature_list</td>
<td>7</td>
</tr>
<tr>
<td>filter_duplicates</td>
<td>8</td>
</tr>
<tr>
<td>filter_good_features</td>
<td>8</td>
</tr>
<tr>
<td>find_good_features</td>
<td>9</td>
</tr>
<tr>
<td>fit_models</td>
<td>9</td>
</tr>
<tr>
<td>get_rescale_vals</td>
<td>10</td>
</tr>
<tr>
<td>init_theft</td>
<td>11</td>
</tr>
<tr>
<td>install_python_pkgs</td>
<td>11</td>
</tr>
<tr>
<td>make_title</td>
<td>12</td>
</tr>
<tr>
<td>maxabs_scaler</td>
<td>12</td>
</tr>
<tr>
<td>minmax_scaler</td>
<td>13</td>
</tr>
<tr>
<td>normalise</td>
<td>13</td>
</tr>
<tr>
<td>plot.feature_calculations</td>
<td>14</td>
</tr>
<tr>
<td>plot.low_dimension</td>
<td>15</td>
</tr>
<tr>
<td>process_hctsa_file</td>
<td>16</td>
</tr>
<tr>
<td>reduce_dims</td>
<td>16</td>
</tr>
<tr>
<td>resampled_ttest</td>
<td>18</td>
</tr>
<tr>
<td>resample_data</td>
<td>19</td>
</tr>
<tr>
<td>rescale_zscore</td>
<td>19</td>
</tr>
<tr>
<td>robustsigmoid_scaler</td>
<td>20</td>
</tr>
<tr>
<td>select_stat_cols</td>
<td>21</td>
</tr>
<tr>
<td>sigmoid_scaler</td>
<td>21</td>
</tr>
<tr>
<td>simData</td>
<td>22</td>
</tr>
<tr>
<td>stat_test</td>
<td>22</td>
</tr>
<tr>
<td>theft</td>
<td>23</td>
</tr>
<tr>
<td>tsfeature_classifier</td>
<td>24</td>
</tr>
<tr>
<td>zscore_scaler</td>
<td>25</td>
</tr>
</tbody>
</table>

**Index**  26
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 = "id",
  time_var = "timepoint",
  values_var = "values",
  group_var = NULL,
  feature_set = c("catch22", "feasts", "tsfeatures", "Kats", "tsfresh", "TSFEL"),
  catch24 = FALSE,
  tsfresh_cleanup = FALSE,
  seed = 123
)
```

**Arguments**

- **data**  
  data.frame with at least 4 columns: id variable, group variable, time variable, value variable
- **id_var**  
  character specifying the ID variable to identify each time series. Defaults to "id"
- **time_var**  
  character specifying the time index variable. Defaults to "timepoint"
- **values_var**  
  character specifying the values variable. Defaults to "values"
- **group_var**  
  character specifying the grouping variable that each unique series sits under (if one exists). Defaults to NULL
- **feature_set**  
  character or vector of character denoting the set of time-series features to calculate. Defaults to "catch22"
- **catch24**  
  Boolean specifying whether to compute catch24 in addition to catch22 if catch22 is one of the feature sets selected. Defaults to FALSE
- **tsfresh_cleanup**  
  Boolean specifying whether to use the in-built tsfresh relevant feature filter or not. Defaults to FALSE
- **seed**  
  integer denoting a fixed number for R’s random number generator to ensure reproducibility. Defaults to 123

**Value**

object of class feature_calculations that contains the summary statistics for each feature
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)

calculate_interval

Calculate interval summaries with a measure of central tendency of classification results

Description

Calculate interval summaries with a measure of central tendency of classification results

Usage

calculate_interval(
data, 
metric = c("accuracy", "precision", "recall", "f1"), 
by_set = TRUE, 
type = c("sd", "qt", "quantile"), 
interval = NULL, 
model_type = c("main", "null")
)

Arguments

data list object containing the classification outputs produce by tsfeature_classifier
metric character denoting the classification performance metric to calculate intervals for. Can be one of "accuracy", "precision", "recall", "f1". Defaults to "accuracy"
by_set Boolean specifying whether to compute intervals for each feature set. Defaults to TRUE. If FALSE, the function will instead calculate intervals for each feature
type character denoting whether to calculate a +/- SD interval with "sd", confidence interval based off the t-distribution with "qt", or based on a quantile with "quantile". Defaults to "sd"
interval numeric scalar denoting the width of the interval to calculate. Defaults to 1 if type = "sd" to produce a +/- 1 SD interval. Defaults to 0.95 if type = "qt" or type = "quantile" for a 95 per cent interval
check_vector_quality

model_type character denoting whether to calculate intervals for main models with "main" or null models with "null" if the use_null argument when using tsfeature_classifier was use_null = TRUE. Defaults to "main"

Value
data.frame containing the results

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)

classifiers <- tsfeature_classifier(featMat,  
by_set = FALSE)

calculate_interval(classifiers,  
by_set = FALSE,  
type = "sd",  
interval = 1)

check_vector_quality  Check for presence of NAs and non-numeric values in a vector

Description
Check for presence of NAs and non-numeric values in a vector

Usage
check_vector_quality(x)

Arguments

x input vector
**Value**

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

**Author(s)**

Trent Henderson

---

**compare_features**

*Conduct statistical testing on time-series feature classification performance to identify top features or compare entire sets*

---

**Description**

Conduct statistical testing on time-series feature classification performance to identify top features or compare entire sets

**Usage**

```r
compare_features(
  data,
  metric = c("accuracy", "precision", "recall", "f1"),
  by_set = TRUE,
  hypothesis = c("null", "pairwise"),
  p_adj = c("none", "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr")
)
```

**Arguments**

- `data` list object containing the classification outputs produce by `tsfeature_classifier`
- `metric` character denoting the classification performance metric to use in statistical testing. Can be one of "accuracy", "precision", "recall", "f1". Defaults to "accuracy"
- `by_set` Boolean specifying whether you want to compare feature sets (if TRUE) or individual features (if FALSE). Defaults to TRUE but this is contingent on whether you computed by set or not in `tsfeature_classifier`
- `hypothesis` character denoting whether p-values should be calculated for each feature set or feature (depending on by_set argument) individually relative to the null if use_null = TRUE in `tsfeature_classifier` through "null", or whether pairwise comparisons between each set or feature should be conducted on main model fits only through "pairwise". Defaults to "null"
- `p_adj` character denoting the adjustment made to p-values for multiple comparisons. Should be a valid argument to `stats::p.adjust`. Defaults to "none" for no adjustment. "holm" is recommended as a starting point for adjustments

**Value**

`data.frame` containing the results
feature_list

Author(s)
  Trent Henderson

References

Examples

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

classifiers <- tsfeature_classifier(featMat,
   by_set = FALSE)

compare_features(classifiers,
   by_set = FALSE,
   hypothesis = "pairwise")
```

feature_list

<table>
<thead>
<tr>
<th>feature_set</th>
<th>Name of the set the feature is from</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature</td>
<td>Name of the feature</td>
</tr>
</tbody>
</table>

Description

The variables include:

Usage

feature_list

Format

A tidy data frame with 2 variables:

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

Remove duplicate features that exist in multiple feature sets and retain a reproducible random selection of one of them

**Description**

Remove duplicate features that exist in multiple feature sets and retain a reproducible random selection of one of them

**Usage**

```r
filter_duplicates(data, preference = NULL, seed = 123)
```

**Arguments**

- `data`: feature_calculations object containing the raw feature matrix produced by `calculate_features`
- `preference`: deprecated. Do not use
- `seed`: integer denoting a fix for R’s pseudo-random number generator to ensure selections are reproducible. Defaults to 123

**Value**

feature_calculations object containing filtered feature data

**Author(s)**

Trent Henderson

---

**filter_good_features**

Filter resample data sets according to good feature list

**Description**

Filter resample data sets according to good feature list

**Usage**

```r
filter_good_features(data, x, good_features)
```

**Arguments**

- `data`: list of "Train" and "Test" data
- `x`: integer denoting the resample index to operate on
- `good_features`: character vector of good features to keep
find_good_features

**Value**

- list of filtered train and test data

**Author(s)**

- Trent Henderson

---

**find_good_features**

*Helper function to find features in both train and test set that are "good"*

---

**Description**

Helper function to find features in both train and test set that are "good"

**Usage**

```
find_good_features(data, x)
```

**Arguments**

- `data` list of "Train" and "Test" data
- `x` integer denoting the resample index to operate on

**Value**

- character vector of "good" feature names

**Author(s)**

- Trent Henderson

---

**fit_models**

*Fit classification model and compute key metrics*

---

**Description**

Fit classification model and compute key metrics

**Usage**

```
fit_models(data, iter_data, row_id, is_null_run = FALSE, classifier)
```
get_rescale_vals

Arguments

- **data**: list containing train and test sets
- **iter_df**: data.frame containing the values to iterate over for seed and either feature name or set name
- **row_id**: integer denoting the row ID for iter_df to filter to
- **is_null_run**: Boolean whether the calculation is for a null model. Defaults to FALSE
- **classifier**: function specifying the classifier to fit. Should be a function with 2 arguments: formula and data. Please note that tsfeature_classifier z-scores data prior to modelling using the train set’s information so disabling default scaling if your function uses it is recommended.

Value

data.frame of classification results

Author(s)

Trent Henderson

Description

Calculate central tendency and spread values for all numeric columns in a dataset

Usage

get_rescale_vals(data)

Arguments

- **data**: data.frame containing data to normalise

Value

list of central tendency and spread values

Author(s)

Trent Henderson
init_theft

Communicate to R the Python virtual environment containing the relevant libraries for calculating features

Description
Communicate to R the Python virtual environment containing the relevant libraries for calculating features

Usage
init_theft(python_path, venv_path)

Arguments
- python_path: character specifying the filepath to the version of Python you wish to use
- venv_path: character specifying the filepath to the Python virtual environment where "tsfresh", "tsfel", and/or "kats" are installed

Value
no return value; called for side effects

Author(s)
Trent Henderson

install_python_pkgs

Download and install all the relevant Python packages into a target location

Description
Download and install all the relevant Python packages into a target location

Usage
install_python_pkgs(python_path, path)

Arguments
- python_path: character specifying the filepath to the location of Python 3.9 on your machine
- path: character denoting the filepath to install the Python libraries and virtual environment to

Author(s)
Trent Henderson
**make_title**  
*Helper function for converting to title case*

**Description**
Helper function for converting to title case

**Usage**

```r
generate_title(x)
```

**Arguments**
- `x` character vector

**Value**
character vector

**Author(s)**
Trent Henderson

---

**maxabs_scaler**  
*Rescales a numeric vector using maximum absolute scaling*

**Description**

\[ z_i = \frac{x_i}{\max(x)} \]

**Usage**

```r
maxabs_scaler(x)
```

**Arguments**
- `x` numeric vector

**Value**
numeric vector

**Author(s)**
Trent Henderson
minmax_scaler

**Description**

\[ z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \]

**Usage**

```r
minmax_scaler(x)
```

**Arguments**

- `x` numeric vector

**Value**

numeric vector

**Author(s)**

Trent Henderson

normalise

**Description**

‘normalise()’ and ‘normalize()’ are synonyms.

**Usage**

```r
normalise( 
  data,  
  norm_method = c("zScore", "Sigmoid", "RobustSigmoid", "MinMax", "MaxAbs"),  
  unit_int = FALSE  
)
```

```r
normalize( 
  data,  
  norm_method = c("zScore", "Sigmoid", "RobustSigmoid", "MinMax", "MaxAbs"),  
  unit_int = FALSE  
)
```
Arguments

data   either a feature_calculations object containing the raw feature matrix produced by calculate_features or a vector of class numeric containing values to be rescaled
norm_method   character denoting the rescaling/normalising method to apply. Can be one of "zScore", "Sigmoid", "RobustSigmoid", "MinMax", or "MaxAbs". Defaults to "zScore"
unit_int   Boolean whether to rescale into unit interval \([0,1]\) after applying normalisation method. Defaults to FALSE

Value

either an object of class data.frame or a numeric vector

Author(s)

Trent Henderson

plot.feature_calculations

Produce a plot for a feature_calculations object

Description

Produce a plot for a feature_calculations object

Usage

## S3 method for class 'feature_calculations'
plot(
x,
type = c("quality", "matrix", "cor", "violin"),
norm_method = c("z-score", "Sigmoid", "RobustSigmoid", "MinMax"),
unit_int = FALSE,
clust_method = c("average", "ward.D", "ward.D2", "single", "complete", "mcquitty", "median", "centroid"),
cor_method = c("pearson", "spearman"),
feature_names = NULL,
...
```
Arguments

x the feature_calculations object containing the raw feature matrix produced by calculate_features

type character specifying the type of plot to draw. Defaults to "quality"

norm_method character specifying a rescaling/normalising method to apply if type = "matrix"
or if type = "cor". Can be one of "z-score", "Sigmoid", "RobustSigmoid", or "MinMax". Defaults to "z-score"

unit_int Boolean whether to rescale into unit interval [0,1] after applying normalisation method. Defaults to FALSE

clust_method character specifying the hierarchical clustering method to use if type = "matrix"
or if type = "cor". Defaults to "average"

cor_method character specifying the correlation method to use if type = "cor". Defaults to "pearson"

feature_names character vector denoting the name of the features to plot if type = "violin". Defaults to NULL

... Arguments to be passed to ggplot2::geom_bar if type = "quality", ggplot2::geom_raster if type = "matrix", ggplot2::geom_raster if type = "cor", or ggplot2::geom_point if type = "violin"

Value

object of class ggplot that contains the graphic

Author(s)

Trent Henderson

plot.low_dimension  Produce a plot for a low_dimension object

Description

Produce a plot for a low_dimension object

Usage

## S3 method for class 'low_dimension'
plot(x, show_covariance = TRUE, ...)

Arguments

x low_dimension object containing the dimensionality reduction projection calculated by reduce_dims

show_covariance Boolean of whether covariance ellipses should be shown on the plot. Defaults to TRUE

... Arguments to be passed to methods
```
Value

object of class ggplot that contains the graphic

Author(s)

Trent Henderson

---

**process_hctsa_file**

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

Description

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

Usage

```
process_hctsa_file(data)
```

Arguments

data string specifying the filepath to the MATLAB file to parse

Value

an object of class data.frame in tidy format

Author(s)

Trent Henderson

---

**reduce_dims**

*Project a feature matrix into a low dimensional representation using PCA or t-SNE*

Description

Project a feature matrix into a low dimensional representation using PCA or t-SNE
reduce_dims

Usage

reduce_dims(
  data,
  norm_method = c("zScore", "Sigmoid", "RobustSigmoid", "MinMax"),
  unit_int = FALSE,
  low_dim_method = c("PCA", "tSNE"),
  na_removal = c("feature", "sample"),
  perplexity = 10,
  seed = 123,
  ...
)

Arguments

data: the feature_calculations object containing the raw feature matrix produced by calculate_features

norm_method: character denoting the rescaling/normalising method to apply. Can be one of "z-score", "Sigmoid", "RobustSigmoid", or "MinMax". Defaults to "z-score"

unit_int: Boolean whether to rescale into unit interval [0,1] after applying normalisation method. Defaults to FALSE

low_dim_method: character specifying the low dimensional embedding method to use. Can be one of "PCA" or "tSNE". Defaults to "PCA"

na_removal: character defining the way to deal with NAs produced during feature calculation. Can be one of "feature" or "sample". "feature" removes all features that produced any NAs in any sample, keeping the number of samples the same. "sample" omits all samples that produced at least one NA. Defaults to "feature"

perplexity: integer denoting the perplexity hyperparameter to use if low_dim_method is "t-SNE". Defaults to 10

seed: integer to fix R's random number generator to ensure reproducibility. Defaults to 123

...: arguments to be passed to either stats::prcomp or Rtsne::Rtsne depending on whether "low_dim_method" is "PCA" or "t-SNE"

Value

object of class low_dimension

Author(s)

Trent Henderson
## resampled_ttest

Compute correlated t-statistic and p-value for resampled data from correctR package

### Description

Compute correlated t-statistic and p-value for resampled data from correctR package

### Usage

```r
resampled_ttest(x, y, n, n1, n2)
```

### Arguments

- **x**: numeric vector of values for model A
- **y**: numeric vector of values for model B
- **n**: integer denoting number of repeat samples. Defaults to `length(x)`
- **n1**: integer denoting train set size
- **n2**: integer denoting test set size

### Value

object of class `data.frame`

### Author(s)

Trent Henderson

### References


**resample_data**  
*Helper function to create a resampled dataset*

**Description**  
Helper function to create a resampled dataset

**Usage**  
```r  
resample_data(data, train_rows, test_rows, train_groups, test_groups, seed)  
```

**Arguments**

- **data**  
data.frame containing time-series features

- **train_rows**  
integer denoting the number of cases in the train set

- **test_rows**  
integer denoting the number of cases in the test set

- **train_groups**  
data.frame containing proportions of each class in original train split

- **test_groups**  
data.frame containing proportions of each class in original test split

- **seed**  
integer denoting fixed value for R’s pseudorandom number generator

**Value**

list containing new train and test data

**Author(s)**

Trent Henderson

**rescale_zscore**  
*Calculate z-score for all columns in a dataset using train set central tendency and spread*

**Description**  
Calculate z-score for all columns in a dataset using train set central tendency and spread

**Usage**

```r  
rescale_zscore(data, rescalers)  
```

**Arguments**

- **data**  
data.frame containing data to normalise

- **rescalers**  
list containing central tendency and spread values for the train set
robustsigmoid_scaler

Rescales a numeric vector using an outlier-robust Sigmoidal transformation

Description

\[ z_i = \left[ 1 + \exp \left( \frac{x_i - \text{median}(x)}{\text{IQR}(x)/1.35} \right) \right]^{-1} \]

Usage

robustsigmoid_scaler(x)

Arguments

x numeric vector

Value

numeric vector

Author(s)

Trent Henderson

References

select_stat_cols

Description

Helper function to select only the relevant columns for statistical testing

Usage

select_stat_cols(data, by_set, metric, hypothesis)

Arguments

- **data**: data.frame of classification accuracy results
- **by_set**: Boolean specifying whether you want to compare feature sets (if TRUE) or individual features (if FALSE).
- **metric**: character denoting the classification performance metric to use in statistical testing. Can be one of "accuracy", "precision", "recall", "f1". Defaults to "accuracy"
- **hypothesis**: character denoting whether p-values should be calculated for each feature set or feature (depending on by_set argument) individually relative to the null if use_null = TRUE in tsfeature_classifier through "null", or whether pairwise comparisons between each set or feature should be conducted on main model fits only through "pairwise".

Value

object of class data.frame

Author(s)

Trent Henderson

sigmoid_scaler

Description

\[ z_i = \left[ 1 + \exp\left( -\frac{x_i - \mu}{\sigma} \right) \right]^{-1} \]

Usage

sigmoid_scaler(x)
Arguments

x numeric vector

Value

numeric vector

Author(s)

Trent Henderson

---

**simData**

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

---

**Description**

The variables include:

**Usage**

```
simData
```

**Format**

A tidy data frame with 4 variables:

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

---

**stat_test**

*Calculate p-values for feature sets or features relative to an empirical null or each other using resampled t-tests*

---

**Description**

Calculate p-values for feature sets or features relative to an empirical null or each other using resampled t-tests
Usage

stat_test(
data,  
it_iter_data,  
row_id,  
by_set = FALSE,  
hypothesis,  
metric,  
train_test_sizes,  
n_resamples
)

Arguments

data data.frame of raw classification accuracy results
iter_data data.frame containing the values to iterate over for seed and either feature name or set name
row_id integer denoting the row ID for iter_data to filter to
by_set Boolean specifying whether you want to compare feature sets (if TRUE) or individual features (if FALSE).
hypothesis character denoting whether p-values should be calculated for each feature set or feature (depending on by_set argument) individually relative to the null if use_null = TRUE in tsfeature_classifier through "null", or whether pairwise comparisons between each set or feature should be conducted on main model fits only through "pairwise".
metric character denoting the classification performance metric to use in statistical testing. Can be one of "accuracy", "precision", "recall", "f1". Defaults to "accuracy"
train_test_sizes integer vector containing the train and test set sample sizes
n_resamples integer denoting the number of resamples that were calculated

Value

object of class data.frame

Author(s)

Trent Henderson

theft

Tools for Handling Extraction of Features from Time-series

Description

Tools for Handling Extraction of Features from Time-series
tsfeature_classifier

Fit classifiers using time-series features using a resample-based approach and get a fast understanding of performance

Description

Fit classifiers using time-series features using a resample-based approach and get a fast understanding of performance

Usage

```r
tsfeature_classifier(
  data,
  classifier = NULL,
  train_size = 0.75,
  n_resamples = 30,
  by_set = TRUE,
  use_null = FALSE,
  seed = 123
)
```

Arguments

- **data** feature_calculations object containing the raw feature matrix produced by calculate_features with an included group column as per theft::calculate_features
- **classifier** function specifying the classifier to fit. Should be a function with 2 arguments: formula and data containing a classifier compatible with R’s predict functionality. Please note that tsfeature_classifier z-scores data prior to modelling using the train set’s information so disabling default scaling if your function uses it is recommended. Defaults to NULL which means the following linear SVM is fit: classifier = function(formula, data){mod <- e1071::svm(formula, data = data, kernel = "linear", scale = FALSE, probability = TRUE)}
- **train_size** numeric denoting the proportion of samples to use in the training set. Defaults to 0.75
- **n_resamples** integer denoting the number of resamples to calculate. Defaults to 30
- **by_set** Boolean specifying whether to compute classifiers for each feature set. Defaults to TRUE. If FALSE, the function will instead find the best individually-performing features
- **use_null** Boolean whether to fit null models where class labels are shuffled in order to generate a null distribution that can be compared to performance on correct class labels. Defaults to FALSE
- **seed** integer to fix R’s random number generator to ensure reproducibility. Defaults to 123
**Value**

list containing a named vector of train-test set sizes, and a data.frame of classification performance 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)

classifiers <- tsfeature_classifier(featMat, 
   by_set = FALSE)
```

---

**zscore_scaler**  
Rescales a numeric vector into z-scores

**Description**

\[ z_i = \frac{x_i - \mu}{\sigma} \]

**Usage**

`zscore_scaler(x)`

**Arguments**

- `x`: numeric vector

**Value**

numeric vector

**Author(s)**

Trent Henderson
Index

* datasets
  feature_list, 7
  simData, 22
  calculate_features, 3
  calculate_interval, 4
  check_vector_quality, 5
  compare_features, 6
  feature_list, 7
  filter_duplicates, 8
  filter_good_features, 8
  find_good_features, 9
  fit_models, 9
  get_rescale_vals, 10
  init_theft, 11
  install_python_pkgs, 11
  make_title, 12
  maxabs_scaler, 12
  minmax_scaler, 13
  normalise, 13
  normalize (normalise), 13
  plot.feature_calculations, 14
  plot.low_dimension, 15
  process_hctsa_file, 16
  reduce_dims, 16
  resample_data, 19
  resampled_ttest, 18
  rescale_zscore, 19
  robustsigmoid_scaler, 20
  select_stat_cols, 21
  sigmoid_scaler, 21
  simData, 22
  stat_test, 22
  theft, 23
  theft-package (theft), 23
  tsfeature_classifier, 24
  zscore_scaler, 25