Package ‘jmotif’

February 14, 2020

Version 1.1.1
Encoding UTF-8
Title Time Series Analysis Toolkit Based on Symbolic Aggregate Discretization, i.e. SAX
Description Implements time series z-normalization, SAX, HOT-SAX, VSM, SAX-VSM, RePair, and RRA algorithms facilitating time series motif (i.e., recurrent pattern), discord (i.e., anomaly), and characteristic pattern discovery along with interpretable time series classification.
URL https://github.com/jMotif/jmotif-R
BugReports https://github.com/jMotif/jmotif-R/issues
Depends R (>= 3.1.0), Rcpp (>= 0.11.1)
Imports stats
Suggests testthat
LinkingTo Rcpp, RcppArmadillo
LazyData true
SystemRequirements C++11
License GPL-2
RoxygenNote 7.0.2
NeedsCompilation yes
Author Pavel Senin [aut, cre]
Maintainer Pavel Senin <seninp@gmail.com>
Repository CRAN
Date/Publication 2020-02-13 23:30:09 UTC

R topics documented:

alphabet_to_cuts .................................................. 2
bags_to_tfidf .......................................................... 3
CBF ................................................................. 4
alphabet_to_cuts

Translates an alphabet size into the array of corresponding SAX cut-lines built using the Normal distribution.

Description

Translates an alphabet size into the array of corresponding SAX cut-lines built using the Normal distribution.

Usage

alphabet_to_cuts(a_size)

Arguments

a_size the alphabet size, a value between 2 and 20 (inclusive).
bags_to_tfidf

Computes a TF-IDF weight vectors for a set of word bags.

Description

Computes a TF-IDF weight vectors for a set of word bags.

Usage

bags_to_tfidf(data)

Arguments

data the list containing the input word bags.

References


Examples

bag1 = data.frame(
    "words" = c("this", "is", "a", "sample"),
    "counts" = c(1, 1, 2, 1),
    stringsAsFactors = FALSE
)

bag2 = data.frame(
    "words" = c("this", "is", "another", "example"),
    "counts" = c(1, 1, 2, 3),
    stringsAsFactors = FALSE
)

ll = list("bag1" = bag1, "bag2" = bag2)
tfidf = bags_to_tfidf(ll)
CBF

A standard UCR Cylinder-Bell-Funnel dataset from http://www.cs.ucr.edu/~eamonn/time_series_data

**Description**
A standard UCR Cylinder-Bell-Funnel dataset from http://www.cs.ucr.edu/~eamonn/time_series_data

**Usage**
CBF

**Format**
A four-elements list containing train and test data along with their labels
- labels_train: the training data labels, correspond to data matrix rows
- data_train: the training data matrix, each row is a time series instance
- labels_test: the test data labels, correspond to data matrix rows
- data_test: the test data matrix, each row is a time series instance

---

**cosine_dist**

Computes the cosine similarity between numeric vectors

**Description**
Computes the cosine similarity between numeric vectors

**Usage**
cosine_dist(m)

**Arguments**
m the data matrix

**Value**
Returns the cosine similarity

**Examples**
a <- c(2, 1, 0, 2, 0, 1, 1)
b <- c(2, 1, 1, 1, 1, 0, 1)
sim <- cosine_dist(rbind(a,b))
**cosine_sim**

*Computes the cosine distance value between a bag of words and a set of TF-IDF weight vectors.*

**Description**

Computes the cosine distance value between a bag of words and a set of TF-IDF weight vectors.

**Usage**

```r
cosine_sim(data)
```

**Arguments**

- `data`: the list containing a word-bag and the TF-IDF object.

**References**


---

**early_abandoned_dist**

*Finds the Euclidean distance between points, if distance is above the threshold, abandons the computation and returns NaN.*

**Description**

Finds the Euclidean distance between points, if distance is above the threshold, abandons the computation and returns NaN.

**Usage**

```r
early_abandoned_dist(seq1, seq2, upper_limit)
```

**Arguments**

- `seq1`: the array 1.
- `seq2`: the array 2.
- `upper_limit`: the max value after reaching which the distance computation stops and the NAN is returned.
ecg0606  A PHYSIONET dataset

Description
A PHYSIONET dataset

Usage
ecg0606

Format
A vector of numeric values

euclidean_dist  Finds the Euclidean distance between points.

Description
Finds the Euclidean distance between points.

Usage
euclidean_dist(seq1, seq2)

Arguments
- seq1  the array 1.
- seq2  the array 2. stops and the NAN is returned.

find_discords_brute_force
Finds a discord using brute force algorithm.

Description
Finds a discord using brute force algorithm.

Usage
find_discords_brute_force(ts, w_size, discsords_num)
find_discords_hotsax

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ts</td>
<td>the input timeseries.</td>
</tr>
<tr>
<td>w_size</td>
<td>the sliding window size.</td>
</tr>
<tr>
<td>discords_num</td>
<td>the number of discords to report.</td>
</tr>
</tbody>
</table>

References

Keogh, E., Lin, J., Fu, A., HOT SAX: Efficiently finding the most unusual time series subsequence. Proceeding ICDM ’05 Proceedings of the Fifth IEEE International Conference on Data Mining

Examples

discords = find_discords_brute_force(ecg0606[1:600], 100, 1)
plot(ecg0606[1:600], type = "l", col = "cornflowerblue", main = "ECG 0606")
lines(x=c(discords[1,2]:(discords[1,2]+100)), y=ecg0606[discords[1,2]:(discords[1,2]+100)], col="red")

find_discords_hotsax  Finds a discord (i.e. time series anomaly) with HOT-SAX. Usually works the best with lower sizes of discretization parameters: PAA and Alphabet.

Description

Finds a discord (i.e. time series anomaly) with HOT-SAX. Usually works the best with lower sizes of discretization parameters: PAA and Alphabet.

Usage

find_discords_hotsax(ts, w_size, paa_size, a_size, n_threshold, discords_num)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ts</td>
<td>the input timeseries.</td>
</tr>
<tr>
<td>w_size</td>
<td>the sliding window size.</td>
</tr>
<tr>
<td>paa_size</td>
<td>the PAA size.</td>
</tr>
<tr>
<td>a_size</td>
<td>the alphabet size.</td>
</tr>
<tr>
<td>n_threshold</td>
<td>the normalization threshold.</td>
</tr>
<tr>
<td>discords_num</td>
<td>the number of discords to report.</td>
</tr>
</tbody>
</table>

References

Keogh, E., Lin, J., Fu, A., HOT SAX: Efficiently finding the most unusual time series subsequence. Proceeding ICDM ’05 Proceedings of the Fifth IEEE International Conference on Data Mining
find_discords_rra

**Examples**

```r
discords = find_discords_hotsax(ecg0606, 100, 3, 3, 0.01, 1)
plot(ecg0606, type = "l", col = "cornflowerblue", main = "ECG 0606")
lines(x=c(discords[1,2]:(discords[1,2]+100)),
     y=ecg0606[discords[1,2]:(discords[1,2]+100)], col="red")
```

**find_discords_rra**  
*Finds a discord with RRA (Rare Rule Anomaly) algorithm. Usually works the best with higher than that for HOT-SAX sizes of discretization parameters (i.e., PAA and Alphabet sizes).*

**Description**

Finds a discord with RRA (Rare Rule Anomaly) algorithm. Usually works the best with higher than that for HOT-SAX sizes of discretization parameters (i.e., PAA and Alphabet sizes).

**Usage**

```r
find_discords_rra(
  series,
  w_size,
  paa_size,
  a_size,
  nr_strategy,
  n_threshold,
  discords_num
)
```

**Arguments**

- `series` the input timeseries.
- `w_size` the sliding window size.
- `paa_size` the PAA size.
- `a_size` the alphabet size.
- `nr_strategy` the numerosity reduction strategy ("none", "exact", "mindist").
- `n_threshold` the normalization threshold.
- `discords_num` the number of discords to report.

**References**

Gun_Point

Examples

discords = find_discords_rra(ecg0606, 100, 4, 4, "none", 0.01, 1)
plot(ecg0606, type = "l", col = "cornflowerblue", main = "ECG 0606")
lines(x=c(discords[1,2]:(discords[1,2]+100)),
y=ecg0606[discords[1,2]:(discords[1,2]+100)], col="red")

Gun_Point

A standard UCR Gun Point dataset from http://www.cs.ucr.edu/~eamonn/time_series_data

Description

A standard UCR Gun Point dataset from http://www.cs.ucr.edu/~eamonn/time_series_data

Usage

Gun_Point

Format

A four-elements list containing train and test data along with their labels

- labels_train: the training data labels, correspond to data matrix rows
- data_train: the training data matrix, each row is a time series instance
- labels_test: the test data labels, correspond to data matrix rows
- data_test: the test data matrix, each row is a time series instance

idx_to_letter

Get the ASCII letter by an index.

Description

Get the ASCII letter by an index.

Usage

idx_to_letter(idx)

Arguments

idx the index.

Examples

# letter 'b'
idx_to_letter(2)
is_equal_mindist

Compares two strings using mindist.

Description

Compares two strings using mindist.

Usage

is_equal_mindist(a, b)

Arguments

a  the string a.
b  the string b.

Examples

is_equal_str("aaa", "bbb")  # true
is_equal_str("aaa", "ccc")  # false

is_equal_str

Compares two strings using natural letter ordering.

Description

Compares two strings using natural letter ordering.

Usage

is_equal_str(a, b)

Arguments

a  the string a.
b  the string b.

Examples

is_equal_str("aaa", "bbb")
is_equal_str("ccc", "ccc")
letters_to_idx

Get an ASCII indexes sequence for a given character array.

Description
Get an ASCII indexes sequence for a given character array.

Usage
letters_to_idx(str)

Arguments
str the character array.

Examples
letters_to_idx(c('a','b','c','a'))

letter_to_idx

Get the index for an ASCII letter.

Description
Get the index for an ASCII letter.

Usage
letter_to_idx(letter)

Arguments
letter the letter.

Examples
# letter 'b' translates to 2
letter_to_idx('b')
manyseries_to_wordbag  Converts a set of time-series into a single bag of words.

Description

Converts a set of time-series into a single bag of words.

Usage

manyseries_to_wordbag(data, w_size, paa_size, a_size, nr_strategy, n_threshold)

Arguments

data  the timeseries data, row-wise.
w_size  the sliding window size.
paa_size  the PAA size.
a_size  the alphabet size.
rn_strategy  the NR strategy.
n_threshold  the normalization threshold.

References


min_dist  Computes the mindist value for two strings

Description

Computes the mindist value for two strings

Usage

min_dist(str1, str2, alphabet_size, compression_ratio = 1)

Arguments

str1  the first string
str2  the second string
alphabet_size  the used alphabet size
compression_ratio  the distance compression ratio
Value

Returns the distance between strings

References


Examples

str1 <- c('a', 'b', 'c')
str2 <- c('c', 'b', 'a')
min_dist(str1, str2, 3)

paa Computes a Piecewise Aggregate Approximation (PAA) for a time series.

Description

Computes a Piecewise Aggregate Approximation (PAA) for a time series.

Usage

paa(ts, paa_num)

Arguments

- ts: a timeseries to compute the PAA for.
- paa_num: the desired PAA size.

References


Examples

x = c(-1, -2, -1, 0, 2, 1, 1, 0)
x_paa3 = paa(x, 3)
# plot(x, type = "l", main = c("8-points time series and its PAA transform into three points", "PAA shown schematically in blue"))
points(x, pch = 16, lwd = 5)
# paa_bounds = c(1, 1+7/3, 1+7/3*2, 8)
abline(v = paa_bounds, lty = 3, lwd = 2, col = "cornflowerblue")
segments(paa_bounds[1:3], x_paa3, paa_bounds[2:4], x_paa3, col = "cornflowerblue", lwd = 2)
points(x = c(1, 1+7/3, 1+7/3*2) + (7/3)/2, y = x_paa3, pch = 15, lwd = 5, col = "cornflowerblue")
sax_by_chunking  

Discretize a time series with SAX using chunking (no sliding window).

Description

Discretize a time series with SAX using chunking (no sliding window).

Usage

sax_by_chunking(ts, paa_size, a_size, n_threshold)

Arguments

ts the input time series.
paa_size  the PAA size.
a_size the alphabet size.
n_threshold  the normalization threshold.

References


sax_distance_matrix  

Generates a SAX MinDist distance matrix (i.e. the "lookup table") for a given alphabet size.

Description

Generates a SAX MinDist distance matrix (i.e. the "lookup table") for a given alphabet size.

Usage

sax_distance_matrix(a_size)

Arguments

a_size  the desired alphabet size (a value between 2 and 20, inclusive)

Value

Returns a distance matrix (for SAX minDist) for a specified alphabet size.
sax_via_window

References


Examples

sax_distance_matrix(5)

---

sax_via_window Discretizes a time series with SAX via sliding window.

Description

Discretizes a time series with SAX via sliding window.

Usage

sax_via_window(ts, w_size, paa_size, a_size, nr_strategy, n_threshold)

Arguments

ts the input timeseries.
w_size the sliding window size.
paa_size the PAA size.
a_size the alphabet size.
nr_strategy the Numerosity Reduction strategy, acceptable values are "exact" and "mindist" – any other value triggers no numerosity reduction.
n_threshold the normalization threshold.

References

### series_to_chars

TRANSFORMS A TIME SERIES INTO THE CHAR ARRAY USING SAX AND THE NORMAL ALPHABET.

**Description**

Transforms a time series into the char array using SAX and the normal alphabet.

**Usage**

```r
series_to_chars(ts, a_size)
```

**Arguments**

- **ts**: the timeseries.
- **a_size**: the alphabet size.

**References**


**Examples**

```r
y = c(-1, -2, -1, 0, 2, 1, 1, 0)
y_paa3 = paa(y, 3)
series_to_chars(y_paa3, 3)
```

### series_to_string

TRANSFORMS A TIME SERIES INTO THE STRING.

**Description**

Transforms a time series into the string.

**Usage**

```r
series_to_string(ts, a_size)
```

**Arguments**

- **ts**: the timeseries.
- **a_size**: the alphabet size.
References


Examples

```r
y = c(-1, -2, -1, 0, 2, 1, 1, 0)
y_paa3 = paa(y, 3)
series_to_string(y_paa3, 3)
```

---

series_to_wordbag

Converts a single time series into a bag of words.

Description

Converts a single time series into a bag of words.

Usage

```r
series_to_wordbag(ts, w_size, paa_size, a_size, nr_strategy, n_threshold)
```

Arguments

- `ts`: the timeseries.
- `w_size`: the sliding window size.
- `paa_size`: the PAA size.
- `a_size`: the alphabet size.
- `nr_strategy`: the NR strategy.
- `n_threshold`: the normalization threshold.

References


**str_to_repair_grammar**  
*Runs the repair on a string.*

**Description**  
Runs the repair on a string.

**Usage**  
```
str_to_repair_grammar(str)
```

**Arguments**  
- `str`  
  the input string.

**References**  

**Examples**  
```
str_to_repair_grammar("abc abc cba cba bac xxx abc abc cba cba bac")
```

---

**subseries**  
*Extracts a subseries.*

**Description**  
Extracts a subseries.

**Usage**  
```
subseries(ts, start, end)
```

**Arguments**  
- `ts`  
  the input timeseries (0-based, left inclusive).
- `start`  
  the interval start.
- `end`  
  the interval end.

**Examples**  
```
y = c(-1, -2, -1, 0, 2, 1, 1, 0)
subseries(y, 0, 3)
```
Z-normalizes a time series by subtracting its mean and dividing by the standard deviation.

Usage

```
znorm(ts, threshold = 0.01)
```

Arguments

- `ts`: the input time series.
- `threshold`: the z-normalization threshold value, if the input time series’ standard deviation will be found less than this value, the procedure will not be applied, so the "under-threshold-noise" would not get amplified.

References


Examples

```
x = seq(0, pi*4, 0.02)
y = sin(x) * 5 + rnorm(length(x))
plot(x, y, type="l", col="blue")
lines(x, znorm(y, 0.01), type="l", col="red")
```
Index

*Topic datasets
  CBF, 4
  ecg0606, 6
  Gun_Point, 9

alphabet_to_cuts, 2

bags_to_tfidf, 3

CBF, 4
cosine_dist, 4
cosine_sim, 5

early_abandoned_dist, 5
ecg0606, 6
euclidean_dist, 6

find_discords_brute_force, 6
find_discords_hotsax, 7
find_discords_rra, 8

Gun_Point, 9

idx_to_letter, 9
is_equal_mindist, 10
is_equal_str, 10

letter_to_idx, 11
letters_to_idx, 11

manyseries_to_wordbag, 12
min_dist, 12

paa, 13

sax_by_chunking, 14
sax_distance_matrix, 14
sax_via_window, 15
series_to_chars, 16
series_to_string, 16
series_to_wordbag, 17
str_to_repair_grammar, 18

subseries, 18
znorm, 19

20