Package ‘jmotif’

February 7, 2018

Version 1.0.3
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
BugReports https://github.com/jMotif/jmotif/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 6.0.1
NeedsCompilation yes
Author Pavel Senin [aut, cre]
Maintainer Pavel Senin <seninp@gmail.com>
Repository CRAN
Date/Publication 2018-02-07 20:05:23 UTC

R topics documented:

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

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

Description
Translations 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).

References
**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**

```r
bags_to_tfidf(data)
```

**Arguments**

- `data`: the list containing the input word bags.

**References**


**Examples**

```r
cat = alphabet_to_cuts(5)
```

```r
data.frame(
  BwordsB = c("this", "is", "a", "sample"),
  BcountsB = c(1L, 1L, 1L),
  stringsAsFactors = FALSE
)
```

```r
data.frame(
  BwordsB = c("this", "is", "another", "example"),
  BcountsB = c(1L, 1L, 2L, 3L),
  stringsAsFactors = FALSE
)
```

```r
ll = list(Bbag1B = bag1, Bbag2B = 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, 1)
b <- c(2, 1, 1, 1, 1, 0, 1, 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.
ecgP6P6  
*A PHYSIONET dataset*

**Description**

A PHYSIONET dataset

**Usage**

```r
ecgP6P6
```

**Format**

A vector of numeric values

---

`euclidean_dist`  
*Finds the Euclidean distance between points.*

**Description**

Finds the Euclidean distance between points.

**Usage**

```r
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**

```r
find_discords_brute_force(ts, w_size, discords_num)
```
find_discords_hotsax

Arguments

- `ts` the input timeseries.
- `w_size` the sliding window size.
- `discords_num` the number of discords to report.

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

```r
discords = find_discords_brute_force(ecg0606[1:600], 100, 1)
plot(ecg0606[1:600], type = "l", col = "cornflowerblue", main = "ECG 0606")
lines(x=seq(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

```r
find_discords_hotsax(ts, w_size, paa_size, a_size, n_threshold, discs_num)
```

Arguments

- `ts` the input timeseries.
- `w_size` the sliding window size.
- `paa_size` the PAA size.
- `a_size` the alphabet size.
- `n_threshold` the normalization threshold.
- `discords_num` the number of discords to report.

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

```r
find_discords_rra(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**


**Examples**

```r
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  \textit{Compares two strings using mindist.}

\textbf{Description}

Compares two strings using mindist.

\textbf{Usage}

\texttt{is\_equal\_mindist(a, b)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{a} \hspace{1cm} the string \texttt{a}.
  \item \texttt{b} \hspace{1cm} the string \texttt{b}.
\end{itemize}

\textbf{Examples}

\begin{itemize}
  \item \texttt{is\_equal\_str("aaa", "bbb")} \# true
  \item \texttt{is\_equal\_str("aaa", "ccc")} \# false
\end{itemize}

\begin{center}
\begin{tabular}{l}
\hline
\texttt{is\_equal\_str} & \textit{Compares two strings using natural letter ordering.} \\
\hline
\end{tabular}
\end{center}

\textbf{Description}

Compares two strings using natural letter ordering.

\textbf{Usage}

\texttt{is\_equal\_str(a, b)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{a} \hspace{1cm} the string \texttt{a}.
  \item \texttt{b} \hspace{1cm} the string \texttt{b}.
\end{itemize}

\textbf{Examples}

\begin{itemize}
  \item \texttt{is\_equal\_str("aaa", "bbb")}
  \item \texttt{is\_equal\_str("ccc", "ccc")}
\end{itemize}
**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**

```python
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**

```python
# 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

```r
x = c(-1, -2, -1, 0, 2, 1, 1, 0)
plot(x, type = "l", main = "8-points time series and its PAA transform into three points")
points(x, pch = 16, lwd = 5)
segments
abline(v = c(1, 1+7/3, 1+7/3 * 2, 8), lty = 3, lwd = 2)
```

---

**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**

```r
sax_distance_matrix(a_size)
```

**Arguments**

- `a_size`  
  the desired alphabet size (a value between 2 and 20, inclusive)
sax\_via\_window

\textbf{Value}

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

\textbf{References}


\textbf{Examples}

\begin{verbatim}
  sax_distance_matrix(5)
\end{verbatim}

\begin{center}
\begin{tabular}{ll}
\hline
\textit{sax\_via\_window} & \textit{Discretizes a time series with SAX via sliding window.} \\
\hline
\end{tabular}
\end{center}

\textbf{Description}

Discretizes a time series with SAX via sliding window.

\textbf{Usage}

\begin{verbatim}
sax\_via\_window(ts, w\_size, paa\_size, a\_size, nr\_strategy, n\_threshold)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \textit{ts} \hspace{1cm} the input timeseries.
  \item \textit{w\_size} \hspace{1cm} the sliding window size.
  \item \textit{paa\_size} \hspace{1cm} the PAA size.
  \item \textit{a\_size} \hspace{1cm} the alphabet size.
  \item \textit{nr\_strategy} \hspace{1cm} the Numerosity Reduction strategy, acceptable values are "exact" and "mindist" – any other value triggers no numerosity reduction.
  \item \textit{n\_threshold} \hspace{1cm} the normalization threshold.
\end{itemize}

\textbf{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

series_to_chars(ts, a_size)

Arguments

ts       the timeseries.
a_size   the alphabet size.

References


Examples

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

series_to_string(ts, a_size)

Arguments

ts       the timeseries.
a_size   the alphabet size.
series_to_wordbag

References

Examples
\[
\begin{align*}
  y &= c(-1, -2, -1, 0, 2, 1, 1, 0) \\
  y_{paa3} &= paa(y, 3) \\
  \text{series_to_string}(y_{paa3}, 3)
\end{align*}
\]

---

**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**

\[
\text{series_to_wordbag}(\text{ts}, \text{w\_size}, \text{paa\_size}, \text{a\_size}, \text{nr\_strategy}, \text{n\_threshold})
\]

**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**

```python
str_to_repair_grammar(str)
```

**Arguments**

- **str**: the input string.

**References**


**Examples**

```python
str_to_repair_grammar("abc abc cba bac xxx abc abc cba cba bac")
```

---

**subseries**  

*Extracts a subseries.*

**Description**

Extracts a subseries.

**Usage**

```python
subseries(ts, start, end)
```

**Arguments**

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

**Examples**

```python
y = c(-1, -2, -1, 0, 2, 1, 1, 0)
subseries(y, 0, 3)
```
Description

Z-normalizes a time series by subtracting its mean and dividing by the standard deviation.

Usage

```r
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

```r
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

znorm, 19