Package 'tsmp'

June 6, 2019

Type Package

Title Time Series with Matrix Profile

Version 0.3.5

Maintainer Francisco Bischoff <fbischoff@med.up.pt>

Description A toolkit implementing the Matrix Profile concept that was created by CS-UCR <http://www.cs.ucr.edu/~eamonn/MatrixProfile.html>.

License GPL-3

URL https://github.com/franzbischoff/tsmp

BugReports https://github.com/franzbischoff/tsmp/issues

Depends R (>= 2.10)

Imports audio, doSNOW, parallel, foreach, progress, magrittr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 6.1.1.9000

NeedsCompilation no

Suggests spelling, testthat, knitr, rmarkdown, gdtools, vdiffr, animation

VignetteBuilder knitr

Author Francisco Bischoff [aut, cre] (<https://orcid.org/0000-0002-5301-8672>), Michael Yeh [res, ccp, ctb] (<https://orcid.org/0000-0002-9807-2963>), Diego Silva [res, ccp, ctb] (<https://orcid.org/0000-0002-5184-9413>), Yan Zhu [res, ccp, ctb] (<https://orcid.org/0000-0002-5952-2108>), Hoang Dau [res, ccp, ctb] (<https://orcid.org/0000-0003-2439-5185>), Michele Linardi [res, ccp, ctb] (<https://orcid.org/0000-0002-3249-2068>)

Repository CRAN

Date/Publication 2019-06-05 22:00:03 UTC

R topics documented:

as.matrixprofile . . . 3
av_apply . . . 4
av_complexity . . . 5
av_hardlimit_artifact . . . 6
av_motion_artifact . . . 7
av_stop_word . . . 8
av_zerocrossing . . . 9
dist_profile . . . 10
fast_movavg . . . 11
fast_movsd . . . 12
find_chains . . . 12
find_discord . . . 13
find_motif . . . 14
floss . . . 15
floss_cac . . . 16
floss_extract . . . 17
fluss . . . 18
fluss_cac . . . 19
fluss_extract . . . 20
fluss_score . . . 21
get_data . . . 22
min_mp_idx . . . 23
mp_fluss_data . . . 23
mp_gait_data . . . 24
mp_meat_data . . . 25
mp_test_data . . . 26
mp_toy_data . . . 27
mstomp_par . . . 27
plot . . . 29
plot_arcs . . . 31
remove_class . . . 32
salient_mds . . . 33
salient_score . . . 34
salient_subsequences . . . 35
scrimp . . . 36
sdts_predict . . . 37
sdts_score . . . 39
sdts_train . . . 40
set_data . . . 41
simple_fast . . . 42
stamp_par . . . 43
stompi_update . . . 44
stomp_par . . . 45
tsmp . . . 47
valmod . . . 49

Index **52**

`as.matrixprofile` *Convert a TSMP object into another if possible*

Description

The base Classes are `MatrixProfile` and `MultiMatrixProfile`, but as other functions are used, classes are pushed behind, since the last output normally is the most significant. If you want, for example, to plot the Matrix Profile from a `Fluss` object, you may use `as.matrixprofile()` to cast it back.

Usage

```
as.matrixprofile(.mp)

as.multimatrixprofile(.mp)

as.valmod(.mp)

as.fluss(.mp)

as.chain(.mp)

as.discord(.mp)

as.motif(.mp)

as.multimotif(.mp)

as.arccount(.mp)

as.salient(.mp)
```

Arguments

`.mp`	a TSMP object.

Value

Returns the object with the new class, if possible.

Functions

- `as.matrixprofile`: Cast an object changed by another function back to `MatrixProfile`.
- `as.multimatrixprofile`: Cast an object changed by another function back to `MultiMatrixProfile`.
- `as.valmod`: Cast an object changed by another function back to `MultiMatrixProfile`.
- `as.fluss`: Cast an object changed by another function back to `Fluss`.

- as.chain: Cast an object changed by another function back to Chain.
- as.discord: Cast an object changed by another function back to Discord.
- as.motif: Cast an object changed by another function back to Motif.
- as.multimotif: Cast an object changed by another function back to MultiMotif.
- as.arccount: Cast an object changed by another function back to ArcCount.
- as.salient: Cast an object changed by another function back to Salient.

Examples

```
w <- 50
data <- mp_gait_data
mp <- tsmp(data, window_size = w, exclusion_zone = 1 / 4, verbose = 0)
mp <- find_motif(mp)
class(mp) # first class will be "Motif"

plot(mp) # plots a motif plot

plot(as.matrixprofile(mp)) # plots a matrix profile plot
```

---

av_apply *Corrects the matrix profile using an annotation vector*

Description

This function overwrites the current Matrix Profile using the Annotation Vector. Use with caution.

Usage

```
av_apply(.mp)
```

Arguments

.mp	A Matrix Profile with an Annotation Vector.

Value

Returns the input .mp object corrected by the embedded annotation vector.

References

- Dau HA, Keogh E. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17. New York, New York, USA: ACM Press; 2017. p. 125–34.

See Also

Other Annotation vectors: av_complexity, av_hardlimit_artifact, av_motion_artifact, av_stop_word, av_zerocrossing

Examples

```
data <- mp_test_data$train$data[1:1000]
w <- 50
mp <- tsmp(data, window_size = w, verbose = 0)
mp <- av_complexity(mp)
av <- av_apply(mp)
```

---

av_complexity *Computes the annotation vector that favors complexity*

Description

Computes the annotation vector that favors complexity

Usage

```
av_complexity(.mp, data, dilution_factor = 0, apply = FALSE)
```

Arguments

.mp	a Matrix Profile object.
data	a vector or a column matrix of numeric.
dilution_factor	a numeric. (Default is 0). Larger numbers means more dilution.
apply	logical. (Default is FALSE). Applies the Annotation Vector over the Matrix Profile. Use with caution.

Value

Returns the input .mp object with an embedded annotation vector.

References

- Dau HA, Keogh E. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17. New York, New York, USA: ACM Press; 2017. p. 125–34.

See Also

Other Annotation vectors: av_apply, av_hardlimit_artifact, av_motion_artifact, av_stop_word, av_zerocrossing

Examples

```
data <- mp_test_data$train$data[1:1000]
w <- 50
mp <- tsmp(data, window_size = w, verbose = 0)
av <- av_complexity(mp, apply = TRUE)
```

`av_hardlimit_artifact` *Computes the annotation vector that suppresses hard-limited artifacts*

Description

Computes the annotation vector that suppresses hard-limited artifacts

Usage

```
av_hardlimit_artifact(.mp, data, apply = FALSE)
```

Arguments

`.mp`	a Matrix Profile object.
`data`	a `vector` or a column `matrix` of `numeric`.
`apply`	logical. (Default is `FALSE`). Applies the Annotation Vector over the Matrix Profile. Use with caution.

Value

Returns the input `.mp` object with an embedded annotation vector.

References

- Dau HA, Keogh E. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17. New York, New York, USA: ACM Press; 2017. p. 125–34.

See Also

Other Annotation vectors: `av_apply`, `av_complexity`, `av_motion_artifact`, `av_stop_word`, `av_zerocrossing`

Examples

```
data <- mp_test_data$train$data[1:1000]
w <- 50
mp <- tsmp(data, window_size = w, verbose = 0)
av <- av_hardlimit_artifact(mp, apply = TRUE)
```

av_motion_artifact *Computes the annotation vector that suppresses motion artifacts*

Description

Computes the annotation vector that suppresses motion artifacts

Usage

```
av_motion_artifact(.mp, data, apply = FALSE)
```

Arguments

.mp	a Matrix Profile object.
data	a vector or a column matrix of numeric.
apply	logical. (Default is FALSE). Applies the Annotation Vector over the Matrix Profile. Use with caution.

Value

Returns the input .mp object with an embedded annotation vector.

References

- Dau HA, Keogh E. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17. New York, New York, USA: ACM Press; 2017. p. 125–34.

See Also

Other Annotation vectors: av_apply, av_complexity, av_hardlimit_artifact, av_stop_word, av_zerocrossing

Examples

```
data <- mp_test_data$train$data[1:1000]
w <- 50
mp <- tsmp(data, window_size = w, verbose = 0)
av <- av_motion_artifact(mp, apply = TRUE)
```

av_stop_word *Computes the annotation vector that suppresses stop-word motifs*

Description

Computes the annotation vector that suppresses stop-word motifs

Usage

```
av_stop_word(.mp, data, stop_word_loc, exclusion_zone = NULL,
  threshold = 0.1, apply = FALSE)
```

Arguments

.mp	a Matrix Profile object.
data	a vector or a column matrix of numeric.
stop_word_loc	an int. The index of stop word location.
exclusion_zone	a numeric. Size of the exclusion zone, based on window_size (default is NULL). See details.
threshold	a numeric. (default is 0.1).
apply	logical. (Default is FALSE). Applies the Annotation Vector over the Matrix Profile. Use with caution.

Details

The function is intended to be generic. However, its parameters (stop_word_loc, exclusion_zone and threshold) are highly dataset dependent.

Value

Returns the input .mp object with an embedded annotation vector.

References

- Dau HA, Keogh E. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17. New York, New York, USA: ACM Press; 2017. p. 125–34.

See Also

Other Annotation vectors: av_apply, av_complexity, av_hardlimit_artifact, av_motion_artifact, av_zerocrossing

Examples

```
data <- mp_test_data$train$data[1:1000]
w <- 50
mp <- tsmp(data, window_size = w, verbose = 0)
av <- av_stop_word(mp, stop_word_loc = 150, apply = TRUE)
```

---

av_zerocrossing *Computes the annotation vector that favors number of zero crossing*

Description

Computes the annotation vector that favors number of zero crossing

Usage

```
av_zerocrossing(.mp, data, apply = FALSE)
```

Arguments

.mp	a Matrix Profile object.
data	a vector or a column matrix of numeric.
apply	logical. (Default is FALSE). Applies the Annotation Vector over the Matrix Profile. Use with caution.

Value

Returns the input .mp object with an embedded annotation vector.

References

- Dau HA, Keogh E. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17. New York, New York, USA: ACM Press; 2017. p. 125–34.

See Also

Other Annotation vectors: av_apply, av_complexity, av_hardlimit_artifact, av_motion_artifact, av_stop_word

Examples

```
data <- mp_test_data$train$data[1:1000]
w <- 50
mp <- tsmp(data, window_size = w, verbose = 0)
av <- av_zerocrossing(mp, apply = TRUE)
```

`dist_profile` *Calculates the distance profile using MASS algorithms*

Description

Mueen's Algorithm for Similarity Search is The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance and Correlation Coefficient.

Usage

```
dist_profile(data, query, ..., window_size = NULL, method = "v3",
  index = 1, k = NULL, weight = NULL, paa = 1)
```

Arguments

`data`	a `matrix` or a `vector`.
`query`	a `matrix` or a `vector`. See details.
`...`	Precomputed values from the first iteration. If not supplied, these values will be computed.
`window_size`	an `int` or `NULL`. Sliding window size. See details.
`method`	method that will be used to calculate the distance profile. See details.
`index`	an `int`. Index of query window. See details.
`k`	an `int` or `NULL`. Default is `NULL`. Defines the size of batch for MASS V3. Prefer to use a power of 2. If `NULL`, it will be set automatically.
`weight`	a `vector` of `numeric` or `NULL` with the same length of the `window_size`. This is a MASS extension to weight the query.
`paa`	a `numeric`. Default is `1`. Factor of PAA reduction (2 == half of size). This is a MASS extension.

Details

This function has several ways to work:

Case 1: You have a small sized query and the data. In this case you only have to provide the first two parameters `data` and `query`. Internally the `window_size` will be get from the query length.

Case 2: You have one or two data vectors and want to compute the join or self-similarity. In this case you need to use the recursive solution. The parameters are `data`, `query`, `window_size` and `index`. The first iteration don't need the `index` unless you are starting somewhere else. The `query` will be the source of a `query_window`, starting on `index`, with length of `window_size`.

The `method` defines which MASS will be used. Current supported values are: `v2`, `v3`, `weighted`.

Value

Returns the `distance_profile` for the given query and the `last_product` for STOMP algorithm and the parameters for recursive call. See details.

References

- Abdullah Mueen, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Kumar Gupta and Eamonn Keogh (2015), The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance

Website: https://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html

Examples

```
w <- mp_toy_data$sub_len
ref_data <- mp_toy_data$data[, 1]
# minimum example, data and query
nn <- dist_profile(ref_data, ref_data[1:w])
distance_profile <- Re(sqrt(nn$distance_profile))

# data and indexed query
nn <- dist_profile(ref_data, ref_data, window_size = w, index = 10)
distance_profile <- Re(sqrt(nn$distance_profile))

# recursive
nn <- NULL

for (i in seq_len(10)) {
  nn <- dist_profile(ref_data, ref_data, nn, window_size = w, index = i)
}

# weighted
weight <- c(rep(1, w / 3), rep(0.5, w / 3), rep(0.8, w / 3)) # just an example

nn <- dist_profile(ref_data, ref_data, window_size = w, index = 1, method = "weighted",
 weight = weight)
distance_profile <- Re(sqrt(nn$distance_profile))
```

`fast_movavg` *Fast implementation of moving average and*

Description

Fast implementation of moving average and

Usage

```
fast_movavg(data, window_size)
```

Arguments

`data`	a `vector` or a column `matrix` of `numeric`.
`window_size`	moving sd window size

Value

Returns a vector with the moving average

Examples

```
data_avg <- fast_movavg(mp_toy_data$data[, 1], mp_toy_data$sub_len)
```

---

fast_movsd *Fast implementation of moving standard deviation using filter*

Description

Fast implementation of moving standard deviation using filter

Usage

```
fast_movsd(data, window_size)
```

Arguments

data	a vector or a column matrix of numeric.
window_size	moving sd window size

Value

Returns a vector with the moving standard deviation

Examples

```
data_sd <- fast_movsd(mp_toy_data$data[, 1], mp_toy_data$sub_len)
```

---

find_chains *Find Time Series Chains*

Description

Time Series Chains is a new primitive for time series data mining.

Usage

```
find_chains(.mp)
```

Arguments

.mp	a TSMP object of class MatrixProfile.

Value

Returns the input `.mp` object with a new name `chain`. It contains: `chains`, a `list` of chains founded with more than 2 patterns and `best` with the best one.

References

- Zhu Y, Imamura M, Nikovski D, Keogh E. Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining. Knowl Inf Syst. 2018 Jun 2;1–27.

Website: https://sites.google.com/site/timeserieschain/

Examples

```
w <- 50
data <- mp_gait_data
mp <- tsmp(data, window_size = w, exclusion_zone = 1 / 4, verbose = 0)
mp <- find_chains(mp)
```

---

find_discord *Search for Discord*

Description

Search for Discord

Usage

```
find_discord(.mp, ...)

## S3 method for class 'MatrixProfile'
find_discord(.mp, data, n_discords = 1,
  n_neighbors = 3, radius = 3, exclusion_zone = NULL, ...)
```

Arguments

`.mp`	a TSMP object of class `MatrixProfile`
`...`	further arguments to be passed to class specific function.
`data`	the data used to build the Matrix Profile, if not embedded.
`n_discords`	an `int`. Number of discords to find. (Default is `1`).
`n_neighbors`	an `int`. Number of neighbors to find. (Default is `3`).
`radius`	an `int`. Set a threshold to exclude matching neighbors with distance > current discord distance * `radius`. (Default is `3`).
`exclusion_zone`	if a `number` will be used instead of embedded value. (Default is `NULL`).

Value

For class `MatrixProfile`, returns the input `.mp` object with a new name `discord`. It contains: `discord_idx`, a `vector` of discords founded.

Examples

```
# Single dimension data
w <- 50
data <- mp_gait_data
mp <- tsmp(data, window_size = w, exclusion_zone = 1 / 4, verbose = 0)
mp <- find_discord(mp)
```

`find_motif` *Search for Motifs*

Description

Search for Motifs

Usage

```
find_motif(.mp, ...)

## S3 method for class 'MatrixProfile'
find_motif(.mp, data, n_motifs = 3,
  n_neighbors = 10, radius = 3, exclusion_zone = NULL, ...)

## S3 method for class 'MultiMatrixProfile'
find_motif(.mp, data, n_motifs = 3,
  mode = c("guided", "unconstrained"), n_bit = 4,
  exclusion_zone = NULL, n_dim = NULL, ...)
```

Arguments

`.mp`	a TSMP object of class `MatrixProfile` or `MultiMatrixProfile`.
`...`	further arguments to be passed to class specific function.
`data`	the data used to build the Matrix Profile, if not embedded.
`n_motifs`	an `int`. Number of motifs to find. (Default is `3`).
`n_neighbors`	an `int`. Number of neighbors to find. (Default is `10`).
`radius`	an `int`. Set a threshold to exclude matching neighbors with distance > current motif distance * `radius`. (Default is `3`).
`exclusion_zone`	if a `number` will be used instead of embedded value. (Default is `NULL`).
`mode`	a `string`. Guided or Unconstrained search. Allow partial match. (Default is `guided`).
`n_bit`	an `int`. Bit size for discretization. Ignored on Guided search. (Default is `4`).
`n_dim`	an `int`. Number of dimensions to use on Guided search instead of embedded value. (Default is `NULL`).

Value

For class MatrixProfile, returns the input .mp object with a new name motif. It contains: motif_idx, a list of motif pairs founded and motif_neighbor a list with respective motif's neighbors.

For class MultiMatrixProfile, returns the input .mp object with a new name motif. It contains: motif_idx, a vector of motifs founded and motif_dim a list the dimensions where the motifs were founded.

Examples

```
# Single dimension data
w <- 50
data <- mp_gait_data
mp <- tsmp(data, window_size = w, exclusion_zone = 1 / 4, verbose = 0)
mp <- find_motif(mp)

# Multidimension data
w <- mp_toy_data$sub_len
data <- mp_toy_data$data[1:200, ]
mp <- tsmp(data, window_size = w, mode = "mstomp", verbose = 0)
mp <- find_motif(mp)
```

---

floss *Fast Low-cost Online Semantic Segmentation (FLOSS)*

Description

Fast Low-cost Online Semantic Segmentation (FLOSS)

Usage

```
floss(.mp, new_data, data_window, threshold = 1, exclusion_zone = NULL,
  chunk_size = NULL, keep_cac = TRUE)
```

Arguments

.mp	.mp a TSMP object of class MatrixProfile.
new_data	a matrixor vector of new observations.
data_window	an int. Sets the size of the buffer used to keep track of semantic changes.
threshold	a number. (Default is 1). Set the maximum value for evaluating semantic changes. This is data specific. It is advised to check what is 'normal' for your data.
exclusion_zone	if a number will be used instead of embedded value. (Default is NULL).
chunk_size	an int . (Default is NULL). Set the size of new data that will be added to Floss in each iteration if new_data is large. If NULL, the size will be 50. This is not needed if new_data is small, like 1 observation.
keep_cac	a logical. (Default is TRUE). If set to FALSE, the cac_final will contain only values within data_window

Value

Returns the input `.mp` object new names: `cac` the corrected arc count, `cac_final`the combination of `cac` after repeated calls of `floss()`, `floss` with the location of semantic changes and `floss_vals` with the normalized arc count value of the semantic change positions.

References

- Gharghabi S, Ding Y, Yeh C-CM, Kamgar K, Ulanova L, Keogh E. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE; 2017. p. 117–26.

Website: https://sites.google.com/site/onlinesemanticsegmentation/

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other Semantic Segmentations: `floss_cac`, `floss_extract`, `fluss_cac`, `fluss_extract`, `fluss_score`, `fluss`

Examples

```
data <- mp_fluss_data$tilt_abp$data[1:1000]
new_data <- mp_fluss_data$tilt_abp$data[1001:1010]
new_data2 <- mp_fluss_data$tilt_abp$data[1011:1020]
w <- 80
mp <- tsmp(data, window_size = w, verbose = 0)
data_window <- 1000
mp <- floss(mp, new_data, data_window)
mp <- floss(mp, new_data2, data_window)
```

---

floss_cac *FLOSS - Corrected Arc Counts*

Description

Computes the arc count with edge and 'online' correction (CAC).

Usage

```
floss_cac(.mp, data_window, exclusion_zone = NULL)
```

Arguments

`.mp`	a TSMP object of class `MatrixProfile`.
`data_window`	an `int`. Sets the size of the buffer used to keep track of semantic changes.
`exclusion_zone`	if a `number` will be used instead of embedded value. (Default is `NULL`).

Details

Original paper suggest using the classic statistical-process-control heuristic to set a threshold where a semantic change may occur in CAC. This may be useful in real-time implementation as we don't know in advance the number of domain changes to look for. Please check original paper (1).

Value

Returns the input `.mp` object a new name `cac` with the corrected arc count and `cac_final` the combination of `cac` after repeated calls of `floss()`.

References

- Gharghabi S, Ding Y, Yeh C-CM, Kamgar K, Ulanova L, Keogh E. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE; 2017. p. 117–26.

Website: https://sites.google.com/site/onlinesemanticsegmentation/

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other Semantic Segmentations: `floss_extract`, `floss`, `fluss_cac`, `fluss_extract`, `fluss_score`, `fluss`

Examples

```
data <- mp_fluss_data$tilt_abp$data[1:1000]
new_data <- mp_fluss_data$tilt_abp$data[1001:1010]
w <- 10
mp <- tsmp(data, window_size = w, verbose = 0)
data_window <- 1000
mp <- stompi_update(mp, new_data, data_window)
mp <- floss_cac(mp, data_window)
```

`floss_extract` *FLOSS - Extract Segments*

Description

Extract candidate points of semantic changes.

Usage

```
floss_extract(.mpac, threshold = 1, exclusion_zone = NULL)
```

Arguments

.mpac	a TSMP object of class ArcCount.
threshold	a number. (Default is 1). Set the maximum value for evaluating semantic changes. This is data specific. It is advised to check what is 'normal' for your data.
exclusion_zone	if a number will be used instead of embedded value. (Default is NULL).

Value

Returns the input .mp object a new name floss with the location of semantic changes and floss_vals with the normalized arc count value of the semantic change positions.

References

- Gharghabi S, Ding Y, Yeh C-CM, Kamgar K, Ulanova L, Keogh E. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE; 2017. p. 117–26.

Website: https://sites.google.com/site/onlinesemanticsegmentation/

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other Semantic Segmentations: floss_cac, floss, fluss_cac, fluss_extract, fluss_score, fluss

Examples

```
data <- mp_fluss_data$tilt_abp$data[1:1000]
w <- 10
mp <- tsmp(data, window_size = w, verbose = 0)
mp <- fluss_cac(mp)
mp <- fluss_extract(mp, 2)
```

---

fluss *Fast Low-cost Unipotent Semantic Segmentation (FLUSS)*

Description

FLUSS is a Domain Agnostic Online Semantic Segmentation that uses the assumption that when few arc are crossing a given index point, means that there is a high probability of semantic change. This function is a wrap to fluss_cac() and fluss_extract().

Usage

```
fluss(.mp, num_segments = 1, exclusion_zone = NULL)
```

Arguments

.mp	a TSMP object of class MatrixProfile.
num_segments	an int. Number of segments to extract. Based on domain knowledge.
exclusion_zone	if a number will be used instead of embedded value. (Default is NULL).

Value

Returns the input .mp object new names: cac, corrected arc count and fluss with the location of semantic changes.

References

- Gharghabi S, Ding Y, Yeh C-CM, Kamgar K, Ulanova L, Keogh E. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE; 2017. p. 117–26.

Website: https://sites.google.com/site/onlinesemanticsegmentation/

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other Semantic Segmentations: floss_cac, floss_extract, floss, fluss_cac, fluss_extract, fluss_score

Examples

```
data <- mp_fluss_data$tilt_abp$data[1:1000]
w <- 10
mp <- tsmp(data, window_size = w, verbose = 0)
mp <- fluss(mp, 2)
```

---

fluss_cac *FLUSS - Corrected Arc Counts*

Description

Computes the arc count with edge correction (CAC).

Usage

```
fluss_cac(.mp, exclusion_zone = NULL)
```

Arguments

.mp	a TSMP object of class MatrixProfile.
exclusion_zone	if a number will be used instead of embedded value. (Default is NULL).

Details

Original paper suggest using the classic statistical-process-control heuristic to set a threshold where a semantic change may occur in CAC. This may be useful in real-time implementation as we don't know in advance the number of domain changes to look for. Please check original paper (1).

Value

Returns the input `.mp` object a new name `cac` with the corrected arc count.

References

- Gharghabi S, Ding Y, Yeh C-CM, Kamgar K, Ulanova L, Keogh E. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE; 2017. p. 117–26.

Website: https://sites.google.com/site/onlinesemanticsegmentation/

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other Semantic Segmentations: `floss_cac`, `floss_extract`, `floss`, `fluss_extract`, `fluss_score`, `fluss`

Examples

```
data <- mp_fluss_data$tilt_abp$data[1:1000]
w <- 10
mp <- tsmp(data, window_size = w, verbose = 0)
mp <- fluss_cac(mp)
```

`fluss_extract` *FLUSS - Extract Segments*

Description

Extract candidate points of semantic changes.

Usage

```
fluss_extract(.mpac, num_segments = 1, exclusion_zone = NULL)
```

Arguments

`.mpac`	a TSMP object of class `ArcCount`.
`num_segments`	an `int`. Number of segments to extract. Based on domain knowledge.
`exclusion_zone`	if a `number` will be used instead of embedded value. (Default is `NULL`).

Value

Returns the input `.mp` object a new name `fluss` with the location of semantic changes.

References

- Gharghabi S, Ding Y, Yeh C-CM, Kamgar K, Ulanova L, Keogh E. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE; 2017. p. 117–26.

Website: https://sites.google.com/site/onlinesemanticsegmentation/

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other Semantic Segmentations: `floss_cac`, `floss_extract`, `floss`, `fluss_cac`, `fluss_score`, `fluss`

Examples

```
data <- mp_fluss_data$tilt_abp$data[1:1000]
w <- 10
mp <- tsmp(data, window_size = w, verbose = 0)
mp <- fluss_cac(mp)
mp <- fluss_extract(mp, 2)
```

`fluss_score` *FLUSS - Prediction score calculation*

Description

FLUSS - Prediction score calculation

Usage

```
fluss_score(gtruth, extracted, data_size)
```

Arguments

`gtruth`	an `int` or `vector` of `int` with the ground truth index of segments.
`extracted`	an `int` or `vector` of `int` with the extracted indexes from `fluss_extract()`.
`data_size`	an `int`. Size of original input data.

Value

Returns the score of predicted semantic transitions compared with the ground truth. Zero is the best, One is the worst.

References

- Gharghabi S, Ding Y, Yeh C-CM, Kamgar K, Ulanova L, Keogh E. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE; 2017. p. 117–26.

Website: https://sites.google.com/site/onlinesemanticsegmentation/

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other Semantic Segmentations: floss_cac, floss_extract, floss, fluss_cac, fluss_extract, fluss

Examples

```
data <- mp_fluss_data$tilt_abp$data[1:1000]
w <- 10
truth <- c(945, 875)
mp <- tsmp(data, window_size = w, verbose = 0)
mp <- fluss_cac(mp)
mp <- fluss_extract(mp, 2)
score <- fluss_score(truth, mp$fluss, length(data))
```

---

get_data *Get the data included in a TSMP object, if any.*

Description

Get the data included in a TSMP object, if any.

Usage

```
get_data(.mp)
```

Arguments

.mp	a TSMP object.

Value

Returns the data as matrix. If there is more than one series, returns a list.

Examples

```
mp <- tsmp(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)
get_data(mp)
```

min_mp_idx *Get index of the minimum value from a matrix profile and its nearest neighbor*

Description

Get index of the minimum value from a matrix profile and its nearest neighbor

Usage

```
min_mp_idx(.mp, n_dim = NULL, valid = TRUE)
```

Arguments

.mp	a TSMP object of class MatrixProfile.
n_dim	number of dimensions of the matrix profile
valid	check for valid numbers

Value

returns the minimum and the nearest neighbor

Examples

```
w <- 50
data <- mp_gait_data
mp <- tsmp(data, window_size = w, exclusion_zone = 1 / 4, verbose = 0)
min_val <- min_mp_idx(mp)
```

mp_fluss_data *Original data used in the FLUSS paper*

Description

Contains two datasets used in FLUSS paper (1), first is TiltABP from (2), and second is WalkJogRun from PAMAP's dataset (3)

Usage

```
mp_fluss_data
```

Format

A list containing:

data one column matrix with the dataset's data

gtruth a vector with the ground truth of semantic change according to provided dataset

window window size used in original paper

Source

https://sites.google.com/site/onlinesemanticsegmentation/

http://www.cs.ucr.edu/~eamonn/time_series_data/

References

- Gharghabi S, Ding Y, Yeh C-CM, Kamgar K, Ulanova L, Keogh E. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE; 2017. p. 117–26.
- Heldt, T., Oefinger, M.B., Hoshiyama, M. and Mark, R.G., 2003, September. Circulatory response to passive and active changes in posture. In IEEE Computers in Cardiology, 2003 (pp. 263-266).
- Reiss, A. and Stricker, D., 2012. Introducing a new benchmarked dataset for activity monitoring. In 16th International Symposium on Wearable Computers (ISWC), 2012, pages 108–109. IEEE, 2012.

---

mp_gait_data *Original data used in the Time Series Chain demo*

Description

Original data used in the Time Series Chain demo

Usage

```
mp_gait_data
```

Format

A `matrix` with 904 rows and 1 column with the Y data from an accelerometer

Source

https://sites.google.com/site/timeserieschain/

References

- Zhu Y, Imamura M, Nikovski D, Keogh E. Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining. Knowl Inf Syst. 2018 Jun 2;1–27.

`mp_meat_data` *Original data used in the Salient Subsequences demo*

Description

This is the Meat dataset from UCR Archive modified for Salient discovery. The original data is mixed with Random Walks and the algorithm must pick only the originals.

Usage

```
mp_meat_data
```

Format

`original` is the original dataset with 60+60 observations mixed with 120 random walks:

data 240 time series with length of 448 each.

labels label of each time series, `-666` means a random walk.

sub_len size of sliding window.

`sub` is the original dataset embedded in random walks:

data One time series with length of 107520.

labels label of each original data.

labels_idx starting point where the original data was placed.

sub_len size of sliding window.

Source

http://www.cs.ucr.edu/~eamonn/time_series_data/

References

- Yeh CCM, Van Herle H, Keogh E. Matrix profile III: The matrix profile allows visualization of salient subsequences in massive time series. Proc - IEEE Int Conf Data Mining, ICDM. 2017;579–88.

- Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E. Discovering the Intrinsic Cardinality and Dimensionality of Time Series Using MDL. In: 2011 IEEE 11th International Conference on Data Mining. IEEE; 2011. p. 1086–91.

Website: https://sites.google.com/site/salientsubs/

`mp_test_data` *Original data used in the STDS demo*

Description

A synthetic dataset base on TRACE dataset and used as Stress Test to STDS algorithm. The TRACE dataset used here is originally from (1), and the version distributed here is from (2)

Usage

```
mp_test_data
```

Format

A list of matrices with 215010 rows and 1 dimension:

train$data training data

train$label label for training data

test$data test data

test$label label for test data

Source

https://sites.google.com/view/weaklylabeled

http://www.cs.ucr.edu/~eamonn/time_series_data/

References

- Roverso, D., Multivariate temporal classification by windowed wavelet decomposition and recurrent neural networks, in 3rd ANS Int'l Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface, vol. 20, Washington, DC, USA, 2000.

- Yeh C-CM, Kavantzas N, Keogh E. Matrix profile IV: Using Weakly Labeled Time Series to Predict Outcomes. Proc VLDB Endow. 2017 Aug 1;10(12):1802–12.

mp_toy_data *Original data used in the mSTAMP demo*

Description

A synthetic dataset with embedded MOTIFs for multidimensional discovery

Usage

```
mp_toy_data
```

Format

A `list` with a `matrix` with 550 rows and 3 dimensions and an `int`:

data data with embedded MOTIFs

sub_len size of sliding window

Source

https://sites.google.com/view/mstamp/

References

- Yeh CM, Kavantzas N, Keogh E. Matrix Profile VI : Meaningful Multidimensional Motif Discovery.

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

mstomp_par *Multivariate STOMP algorithm Parallel version*

Description

Computes the Matrix Profile and Profile Index for Multivariate Time Series.

Usage

```
mstomp_par(data, window_size, exclusion_zone = 1/2, verbose = 2,
  must_dim = NULL, exc_dim = NULL, n_workers = 2)

mstomp(data, window_size, exclusion_zone = 1/2, verbose = 2,
  must_dim = NULL, exc_dim = NULL)
```

Arguments

data	a matrix of numeric, where each column is a time series. Accepts vector (see details), list and data.frame too.
window_size	an int with the size of the sliding window.
exclusion_zone	a numeric. Size of the exclusion zone, based on window size (default is 1/2).
verbose	an int. See details. (Default is 2).
must_dim	an int or vector of which dimensions to forcibly include (default is NULL).
exc_dim	an int or vector of which dimensions to exclude (default is NULL).
n_workers	an int. Number of workers for parallel. (Default is 2).

Details

The Matrix Profile, has the potential to revolutionize time series data mining because of its generality, versatility, simplicity and scalability. In particular it has implications for time series motif discovery, time series joins, shapelet discovery (classification), density estimation, semantic segmentation, visualization, rule discovery, clustering etc. The MSTOMP computes the Matrix Profile and Profile Index for Multivariate Time Series that is meaningful for multidimensional MOTIF discovery. It uses the STOMP algorithm that is faster than STAMP but lacks its anytime property.

Although this functions handles Multivariate Time Series, it can also be used to handle Univariate Time Series. verbose changes how much information is printed by this function; 0 means nothing, 1 means text, 2 adds the progress bar, 3 adds the finish sound.

Value

Returns a MultiMatrixProfile object, a list with the matrix profile mp, profile index pi left and right matrix profile lmp, rmp and profile index lpi, rpi, window size w, number of dimensions n_dim, exclusion zone ez, must dimensions must and excluded dimensions exc.

If the input has only one dimension, returns the same as stomp().

Functions

- mstomp_par: Parallel version.
- mstomp: Single thread version.

References

- Yeh CM, Kavantzas N, Keogh E. Matrix Profile VI : Meaningful Multidimensional Motif Discovery.
- Zhu Y, Imamura M, Nikovski D, Keogh E. Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining. Knowl Inf Syst. 2018 Jun 2;1–27.

Website: https://sites.google.com/view/mstamp/

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other matrix profile computations: scrimp, stamp_par, stomp_par, tsmp, valmod

Examples

```
# using all dimensions
mp <- mstomp(mp_toy_data$data[1:150, ], 30, verbose = 0)

# using threads
mp <- mstomp_par(mp_toy_data$data[1:150, ], 30, verbose = 0)
## Not run:
# force using dimensions 1 and 2
mp <- mstomp(mp_toy_data$data[1:200, ], 30, must_dim = c(1, 2))
# exclude dimensions 2 and 3
mp2 <- mstomp(mp_toy_data$data[1:200, ], 30, exc_dim = c(2, 3))

## End(Not run)
```

---

plot *Plot a TSMP object*

Description

Plot a TSMP object

Usage

```
## S3 method for class 'ArcCount'
plot(x, data, type = c("data", "matrix"),
  exclusion_zone = NULL, edge_limit = NULL,
  threshold = stats::quantile(x$cac, 0.1), main = "Arcs Discover",
  xlab = "index", ylab = "", ...)

## S3 method for class 'Valmod'
plot(x, ylab = "distance", xlab = "index",
  main = "Valmod Matrix Profile", data = FALSE, ...)

## S3 method for class 'MatrixProfile'
plot(x, ylab = "distance", xlab = "index",
  main = "Unidimensional Matrix Profile", data = FALSE, ...)

## S3 method for class 'MultiMatrixProfile'
plot(x, ylab = "distance", xlab = "index",
  main = "Multidimensional Matrix Profile", ...)

## S3 method for class 'SimpleMatrixProfile'
plot(x, ylab = "distance",
```

```
  xlab = "index", main = "SiMPle Matrix Profile", data = FALSE, ...)

## S3 method for class 'Fluss'
plot(x, data, type = c("data", "matrix"),
  main = "Fast Low-cost Unipotent Semantic Segmentation",
  xlab = "index", ylab = "", ...)

## S3 method for class 'Floss'
plot(x, data, type = c("data", "matrix"),
  main = "Fast Low-cost Online Semantic Segmentation", xlab = "index",
  ylab = "", ...)

## S3 method for class 'Chain'
plot(x, data, type = c("data", "matrix"),
  main = "Chain Discover", xlab = "index", ylab = "", ...)

## S3 method for class 'Discord'
plot(x, data, type = c("data", "matrix"), ncol = 3,
  main = "Discord Discover", xlab = "index", ylab = "", ...)

## S3 method for class 'Motif'
plot(x, data, type = c("data", "matrix"), ncol = 3,
  main = "MOTIF Discover", xlab = "index", ylab = "", ...)

## S3 method for class 'MultiMotif'
plot(x, data, type = c("data", "matrix"),
  ncol = 3, main = "Multidimensional MOTIF Discover", xlab = "index",
  ylab = "", ...)

## S3 method for class 'Salient'
plot(x, data, main = "Salient Subsections",
  xlab = "index", ylab = "", ...)
```

Arguments

x	a Matrix Profile
data	the data used to build the Matrix Profile, if not embedded to it.
type	"data" or "matrix". Choose what will be plotted.
exclusion_zone	if a number will be used instead of Matrix Profile's. (Default is NULL).
edge_limit	if a number will be used instead of Matrix Profile's exclusion zone. (Default is NULL).
threshold	the maximum value to be used to plot.
main	a string. Main title.
xlab	a string. X label.
ylab	a string. Y label.
...	further arguments to be passed to plot(). See par().
ncol	an int. Number of columns to plot Motifs.

Value

None

Examples

```
mp <- tsmp(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)
plot(mp)
```

`plot_arcs` *Plot arcs between indexes of a Profile Index*

Description

Sometimes may be useful to see where is the nearest neighbor graphically. This is the reasoning behind, for example, FLUSS which uses the arc count to infer a semantic change, and SiMPle which infer that arcs connect similar segments of a music. See details for a deeper explanation how to use this function.

Usage

```
plot_arcs(pairs, alpha = NULL, quality = 30, lwd = 15,
  col = c("blue", "orange"), main = "Arc Plot", ylab = "",
  xlab = "Profile Index", xmin = NULL, xmax = NULL, ...)
```

Arguments

`pairs`	a `matrix` with 2 columns.
`alpha`	a `numeric`. (Default is `NULL`, automatic). Alpha value for lines transparency.
`quality`	an `int`. (Default is `30`). Number of segments to draw the arc. Bigger value, harder to render.
`lwd`	an `int`. (Default is `15`). Line width.
`col`	a `vector` of colors. (Default is `c("blue", "orange")`). Colors for right and left arc, respectively. Accepts one color.
`main`	a `string`. (Default is `"Arc Plot"`). Main title.
`ylab`	a `string`. (Default is `""`). Y label.
`xlab`	a `string`. (Default is `"Profile Index"`). X label.
`xmin`	an `int`. (Default is `NULL`). Set the minimum value of x axis.
`xmax`	an `int`. (Default is `NULL`). Set the maximum value of x axis.
`...`	further arguments to be passed to `plot()`. See `par()`.

Details

You have two options to use this function. First you can provide just the data, and the function will try its best to retrieve the pairs for plotting. Second, you can skip the first parameters and just provide the `pairs`, which is a `matrix` with two columns; the first is the starting index, the second is the end index. Two colors are used to allow you to identify the direction of the arc. If you use the `rpi` or `lpi` as input, you will see that these profile indexes have just one direction.

`exclusion_zone` is used to filter out small arcs that may be useless (e.g. you may be interested in similarities that are far away). `edge_limit` is used to filter out spurious arcs that are used connect the beginning and the end of the profile (e.g. silent audio). `threshold` is used to filter indexes that have distant nearest neighbor (e.g. retrieve only the best motifs).

Value

None

Examples

```
plot_arcs(pairs = matrix(c(5, 10, 1, 10, 20, 5), ncol = 2, byrow = TRUE))
```

`remove_class` *Remove a* `TSMP` *class from an object*

Description

Remove a `TSMP` class from an object

Usage

```
remove_class(x, class)
```

Arguments

`x`	a `TSMP` object
`class`	`character` string with the class name

Value

the object without the class

Examples

```
w <- 50
data <- mp_gait_data
mp <- tsmp(data, window_size = w, exclusion_zone = 1 / 4, verbose = 0)
mp <- find_chains(mp)
# Remove the "Chain" class information
mp <- remove_class(mp, "Chain")
```

`salient_mds` *Convert salient sequences into MDS space*

Description

Convert salient sequences into MDS space

Usage

```
salient_mds(.mp, data, bit_idx = 1)
```

Arguments

`.mp`	a Matrix Profile object.
`data`	the data used to build the Matrix Profile, if not embedded.
`bit_idx`	an `int`. The index of `n_bits` used for MDL discretization if more than one was used. (Default is `1`).

Value

Returns X,Y values for plotting

References

- Yeh CCM, Van Herle H, Keogh E. Matrix profile III: The matrix profile allows visualization of salient subsequences in massive time series. Proc - IEEE Int Conf Data Mining, ICDM. 2017;579–88.

- Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E. Discovering the Intrinsic Cardinality and Dimensionality of Time Series Using MDL. In: 2011 IEEE 11th International Conference on Data Mining. IEEE; 2011. p. 1086–91.

Website: https://sites.google.com/site/salientsubs/

Examples

```
# toy example
data <- mp_toy_data$data[, 1]
mp <- tsmp(data, window_size = 30, verbose = 0)
mps <- salient_subsequences(mp, verbose = 0)
mds_data <- salient_mds(mps)
plot(mds_data, main = "Multi dimensional scale")
```

`salient_score` *Computes the F-Score of salient algorithm.*

Description

This score function is useful for testing several values of `n_bits` for MDL discretization and checking against a known set of indexes. This increase the probability of better results on relevant subsequence extraction.

Usage

```
salient_score(.mp, gtruth)
```

Arguments

`.mp`	a Matrix Profile object.
`gtruth`	a `vector` of `integers` with the indexes of relevant subsequences.

Value

Returns a `list` with `f_score`, `precision`, `recall` and `bits` used in the algorithm.

References

- Yeh CCM, Van Herle H, Keogh E. Matrix profile III: The matrix profile allows visualization of salient subsequences in massive time series. Proc - IEEE Int Conf Data Mining, ICDM. 2017;579–88.

- Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E. Discovering the Intrinsic Cardinality and Dimensionality of Time Series Using MDL. In: 2011 IEEE 11th International Conference on Data Mining. IEEE; 2011. p. 1086–91.

Website: https://sites.google.com/site/salientsubs/

Examples

```
# toy example
data <- mp_toy_data$data[, 1]
mp <- tsmp(data, window_size = 30, verbose = 0)
mps <- salient_subsequences(mp, n_bits = c(4, 6, 8), verbose = 0)
label_idx <- seq(2, 500, by = 110) # fake data
salient_score(mps, label_idx)
```

`salient_subsequences` *Framework for retrieve salient subsequences from a dataset*

Description

In order to allow a meaningful visualization in Multi-Dimensional Space (MDS), this function retrieves the most relevant subsequences using Minimal Description Length (MDL) framework.

Usage

```
salient_subsequences(.mp, data, n_bits = 8, n_cand = 10,
  exclusion_zone = NULL, verbose = 2)
```

Arguments

`.mp`	a TSMP object of class `MatrixProfile`.
`data`	the data used to build the Matrix Profile, if not embedded.
`n_bits`	an `int` or `vector` of `int`. Number of bits for MDL discretization. (Default is `8`).
`n_cand`	an `int`. number of candidate when picking the subsequence in each iteration. (Default is `10`).
`exclusion_zone`	if a `number` will be used instead of embedded value. (Default is `NULL`).
`verbose`	an `int`. See details. (Default is `2`).

Details

`verbose` changes how much information is printed by this function; `0` means nothing, `1` means text, `2` adds the progress bar, `3` adds the finish sound.

Value

Returns the input `.mp` object with a new name `salient`. It contains: `indexes`, a `vector` with the starting position of each subsequence, `idx_bit_size`, a `vector` with the associated bitsize for each iteration and `bits` the value used as input on `n_bits`.

References

- Yeh CCM, Van Herle H, Keogh E. Matrix profile III: The matrix profile allows visualization of salient subsequences in massive time series. Proc - IEEE Int Conf Data Mining, ICDM. 2017;579–88.
- Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E. Discovering the Intrinsic Cardinality and Dimensionality of Time Series Using MDL. In: 2011 IEEE 11th International Conference on Data Mining. IEEE; 2011. p. 1086–91.

Website: https://sites.google.com/site/salientsubs/

Examples

```
# toy example
data <- mp_toy_data$data[, 1]
mp <- tsmp(data, window_size = 30, verbose = 0)
mps <- salient_subsequences(mp, data, verbose = 0)
## Not run:
# full example
data <- mp_meat_data$sub$data
w <- mp_meat_data$sub$sub_len
mp <- tsmp(data, window_size = w, verbose = 2, n_workers = 6)
mps <- salient_subsequences(mp, data, n_bits = c(4, 6, 8), verbose = 2)

## End(Not run)
```

---

scrimp *Anytime univariate SCRIMP++ algorithm*

Description

Computes the best so far Matrix Profile and Profile Index for Univariate Time Series. DISCLAIMER: This algorithm still in development by its authors. Join similarity, RMP and LMP not implemented yet.

Usage

```
scrimp(..., window_size, exclusion_zone = 1/2, verbose = 2,
  s_size = Inf, pre_scrimp = 1/4)
```

Arguments

`...`	a `matrix` or a `vector`.
`window_size`	an `int`. Size of the sliding window.
`exclusion_zone`	a `numeric`. Size of the exclusion zone, based on window size (default is `1/2`). See details.
`verbose`	an `int`. See details. (Default is `2`).
`s_size`	a `numeric`. for anytime algorithm, represents the size (in observations) the random calculation will occur (default is `Inf`).
`pre_scrimp`	a `numeric`. Set the pre-scrimp step based on `window_size`, if `0`, disables pre-scrimp. (default is `1/4`).

Details

The Matrix Profile, has the potential to revolutionize time series data mining because of its generality, versatility, simplicity and scalability. In particular it has implications for time series motif discovery, time series joins, shapelet discovery (classification), density estimation, semantic segmentation, visualization, rule discovery, clustering etc. The anytime SCRIMP computes the Matrix Profile and Profile Index in such manner that it can be stopped before its complete calculation and return the best so far results allowing ultra-fast approximate solutions. `verbose` changes how much information is printed by this function; `0` means nothing, `1` means text, `2` adds the progress bar, `3` adds the finish sound. `exclusion_zone` is used to avoid trivial matches.

Value

Returns a `MatrixProfile` object, a `list` with the matrix profile `mp`, profile index `pi` left and right matrix profile `lmp`, `rmp` and profile index `lpi`, `rpi`, window size `w` and exclusion zone `ez`.

References

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other matrix profile computations: `mstomp_par`, `stamp_par`, `stomp_par`, `tsmp`, `valmod`

Examples

```
mp <- scrimp(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)
## Not run:
ref_data <- mp_toy_data$data[, 1]
query_data <- mp_toy_data$data[, 2]
# self similarity
mp <- scrimp(ref_data, window_size = 30, s_size = round(nrow(ref_data) * 0.1))
# join similarity
mp <- scrimp(ref_data, query_data, window_size = 30, s_size = round(nrow(query_data) * 0.1))

## End(Not run)
```

`sdts_predict` *Framework for Scalable Dictionary learning for Time Series (SDTS) prediction function*

Description

This function trains a model that uses a dictionary to predict state changes. Differently from `fluss()`, it doesn't look for semantic changes (that may be several), but for binary states like "on" or "off". Think for example that a human annotator is pressing a switch any time he thinks that the recorded data is relevant, and releases the switch when he thinks the data is noise. This algorithm will learn the switching points (even better) and try to predict using new data.

Usage

```
sdts_predict(model, data, window_size)
```

Arguments

model	a model created by SDTS training function sdts_train().
data	a vector of numeric. Time series.
window_size	an int. The average sliding window size.

Value

Returns a vector of logical with predicted annotations.

References

- Yeh C-CM, Kavantzas N, Keogh E. Matrix profile IV: Using Weakly Labeled Time Series to Predict Outcomes. Proc VLDB Endow. 2017 Aug 1;10(12):1802–12.

Website: https://sites.google.com/view/weaklylabeled

See Also

Other Scalable Dictionaries: sdts_score, sdts_train

Examples

```
# This is a fast toy example and results are useless. For a complete result, run the code inside
#' Not run' section below.
w <- c(110, 220)
subs <- 11000:20000
tr_data <- mp_test_data$train$data[subs]
tr_label <- mp_test_data$train$label[subs]
te_data <- mp_test_data$test$data[subs]
te_label <- mp_test_data$test$label[subs]
model <- sdts_train(tr_data, tr_label, w, verbose = 0)
predict <- sdts_predict(model, te_data, round(mean(w)))
sdts_score(predict, te_label, 1)
## Not run:
windows <- c(110, 220, 330)
model <- sdts_train(mp_test_data$train$data, mp_test_data$train$label, windows, verbose = 0)
predict <- sdts_predict(model, mp_test_data$test$data, round(mean(windows)))
sdts_score(predict, mp_test_data$test$label, 1)

## End(Not run)
```

sdts_score *Computes the F-Score of a SDTS prediction*

Description

Computes the F-Score of a SDTS prediction.

Usage

```
sdts_score(pred, gtruth, beta = 1)
```

Arguments

pred	a vector of logical. Predicted annotation from sdts_predict()
gtruth	a vector of logical. Ground truth annotation.
beta	a numeric. See details. (default is 1).

Details

beta is used to balance F-score towards recall (>1) or precision (<1).

Value

Returns a list with f_score, precision and recall.

References

- Yeh C-CM, Kavantzas N, Keogh E. Matrix profile IV: Using Weakly Labeled Time Series to Predict Outcomes. Proc VLDB Endow. 2017 Aug 1;10(12):1802–12.

Website: https://sites.google.com/view/weaklylabeled

See Also

Other Scalable Dictionaries: sdts_predict, sdts_train

Examples

```
# This is a fast toy example and results are useless. For a complete result, run the code inside
#' Not run' section below.
w <- c(110, 220)
subs <- 11000:20000
tr_data <- mp_test_data$train$data[subs]
tr_label <- mp_test_data$train$label[subs]
te_data <- mp_test_data$test$data[subs]
te_label <- mp_test_data$test$label[subs]
model <- sdts_train(tr_data, tr_label, w, verbose = 0)
predict <- sdts_predict(model, te_data, round(mean(w)))
sdts_score(predict, te_label, 1)
```

```
## Not run:
windows <- c(110, 220, 330)
model <- sdts_train(mp_test_data$train$data, mp_test_data$train$label, windows)
predict <- sdts_predict(model, mp_test_data$test$data, round(mean(windows)))
sdts_score(predict, mp_test_data$test$label, 1)

## End(Not run)
```

`sdts_train` *Framework for Scalable Dictionary learning for Time Series (SDTS) training function*

Description

This function trains a model that uses a dictionary to predict state changes. Differently from `fluss()`, it doesn't look for semantic changes (that may be several), but for binary states like "on" or "off". Think for example that a human annotator is pressing a switch any time he thinks that the recorded data is relevant, and releases the switch when he thinks the data is noise. This algorithm will learn the switching points (even better) and try to predict using new data.

Usage

```
sdts_train(data, label, window_size, beta = 1, pat_max = Inf,
  parallel = TRUE, verbose = 2)
```

Arguments

`data`	a `vector` of `numeric`. Time series.
`label`	a `vector` of `logical`. Annotations.
`window_size`	an `int` or a `vector` of `int`. Sliding window sizes.
`beta`	a `numeric`. See details. (default is `1`).
`pat_max`	an `int`. Max number of shape features captured. (default is `Inf`).
`parallel`	a `logical`. Use parallel computation inside (default is `TRUE`).
`verbose`	an `int`. See details. (Default is `2`).

Details

`beta` is used to balance F-score towards recall (`>1`) or precision (`<1`). `verbose` changes how much information is printed by this function; `0` means nothing, `1` means text, `2` adds the progress bar, `3` adds the finish sound.

Value

Returns a list with the learned dictionary `score` (estimated score), `score_hist` (history of scores), `pattern` (shape features), `thold` (threshold values).

References

- Yeh C-CM, Kavantzas N, Keogh E. Matrix profile IV: Using Weakly Labeled Time Series to Predict Outcomes. Proc VLDB Endow. 2017 Aug 1;10(12):1802–12.

Website: https://sites.google.com/view/weaklylabeled

See Also

Other Scalable Dictionaries: sdts_predict, sdts_score

Examples

```
# This is a fast toy example and results are useless. For a complete result, run the code inside
#' Not run' section below.
w <- c(110, 220)
subs <- 11000:20000
tr_data <- mp_test_data$train$data[subs]
tr_label <- mp_test_data$train$label[subs]
te_data <- mp_test_data$test$data[subs]
te_label <- mp_test_data$test$label[subs]
model <- sdts_train(tr_data, tr_label, w, verbose = 0)
predict <- sdts_predict(model, te_data, round(mean(w)))
sdts_score(predict, te_label, 1)
## Not run:
windows <- c(110, 220, 330)
model <- sdts_train(mp_test_data$train$data, mp_test_data$train$label, windows)
predict <- sdts_predict(model, mp_test_data$test$data, round(mean(windows)))
sdts_score(predict, mp_test_data$test$label, 1)

## End(Not run)
```

---

set_data *Set/changes the data included in TSMP object.*

Description

This may be useful if you want to include the data lately or remove the included data (set as NULL).

Usage

```
set_data(.mp, data)
```

Arguments

.mp	a TSMP object.
data	a matrix (for one series) or a list of matrices (for two series).

Value

Returns silently the original TSMP object with changed data.

Examples

```
mp <- tsmp(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)
mp <- set_data(mp, NULL)
```

`simple_fast` *Compute the join similarity for Sound data*

Description

Compute the join similarity for Sound data

Usage

```
simple_fast(..., window_size, exclusion_zone = 1/2, verbose = 2)
```

Arguments

`...`	a `matrix` of `numeric`, where each column is a time series. Accepts `list` and `data.frame` too. If a second time series is supplied it will be a join matrix profile.
`window_size`	an `int` with the size of the sliding window.
`exclusion_zone`	a `numeric`. Size of the exclusion zone, based on window size (default is `1/2`).
`verbose`	an `int`. See details. (Default is `2`).

Details

`verbose` changes how much information is printed by this function; `0` means nothing, `1` means text, `2` adds the progress bar, `3` adds the finish sound.

Value

Returns a `SimpleMatrixProfile` object, a `list` with the matrix profile `mp`, profile index `pi`, number of dimensions `n_dim`, window size `w` and exclusion zone `ez`.

References

- Silva D, Yeh C, Batista G, Keogh E. Simple: Assessing Music Similarity Using Subsequences Joins. Proc 17th ISMIR Conf. 2016;23–30.
- Silva DF, Yeh C-CM, Zhu Y, Batista G, Keogh E. Fast Similarity Matrix Profile for Music Analysis and Exploration. IEEE Trans Multimed. 2018;14(8):1–1.

Website: https://sites.google.com/view/simple-fast

Website: https://sites.google.com/site/ismir2016simple/home

Examples

```
w <- 30
data <- mp_toy_data$data # 3 dimensions matrix
result <- simple_fast(data, window_size = w, verbose = 0)
```

`stamp_par` *Anytime univariate STAMP algorithm Parallel version*

Description

Computes the best so far Matrix Profile and Profile Index for Univariate Time Series.

Usage

```
stamp_par(..., window_size, exclusion_zone = 1/2, verbose = 2,
  s_size = Inf, n_workers = 2, weight = NULL)

stamp(..., window_size, exclusion_zone = 1/2, verbose = 2,
  s_size = Inf, weight = NULL)
```

Arguments

`...`	a `matrix` or a `vector`. If a second time series is supplied it will be a join matrix profile.
`window_size`	an `int`. Size of the sliding window.
`exclusion_zone`	a `numeric`. Size of the exclusion zone, based on window size (default is `1/2`). See details.
`verbose`	an `int`. See details. (Default is `2`).
`s_size`	a `numeric`. for anytime algorithm, represents the size (in observations) the random calculation will occur (default is `Inf`).
`n_workers`	an `int`. Number of workers for parallel. (Default is `2`).
`weight`	a `vector` of `numeric` or `NULL` with the same length of the `window_size`. This is a MASS extension to weight the query.

Details

The Matrix Profile, has the potential to revolutionize time series data mining because of its generality, versatility, simplicity and scalability. In particular it has implications for time series motif discovery, time series joins, shapelet discovery (classification), density estimation, semantic segmentation, visualization, rule discovery, clustering etc. The anytime STAMP computes the Matrix Profile and Profile Index in such manner that it can be stopped before its complete calculation and return the best so far results allowing ultra-fast approximate solutions. `verbose` changes how much information is printed by this function; `0` means nothing, `1` means text, `2` adds the progress bar, `3` adds the finish sound. `exclusion_zone` is used to avoid trivial matches; if a query data is provided (join similarity), this parameter is ignored.

Value

Returns a `MatrixProfile` object, a `list` with the matrix profile `mp`, profile index `pi` left and right matrix profile `lmp`, `rmp` and profile index `lpi`, `rpi`, window size `w` and exclusion zone `ez`.

Functions

- `stamp_par`: Parallel version.
- `stamp`: Single thread version.

References

- Yeh CCM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, et al. Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. Proc - IEEE Int Conf Data Mining, ICDM. 2017;1317–22.
- Zhu Y, Imamura M, Nikovski D, Keogh E. Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining. Knowl Inf Syst. 2018 Jun 2;1–27.

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other matrix profile computations: `mstomp_par`, `scrimp`, `stomp_par`, `tsmp`, `valmod`

Examples

```
mp <- stamp(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)

# using threads
mp <- stamp_par(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)
## Not run:
ref_data <- mp_toy_data$data[, 1]
query_data <- mp_toy_data$data[, 2]
# self similarity
mp <- stamp(ref_data, window_size = 30, s_size = round(nrow(ref_data) * 0.1))
# join similarity
mp <- stamp(ref_data, query_data, window_size = 30, s_size = round(nrow(query_data) * 0.1))

## End(Not run)
```

---

stompi_update *Real-time STOMP algorithm*

Description

Real-time STOMP algorithm

Usage

```
stompi_update(.mp, new_data, history_size = FALSE)
```

Arguments

.mp	a TSMP object of class MatrixProfile.
new_data	new data to append to original data.
history_size	an int or FALSE. (Default is FALSE). Keep only this amount of data in the object. The value is for the data, not the matrix profile. Notice that the lmpand lpi will be inconsistent when repeatedly updating limiting the history size and thus will affect the mp and pi.

Value

Returns the input .mp updated with the new information.

Examples

```
mp <- tsmp(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)
mpi <- stompi_update(mp, mp_toy_data$data[201:300, 1])
mp <- tsmp(mp_toy_data$data[1:300, 1], window_size = 30, verbose = 0)
all.equal(mp, mpi, check.attributes = FALSE)
```

---

stomp_par *Univariate STOMP algorithm*

Description

Computes the Matrix Profile and Profile Index for Univariate Time Series.

Usage

```
stomp_par(..., window_size, exclusion_zone = 1/2, verbose = 2,
  n_workers = 2)

stomp(..., window_size, exclusion_zone = 1/2, verbose = 2)
```

Arguments

...	a matrix or a vector. If a second time series is supplied it will be a join matrix profile.
window_size	an int. Size of the sliding window.
exclusion_zone	a numeric. Size of the exclusion zone, based on window size (default is 1/2). See details.
verbose	an int. See details. (Default is 2).
n_workers	an int. Number of workers for parallel. (Default is 2).

Details

The Matrix Profile, has the potential to revolutionize time series data mining because of its generality, versatility, simplicity and scalability. In particular it has implications for time series motif discovery, time series joins, shapelet discovery (classification), density estimation, semantic segmentation, visualization, rule discovery, clustering etc. `verbose` changes how much information is printed by this function; `0` means nothing, `1` means text, `2` adds the progress bar, `3` adds the finish sound. `exclusion_zone` is used to avoid trivial matches; if a query data is provided (join similarity), this parameter is ignored.

Value

Returns a `MatrixProfile` object, a `list` with the matrix profile `mp`, profile index `pi` left and right matrix profile `lmp`, `rmp` and profile index `lpi`, `rpi`, window size `w` and exclusion zone `ez`.

Functions

- `stomp_par`: Parallel version.
- `stomp`: Single thread version.

References

- Zhu Y, Zimmerman Z, Senobari NS, Yeh CM, Funning G. Matrix Profile II : Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. Icdm. 2016 Jan 22;54(1):739–48.

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other matrix profile computations: `mstomp_par`, `scrimp`, `stamp_par`, `tsmp`, `valmod`

Examples

```
mp <- stomp(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)

# using threads
mp <- stomp_par(mp_toy_data$data[1:400, 1], window_size = 30, verbose = 0)
## Not run:
ref_data <- mp_toy_data$data[, 1]
query_data <- mp_toy_data$data[, 2]
# self similarity
mp <- stomp(ref_data, window_size = 30)
# join similarity
mp2 <- stomp(ref_data, query_data, window_size = 30)

## End(Not run)
```

tsmp *Computes the Matrix Profile and Profile Index*

Description

This is a wrap function that makes easy to use all available algorithms to compute the Matrix Profile and Profile Index for multiple purposes.

Usage

```
tsmp(..., window_size, exclusion_zone = 1/2, mode = c("stomp", "stamp",
  "simple", "mstomp", "scrimp", "valmod"), verbose = 2, n_workers = 1,
  s_size = Inf, must_dim = NULL, exc_dim = NULL, heap_size = 50,
  paa = 1, .keep_data = TRUE)
```

Arguments

`...`	a `matrix` or a `vector`. If a second time series is supplied it will be a join matrix profile (except for `mstomp()`).
`window_size`	an `int` with the size of the sliding window. Use a vector for Valmod.
`exclusion_zone`	a `numeric`. Size of the exclusion zone, based on window size (default is `1/2`). See details.
`mode`	the algorithm that will be used to compute the matrix profile. (Default is `stomp`). See details.
`verbose`	an `int`. (Default is `2`). See details.
`n_workers`	an `int`. Number of workers for parallel. (Default is `1`).
`s_size`	a `numeric`. for anytime algorithm, represents the size (in observations) the random calculation will occur (default is `Inf`). See details.
`must_dim`	an `int` or `vector` of which dimensions to forcibly include (default is `NULL`). See details.
`exc_dim`	an `int` or `vector` of which dimensions to exclude (default is `NULL`). See details.
`heap_size`	an `int`. (Default is `50`). Size of the distance profile heap buffer.
`paa`	an `int`. (Default is `1`). Factor of PAA reduction (2 == half of size)
`.keep_data`	a `logical`. (Default is `TRUE`). Keeps the data embedded to resultant object.

Details

The Matrix Profile, has the potential to revolutionize time series data mining because of its generality, versatility, simplicity and scalability. In particular it has implications for time series motif discovery, time series joins, shapelet discovery (classification), density estimation, semantic segmentation, visualization, rule discovery, clustering etc.

The first algorithm invented was the `stamp()` that using `mass()` as an ultra-fast Algorithm for Similarity Search allowed to compute the Matrix Profile in reasonable time. One of its main feature

was its Anytime property which using a randomized approach could return a "best-so-far" matrix that could give us the correct answer (using for example 1/10 of all iterations) almost every time.

The next algorithm was `stomp()` that currently is the most used. Researchers noticed that the dot products do not need to be recalculated from scratch for each subsequence. Instead, we can reuse the values calculated for the first subsequence to make a faster calculation in the next iterations. The idea is to make use of the intersections between the required products in consecutive iterations. This approach reduced the time to compute the Matrix Profile to about 3

Currently there is a new algorithm that I'll not explain further here. It is called `scrimp()`, and is as fast as `stomp()`, and have the Anytime property. This algorithm is implemented in this package, but still waiting for an article publication.

Further, there is the `mstomp()` that computes a multidimensional Matrix Profile that allows to meaningful MOTIF discovery in Multivariate Time Series. And `simple_fast()` that also handles Multivariate Time Series, but focused in Music Analysis and Exploration.

The `valmod()` uses a new pruning algorithm allowing a similarity search with a range of sliding window sizes.

Some parameters are global across the algorithms:

... One or two time series (except for `mstomp()`). The second time series can be smaller than the first.

window_size The sliding window.

exclusion_zone Is used to avoid trivial matches; if a query data is provided (join similarity), this parameter is ignored.

verbose Changes how much information is printed by this function; `0` means nothing, `1` means text, `2` adds the progress bar, `3` adds the finish sound.

n_workers number of threads for parallel computing (except `simple_fast`, `scrimp` and `valmod`). If the value is 2 or more, the '_par' version of the algorithm will be used.

`s_size` is used only in Anytime algorithms: `stamp()` and `scrimp()`. `must_dim` and `exc_dim` are used only in `mstomp()`. `heap_size` is used only for `valmod()` `mode` can be any of the following: `stomp`, `stamp`, `simple`, `mstomp`, `scrimp`, `valmod`.

Value

Returns the matrix profile `mp` and profile index `pi`. It also returns the left and right matrix profile `lmp`, `rmp` and profile index `lpi`, `rpi` that may be used to detect Time Series Chains. `mstomp()` returns a multidimensional Matrix Profile.

References

- Silva D, Yeh C, Batista G, Keogh E. Simple: Assessing Music Similarity Using Subsequences Joins. Proc 17th ISMIR Conf. 2016;23–30.
- Silva DF, Yeh C-CM, Zhu Y, Batista G, Keogh E. Fast Similarity Matrix Profile for Music Analysis and Exploration. IEEE Trans Multimed. 2018;14(8):1–1.
- Yeh CM, Kavantzas N, Keogh E. Matrix Profile VI : Meaningful Multidimensional Motif Discovery.

- Yeh CCM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, et al. Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. Proc - IEEE Int Conf Data Mining, ICDM. 2017;1317–22.

- Zhu Y, Imamura M, Nikovski D, Keogh E. Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining. Knowl Inf Syst. 2018 Jun 2;1–27.

- Zhu Y, Zimmerman Z, Senobari NS, Yeh CM, Funning G. Matrix Profile II : Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. Icdm. 2016 Jan 22;54(1):739–48.

Website: https://sites.google.com/view/simple-fast

Website: https://sites.google.com/site/ismir2016simple/home

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other matrix profile computations: mstomp_par, scrimp, stamp_par, stomp_par, valmod

Examples

```
# default with [stomp()]
mp <- tsmp(mp_toy_data$data[1:200, 1], window_size = 30, verbose = 0)

# parallel with [stomp_par()]
mp <- tsmp(mp_test_data$train$data[1:1000, 1], window_size = 30, n_workers = 2, verbose = 0)

# Anytime STAMP
mp <- tsmp(mp_toy_data$data[1:200, 1], window_size = 30, mode = "stamp", s_size = 50, verbose = 0)

# [mstomp()]
mp <- tsmp(mp_toy_data$data[1:200, ], window_size = 30, mode = "mstomp", verbose = 0)

# [simple_fast()]
mp <- tsmp(mp_toy_data$data[1:200, ], window_size = 30, mode = "simple", verbose = 0)
```

---

valmod *Variable Length Motif Discovery*

Description

Computes the Matrix Profile and Profile Index for a range of query window sizes

Usage

```
valmod(..., window_min, window_max, heap_size = 50,
  exclusion_zone = 1/2, lb = TRUE, verbose = 2)
```

Arguments

...	a matrix or a vector. If a second time series is supplied it will be a join matrix profile.
window_min	an int. Minimum size of the sliding window.
window_max	an int. Maximum size of the sliding window.
heap_size	an int. (Default is 50). Size of the distance profile heap buffer
exclusion_zone	a numeric. Size of the exclusion zone, based on window size (default is 1/2). See details.
lb	a logical. (Default is TRUE). If FALSE all window sizes will be calculated using STOMP instead of pruning. This is just for academic purposes.
verbose	an int. See details. (Default is 2).

Details

This algorithm uses an exact algorithm based on a novel lower bounding technique, which is specifically designed for the motif discovery problem. verbose changes how much information is printed by this function; 0 means nothing, 1 means text, 2 adds the progress bar, 3 adds the finish sound. exclusion_zone is used to avoid trivial matches; if a query data is provided (join similarity), this parameter is ignored.

Value

Returns a Valmod object, a list with the matrix profile mp, profile index pi left and right matrix profile lmp, rmp and profile index lpi, rpi, best window size w for each index and exclusion zone ez. Additionally: evolution_motif the best motif distance per window size, and non-length normalized versions of mp, pi and w: mpnn, pinn and wnn.

References

- Linardi M, Zhu Y, Palpanas T, Keogh E. VALMOD: A Suite for Easy and Exact Detection of Variable Length Motifs in Data Series. In: Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18. New York, New York, USA: ACM Press; 2018. p. 1757–60.

Website: http://www.cs.ucr.edu/~eamonn/MatrixProfile.html

See Also

Other matrix profile computations: mstomp_par, scrimp, stamp_par, stomp_par, tsmp

Examples

```
mp <- valmod(mp_toy_data$data[1:200, 1], window_min = 30, window_max = 40)
## Not run:
ref_data <- mp_toy_data$data[, 1]
query_data <- mp_toy_data$data[, 2]
# self similarity
mp <- valmod(ref_data, window_min = 30, window_max = 40)
```

```
# join similarity
mp <- valmod(ref_data, query_data, window_min = 30, window_max = 40)

## End(Not run)
```

Index

∗Topic **datasets**
mp_fluss_data, 23
mp_gait_data, 24
mp_meat_data, 25
mp_test_data, 26
mp_toy_data, 27
∗Topic **hplot**
plot, 29
plot_arcs, 31

as.arccount *(*as.matrixprofile*)*, 3
as.chain *(*as.matrixprofile*)*, 3
as.discord *(*as.matrixprofile*)*, 3
as.fluss *(*as.matrixprofile*)*, 3
as.matrixprofile, 3
as.motif *(*as.matrixprofile*)*, 3
as.multimatrixprofile *(*as.matrixprofile*)*, 3
as.multimotif *(*as.matrixprofile*)*, 3
as.salient *(*as.matrixprofile*)*, 3
as.valmod *(*as.matrixprofile*)*, 3
av_apply, 4, *5–9*
av_complexity, *5*, 5, *6–9*
av_hardlimit_artifact, *5*, 6, *7–9*
av_motion_artifact, *5, 6*, 7, *8, 9*
av_stop_word, *5–7*, 8, *9*
av_zerocrossing, *5–8*, 9

dist_profile, 10

fast_movavg, 11
fast_movsd, 12
find_chains, 12
find_discord, 13
find_motif, 14
floss, 15, *17–22*
floss_cac, *16*, 16, *18–22*
floss_extract, *16, 17*, 17, *19–22*
fluss, *16–18*, 18, *20–22*
fluss(), *37*, *40*
fluss_cac, *16–19*, 19, *21, 22*
fluss_cac(), *18*
fluss_extract, *16–20*, 20, *22*
fluss_extract(), *18*, *21*
fluss_score, *16–21*, 21

get_data, 22

mass(), *47*
min_mp_idx, 23
mp_fluss_data, 23
mp_gait_data, 24
mp_meat_data, 25
mp_test_data, 26
mp_toy_data, 27
mstomp *(*mstomp_par*)*, 27
mstomp(), *47, 48*
mstomp_par, 27, *37*, *44*, *46*, *49, 50*

par(), *30, 31*
plot, 29
plot(), *30, 31*
plot_arcs, 31

remove_class, 32

salient_mds, 33
salient_score, 34
salient_subsequences, 35
scrimp, *29*, 36, *44*, *46*, *49, 50*
scrimp(), *48*
sdts_predict, 37, *39*, *41*
sdts_predict(), *39*
sdts_score, *38*, 39, *41*
sdts_train, *38, 39*, 40
sdts_train(), *38*
set_data, 41
simple_fast, 42
simple_fast(), *48*
stamp *(*stamp_par*)*, 43
stamp(), *47, 48*

stamp_par, *29*, *37*, 43, *46*, *49, 50*
stomp *(*stomp_par*)*, 45
stomp(), *28*, *48*
stomp_par, *29*, *37*, *44*, 45, *49, 50*
stompi_update, 44

tsmp, *29*, *37*, *44*, *46*, 47, *50*

valmod, *29*, *37*, *44*, *46*, *49*, 49
valmod(), *48*