Package 'DTWBI'

January 20, 2025

Type Package

Title Imputation of Time Series Based on Dynamic Time Warping

Version 1.1

Date 2018-07-10

Author Camille Dezecache, T. T. Hong Phan, Emilie Poisson-Caillault

Maintainer Emilie Poisson-Caillault <emilie.poisson@univ-littoral.fr>

Description Functions to impute large gaps within time series based on Dynamic Time Warping methods. It contains all required functions to create large missing consecutive values within time series and to fill them, according to the paper Phan et al. (2017), <DOI:10.1016/j.patrec.2017.08.019>. Performance criteria are added to compare similarity between two signals (query and reference).

Depends R (>= 3.0.0)

Imports dtw, rlist, stats, e1071, entropy, lsa

License GPL (>= 2)

RoxygenNote 6.0.1

URL http://mawenzi.univ-littoral.fr/DTWBI/

NeedsCompilation no

Repository CRAN

Date/Publication 2018-07-11 10:50:16 UTC

Contents

DTWBI-package 2
compute.fa2 3
compute.fb 4
compute.fsd 5
compute.nmae 6
compute.rmse 7
compute.sim 8
dataDTWBI 9
dist_afbdtw 9

DTWBI_univariate 10
gapCreation 12
local.derivative.ddtw 13
minCost 13

Index **15**

DTWBI-package *Imputation of Time Series Based on Dynamic Time Warping*

Description

Functions to impute large gaps within time series based on Dynamic Time Warping methods. It contains all required functions to create large missing consecutive values within time series and to fill them, according to the paper Phan et al. (2017), <DOI:10.1016/j.patrec.2017.08.019>. Performance criteria are added to compare similarity between two signals (query and reference).

Details

Index of help topics:

```
DTWBI-package           Imputation of Time Series Based on Dynamic Time
                        Warping
DTWBI_univariate        DTWBI algorithm for univariate signals
compute.fa2             FA2
compute.fb              Fractional Bias (FB)
compute.fsd             Fraction of Standard Deviation (FSD)
compute.nmae            Normalized Mean Absolute Error (NMAE)
compute.rmse            Root Mean Square Error (RMSE)
compute.sim             Similarity
dataDTWBI               Six univariate signals as example for DTWBI
                        package
dist_afbdtw             Adaptive Feature Based Dynamic Time Warping
                        algorithm
gapCreation             Gap creation
local.derivative.ddtw   Local derivative estimate to compute DDTW
minCost                 DTW-based methods for univariate signals
```

Author(s)

Camille Dezecache, T. T. Hong Phan, Emilie Poisson-Caillault

Maintainer: Emilie Poisson-Caillault <emilie.poisson@univ-littoral.fr>

References

Thi-Thu-Hong Phan, Emilie Poisson-Caillault, Alain Lefebvre, Andre Bigand. Dynamic time warping- based imputation for univariate time series data. Pattern Recognition Letters, Elsevier, 2017, <DOI:10.1016/j.patrec.2017.08.019>. <hal-01609256>

Examples

```
# Load package dataset
data(dataDTWBI)

# Create a query and a reference signal
query <- dataDTWBI$query
ref <- dataDTWBI$query

# Create a gap within query (10% of signal size)
query <- gapCreation(query, rate = 0.1)
data <- query$output_vector
begin_gap <- query$begin_gap
size_gap <- query$gap_size

# Fill gap using DTWBI algorithm
results_DTWBI <- DTWBI_univariate(data, t_gap = begin_gap, T_gap = size_gap)

# Plot
plot(ref, type = "l")
lines(results_DTWBI$output_vector, col = "red", lty = "dashed")

# Compute the similarity of imputed vector and reference
compute.sim(ref, results_DTWBI$output_vector)
```

---

compute.fa2 *FA2*

Description

Estimates the FA2 of two univariate signals Y (imputed values) and X (true values).

Usage

```
compute.fa2(Y, X, verbose = F)
```

Arguments

Y	vector of imputed values
X	vector of true values
verbose	if TRUE, print advice about the quality of the model

Details

This function returns the value of FA2 of two vectors corresponding to univariate signals X (true values) and Y (imputed values). This FA2 corresponds to the percentage of pairs of values (x_i, y_i) satisfying the condition $0,5 <= (Y_i/X_i) <= 2$. The closer FA2 is to 1, the more accurate is the imputation model. Both vectors Y and X must be of equal length, on the contrary an error will be displayed. In both input vectors, eventual NA will be exluded with a warning diplayed.

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1] ; Y <- dataDTWBI[, 2]
compute.fa2(Y,X)
compute.fa2(Y,X, verbose = TRUE)

# By definition, if pairs of true and imputed values are zero,
# FA2 corresponding to this pair of values equals 1.
X[1] <- 0
Y[1] <- 0
compute.fa2(Y,X)
```

---

compute.fb *Fractional Bias (FB)*

Description

Estimates the Fractional Bias (FB) of two univariate signals Y (imputed values) and X (true values).

Usage

```
compute.fb(Y, X, verbose = F)
```

Arguments

Y	vector of imputed values
X	vector of true values
verbose	if TRUE, print advice about the quality of the model

Details

This function returns the value of FB of two vectors corresponding to univariate signals, indicating whether predicted values are underestimated or overestimated compared to true values. A perfect imputation model gets $FB = 0$. An acceptable imputation model gives $FB <= 0.3$. Both vectors Y and X must be of equal length, on the contrary an error will be displayed. In both input vectors, eventual NA will be exluded with a warning diplayed.

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1] ; Y <- dataDTWBI[, 2]
compute.fb(Y,X)
compute.fb(Y,X, verbose = TRUE)

# If mean(X)=mean(Y)=0, it is impossible to estimate FB,
# unless both true and imputed values vectors are constant.
# By definition, in this case, FB = 0.
X <- rep(0, 10) ; Y <- rep(0, 10)
compute.fb(Y,X)

# If true and imputed values are not zero and are opposed, FB = Inf.
X <- rep(runif(1), 10)
Y <- -X
compute.fb(Y,X)
```

`compute.fsd` *Fraction of Standard Deviation (FSD)*

Description

Estimates the Fraction of Standard Deviation (FSD) of two univariate signals Y (imputed values) and X (true values).

Usage

```
compute.fsd(Y, X, verbose = F)
```

Arguments

`Y`	vector of imputed values
`X`	vector of true values
`verbose`	if TRUE, print advice about the quality of the model

Details

This function returns the value of FSD of two vectors corresponding to univariate signals. Values of FSD closer to zero indicate a better performance method for the imputation task. Both vectors Y and X must be of equal length, on the contrary an error will be displayed. In both input vectors, eventual NA will be exluded with a warning diplayed.

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1] ; Y <- dataDTWBI[, 2]
compute.fsd(Y,X)
compute.fsd(Y,X, verbose = TRUE)

# By definition, if true and imputed values are equal and constant,
# FSD = 0.
X <- rep(runif(1), 10)
Y <- X
compute.fsd(Y,X)

# However, if true and imputed values are constant but different,
# FSD is not calculable. An error is displayed.
## Not run:
X <- rep(runif(1), 10);Y <- rep(runif(1), 10)
compute.fsd(Y,X)
## End(Not run)
```

---

compute.nmae *Normalized Mean Absolute Error (NMAE)*

Description

Estimates the Normalized Mean Absolute Error of two univariate signals Y (imputed values) and X (true values).

Usage

```
compute.nmae(Y, X)
```

Arguments

`Y`	vector of imputed values
`X`	vector of true values

Details

This function returns the value of NMAE of two vectors corresponding to univariate signals. A lower NMAE ($NMAE \in [0, \inf]$) value indicates a better performance method for the imputation task. Both vectors Y and X must be of equal length, on the contrary an error will be displayed. In both input vectors, eventual NA will be exluded with a warning diplayed.

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1] ; Y <- dataDTWBI[, 2]
compute.nmae(Y,X)

# If true values is a constant vector, NMAE = Inf.
# A warning is displayed and MAE is estimated instead of NMAE,
# unless true and imputed values are equal. In this case,
# by definition, NMAE = 0.
X <- rep(0, 10)
Y <- runif(10)
compute.nmae(Y,X) # MAE computed
Y <- X
compute.nmae(Y,X) # By definition, NMAE = 0
```

`compute.rmse` *Root Mean Square Error (RMSE)*

Description

Estimates the Root Mean Square Error of two univariate signals Y (imputed values) and X (true values).

Usage

```
compute.rmse(Y, X)
```

Arguments

Y	vector of imputed values
X	vector of true values

Details

This function returns the value of RMSE of two vectors corresponding to univariate signals. A lower RMSE ($RMSE \in [0, \inf]$) value indicates a better performance method for the imputation task. Both vectors Y and X must be of equal length, on the contrary an error will be displayed. In both input vectors, eventual NA will be exluded with a warning diplayed.

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1] ; Y <- dataDTWBI[, 2]
compute.rmse(Y,X)
```

`compute.sim` *Similarity*

Description

Estimates the percentage of similarity of two univariate signals Y (imputed values) and X (true values).

Usage

```
compute.sim(Y, X)
```

Arguments

`Y`	vector of imputed values
`X`	vector of true values

Details

This function returns the value of similarity of two vectors corresponding to univariate signals. A higher similarity ($Similarity \in [0, 1]$) highlights a more accurate method for completing missing values in univariate datasets. Both vectors Y and X must be of equal length, on the contrary an error will be displayed. In both input vectors, eventual NA will be excluded with a warning diplayed.

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1] ; Y <- dataDTWBI[, 2]
compute.sim(Y,X)

# By definition, if true values is a constant vector
# and one or more imputed values are equal to the true values,
# similarity = 1.
X <- rep(2, 10)
Y <- X
compute.sim(Y,X)
```

dataDTWBI — *Six univariate signals as example for DTWBI package*

Description

Query and ref1 are two dephased sigmoidal signals. Ref2 presents a linear decrease. Ref3 and ref4 are constant signals of value 3 and 0 respectively. Ref5 is similar to the query with small noise added.

Usage

```
dataDTWBI
```

Format

A data frame with six variables: `query`, `ref1`, `ref2`, `ref3`, `ref4` and `ref5`.

dist_afbdtw — *Adaptive Feature Based Dynamic Time Warping algorithm*

Description

This function estimates a distance matrix which is used as an input in dtw() function (package dtw) to align two univariate signals following Adaptative Feature Based Dynamic Time Warping algorithm (AFBDTW).

Usage

```
dist_afbdtw(q, r, w1 = 0.5)
```

Arguments

`q`	query vector
`r`	reference vector
`w1`	weight of local feature VS global feature. By default, w1 = 0.5, and by definition, w2 = 1 - w1.

Value

A list containing the following elements:

- query: the query vector
- response: the response vector
- query_local: local feature of the query
- response_local: local feature of the response vector

- query_global: global feature of the query
- response_global: global feature of the response vector
- dist_local: distance matrix of the local feature
- dist_local: distance matrix of the global feature
- distAFBDTW: AFBDTW distance matrix

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1] ; Y <- dataDTWBI[, 2]
AFBDTW_Dist <- dist_afbdtw(X, Y)
```

`DTWBI_univariate` *DTWBI algorithm for univariate signals*

Description

Imputes values of a gap of position t_gap and size T in a univariate signal based on DTW algorithm. For more details on the method, see Phan et al. (2017) DOI: <10.1016/j.patrec.2017.08.019>. Default arguments of dtw() function are used but can be manually explicited and modified.

Usage

```
DTWBI_univariate(data, t_gap, T_gap, DTW_method = "DTW",
  threshold_cos = NULL, step_threshold = NULL, thresh_cos_stop = 0.8, ...)
```

Arguments

`data`	input vector containing a large and continuous gap (eventually derived from local.derivative.ddtw() function)
`t_gap`	location of the begining of the gap (eventually extracted from gapCreation function)
`T_gap`	gap size (eventually extracted from gapCreation function)
`DTW_method`	DTW method used for imputation ("DTW", "DDTW", "AFBDTW"). By default "DTW".
`threshold_cos`	threshold used to define similar sequences to the query. By default, threshold_cos=0.9995 if sequence is longer than 10'000, and threshold_cos=0.995 if shorter.
`step_threshold`	step used within the loop determining the threshold. By default, step_threshold=50 if sequence is longer than 10'000, step_threshold=10 if sequence length is between 1'000 and 10'000. Else, step_threshold=2.

thresh_cos_stop
: Define the lowest cosine threshold acceptable to find a similar window to the query. By default, thresh_cos_stop=0.8.

...
: additional arguments from the dtw() function

Value

DTWBI_univariate returns a list containing the following elements:

- output_vector: output vector containing complete data including the imputation proposal
- input_vector: original vector used as input
- query: the query i.e. the adjacent sequence to the gap
- pos_query: index of the begining and end of the query
- sim_window: vector containing the values of the most similar sequence to the query
- pos_sim_window: index of the begining and end of the similar window
- imputation_window: vector containing imputed values
- pos_imp_window: index of the begining and end of the imputation window

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1]

rate <- 0.1
output <- gapCreation(X, rate)
data <- output$output_vector
gap_begin <- output$begin_gap
gap_size <- output$gap_size
imputed_data <- DTWBI_univariate(data, t_gap=gap_begin, T_gap=gap_size)
plot(imputed_data$input_vector, type = "l", lwd = 2) # Uncomplete signal
lines(imputed_data$output_vector, col = "red") # Imputed signal
lines(y = imputed_data$query,
      x = imputed_data$pos_query[1]:imputed_data$pos_query[2],
      col = "green", lwd = 4) # Query
lines(y = imputed_data$sim_window,
      x = imputed_data$pos_sim_window[1]:imputed_data$pos_sim_window[2],
      col = "orange", lwd = 4) # Similar sequence to the query
lines(y = imputed_data$imputation_window,
      x = imputed_data$pos_imp_window[1]:imputed_data$pos_imp_window[2],
      col = "blue", lwd = 4) # Imputing proposal
```

`gapCreation` *Gap creation*

Description

This function creates a large continuous gap within a univariate signal. Gap size is defined as a percentage of input vector length. By default, the created gap starts at a random location.

Usage

```
gapCreation(X, rate, begin = NULL)
```

Arguments

`X`	input vector
`rate`	size of desired gap, as a percentage of input vector size
`begin`	location of the begining of the gap (random by default)

Value

gapCreation returns a list containing the following elements:

- output_vector: output vector containing the created gap
- input_vector: original vector used as input
- begin_gap: index of the begining of the gap
- rate: size of the created gap in percentage of the input vector length
- gap_size: length of the created gap

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1]
rate <- 0.1
output <- gapCreation(X, rate)
plot(output$input_vector, type = "l", col = "red", lwd = 2)
lines(output$output_vector, lty = "dashed", lwd = 2)
```

local.derivative.ddtw *Local derivative estimate to compute DDTW*

Description

This function estimates the local derivative of a vector. It can be used as an input in dtw() function (package dtw) to align two univariate signals.

Usage

```
local.derivative.ddtw(X)
```

Arguments

X	input vector from which local derivative has to be calculated

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1]
local.derivative.ddtw(X)

# Plot
plot(X, type = "b", ylim = c(-1, 1))
lines(local.derivative.ddtw(X), col = "red")
```

minCost *DTW-based methods for univariate signals*

Description

Finds the optimal alignment between two univariate time series based on DTW methods.

Usage

```
minCost(X, Y, method, ...)
```

Arguments

X	query vector
Y	response vector
method	"DTW", "DDTW", "AFBDTW", "DTW-D"
...	additional arguments from functions dtw or dist_afbdtw

Author(s)

Camille Dezecache, Hong T. T. Phan, Emilie Poisson-Caillault

Examples

```
data(dataDTWBI)
X <- dataDTWBI[, 1] ; Y <- dataDTWBI[, 2]

# Plot query and reference
plot(X, type = "l", ylim = c(-5,3))
lines(1:length(X), Y, col = "red")

#= Align signals using DTW
align_dtw <- minCost(X, Y, method = "DTW")
#= Align signals using DDTW
align_ddtw <- minCost(X, Y, method = "DDTW")
#= Align signals using AFBDTW
align_afbdtw <- minCost(X, Y, method = "AFBDTW")
#= Align signals using DTW-D
align_dtwd <- minCost(X, Y, method = "DTW-D")

#= Plots
library(dtw)
dtwPlotTwoWay(d = align_dtw, xts <- X, yts = Y, main = "DTW")
dtwPlotTwoWay(d = align_ddtw, xts <- X, yts = Y, main = "DDTW")
dtwPlotTwoWay(d = align_afbdtw, xts <- X, yts = Y, main = "AFBDTW")
dtwPlotTwoWay(d = align_dtwd, xts <- X, yts = Y, main = "DTW-D")

#= Compare cost of each method
comparative_cost <- matrix(c(align_dtw$normalizedDistance,
align_ddtw$normalizedDistance,
align_afbdtw$normalizedDistance,
align_dtwd$normalizedDistance), ncol = 4)
colnames(comparative_cost) <- c("DTW", "DDTW", "AFBDTW", "DTW-D")
comparative_cost
```

Index

∗ **DTW**
DTWBI-package, 2
∗ **datasets**
dataDTWBI, 9
∗ **imputation**
DTWBI-package, 2
∗ **package**
DTWBI-package, 2
∗ **times series**
DTWBI-package, 2

compute.fa2, 3
compute.fb, 4
compute.fsd, 5
compute.nmae, 6
compute.rmse, 7
compute.sim, 8

dataDTWBI, 9
dist_afbdtw, 9
DTWBI *(DTWBI-package)*, 2
DTWBI-package, 2
DTWBI_univariate, 10

gapCreation, 12

local.derivative.ddtw, 13

minCost, 13