

# Package ‘TSclust’

February 19, 2015

**Type** Package

**Title** Time Series Clustering Utilities

**Version** 1.2.3

**Date** 2014-4-30

**Encoding** UTF-8

**Author** Pablo Montero Manso, José Antonio Vilar

**Maintainer** Pablo Montero <pmontm@gmail.com>

**Description** This package contains a set of measures of dissimilarity between time series to perform time series clustering. Metrics based on raw data, on generating models and on the forecast behavior are implemented. Some additional utilities related to time series clustering are also provided, such as clustering algorithms and cluster evaluation metrics.

**License** GPL-2

**URL** <http://www.jstatsoft.org/v62/i01/>

**Depends** R (>= 3.0.1), wmtsa, pdc, cluster

**Imports** locpol, KernSmooth, dtw, longitudinalData

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2014-11-18 22:30:04

## R topics documented:

cluster.evaluation . . . . .	2
diss . . . . .	4
diss.ACF . . . . .	6
diss.AR.LPC.CEPS . . . . .	8
diss.AR.MAH . . . . .	9
diss.AR.PIC . . . . .	11
diss.CDM . . . . .	12
diss.CID . . . . .	14
diss.COR . . . . .	15
diss.CORT . . . . .	17

diss.DTWARP . . . . .	18
diss.DWT . . . . .	19
diss.EUCL . . . . .	20
diss.FRECHET . . . . .	21
diss.INT.PER . . . . .	22
diss.MINDIST.SAX . . . . .	23
diss.NCD . . . . .	25
diss.PDC . . . . .	27
diss.PER . . . . .	28
diss.PRED . . . . .	29
diss.SPEC.GLK . . . . .	32
diss.SPEC.ISD . . . . .	33
diss.SPEC.LLR . . . . .	35
electricity . . . . .	36
interest.rates . . . . .	37
loo1nn.cv . . . . .	37
paired.tseries . . . . .	38
pvalues.clust . . . . .	40
synthetic.tseries . . . . .	41
TSclust . . . . .	42
<b>Index</b>	<b>44</b>

---

cluster.evaluation      *Clustering Evaluation Index Based on Known Ground Truth*

---

### Description

Computes the similarity between the true cluster solution and the one obtained with a method under evaluation.

### Usage

```
cluster.evaluation(G, S)
```

### Arguments

- |   |  |
|---|--|
| G | Integer vector with the labels of the true cluster solution. Each element of the vector specifies the cluster 'id' that the element belongs to.            |
| S | Integer vector with the labels of the cluster solution to be evaluated. Each element of the vector specifies the cluster 'id' that the element belongs to. |

**Details**

The measure of clustering evaluation is defined as

$$Sim(G, C) = 1/k \sum_{i=1}^k \max_{1 \leq j \leq k} Sim(G_i, C_j),$$

where

$$Sim(G_i, C_j) = \frac{2|G_i \cap C_j|}{|G_i| + |C_j|}$$

with  $|I|$  denoting the cardinality of the elements in the set. This measure has been used for comparing different clusterings, e.g. in Kalpakis et al. (2001) and Pérttega and Vilar (2010).

**Value**

The computed index.

**Note**

This index is not symmetric.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Larsen, B. and Aone, C. (1999) Fast and effective text mining using linear-time document clustering. *Proc. KDD' 99*.16–22.

Kalpakis, K., Gada D. and Puttagunta, V. (2001) Distance measures for effective clustering of arima time-series. *Proceedings 2001 IEEE International Conference on Data Mining*, 273–280.

Pérttega S. and Vilar, J.A (2010) Comparing several parametric and nonparametric approaches to time series clustering: A simulation study. *J. Classification*, **27**(3), 333-362.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. *Journal of Statistical Software*, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[cluster.stats](#), [clValid](#), [std.ext](#)

**Examples**

```
#create a true cluster
#(first 4 elements belong to cluster '1', next 4 to cluster '2' and the last 4 to cluster '3'.
true_cluster <- c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3)
#the cluster to be tested
```

```

new_cluster <- c( 2, 1, 2, 3, 3, 2, 2, 1, 3, 3, 3, 3)

#get the index
cluster.evaluation(true_cluster, new_cluster)

#it can be seen that the index is not symmetric
cluster.evaluation(new_cluster, true_cluster)

```

---

diss

*TSclust Dissimilarity Computation*


---

## Description

Computes the dissimilarity matrix of the given numeric matrix, list, data.frame or mts object using the selected TSclust dissimilarity method.

## Usage

```
diss(SERIES, METHOD, ...)
```

## Arguments

SERIES	Numeric matrix, list, data.frame or mts object. Numeric matrices are interpreted row-wise (one series per row) meanwhile data.frame and mts objects are interpreted column-wise.
METHOD	the dissimilarity measure to be used. This must be one of "ACF", "AR.LPC.CEPS", "AR.MAH", "AR.PIC", "CDM", "CID", "COR", "CORT", "DTWARP", "DWT", "EUCL", "FRECHET", "INT.PER", "NCD", "PACF", "PDC", "PER", "PRED", "MINDIST.SAX", "SPEC.LLR", "SPEC.GLK" or "SPEC.ISD". Any unambiguous substring can be given. See details for individual usage.
...	Additional arguments for the selected method.

## Details

SERIES argument can be a numeric matrix, with one row per series, a list object with one numeric vector per element, a data.frame or a mts object. Some methods can have additional arguments. See the individual help page for each dissimilarity method, detailed below. Methods that have arguments that require one value per time series in series must provide so using a vector, a matrix (in the case of a multivalued argument) or a list when appropriate. In the case of a matrix, the values are conveyed row-wise. See the AR.LPC.CEPS example below.

- "ACF" Autocorrelation-based method. See [diss.ACF](#).
- "AR.LPC.CEPS" Linear Predictive Coding ARIMA method. This method has two value-per-series arguments, the ARIMA order, and the seasonality. See [diss.AR.LPC.CEPS](#).
- "AR.MAH" Model-based ARMA method. See [diss.AR.MAH](#).
- "AR.PIC" Model-based ARMA method. This method has a value-per-series argument, the ARIMA order. See [diss.AR.PIC](#).

- "CDM" Compression-based dissimilarity method. See [diss.CDM](#).
- "CID" Complexity-Invariant distance. See [diss.CID](#).
- "COR" Correlation-based method. See [diss.COR](#).
- "CORT" Temporal Correlation and Raw values method. See [diss.CORT](#).
- "DTWARP" Dynamic Time Warping method. See [diss.DTWARP](#).
- "DWT" Discrete wavelet transform method. See [diss.DWT](#).
- "EUCL" Euclidean distance. See [diss.EUCL](#). For many more conventional distances, see `link[stats]{dist}`, though you may need to transpose the dataset.
- "FRECHET" Frechet distance. See [diss.FRECHET](#).
- "INT.PER" Integrate Periodogram-based method. See [diss.INT.PER](#).
- "NCD" Normalized Compression Distance. See [diss.NCD](#).
- "PACF" Partial Autocorrelation-based method. See [diss.PACF](#).
- "PDC" Permutation distribution divergence. Uses the [pdc](#) package. See [pdcDist](#) for additional arguments and details. Note that series given by numeric matrices are interpreted row-wise and not column-wise, opposite as in [pdcDist](#).
- "PER" Periodogram-based method. See [diss.PER](#).
- "PRED" Prediction Density-based method. This method has two value-per-series argument, the logarithm and difference transform. See [diss.PRED](#).
- "MINDIST.SAX" Distance that lower bounds the Euclidean, based on the Symbolic Aggregate approximation measure. See [diss.MINDIST.SAX](#).
- "SPEC.LLR" Spectral Density by Local-Linear Estimation method. See [diss.SPEC.LLR](#).
- "SPEC.GLK" Log-Spectra Generalized Likelihood Ratio test method. See [diss.SPEC.GLK](#).
- "SPEC.ISD" Integrated Squared Differences between Log-Spectras method. See [diss.SPEC.ISD](#).

## Value

`dist` A `dist` object with the pairwise dissimilarities between series.

Some methods produce additional output, see their respective documentation pages for more information.

## Author(s)

Pablo Montero Manso, José Antonio Vilar.

## References

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

## See Also

[pdc](#), [dist](#)

**Examples**

```

data(electricity)
diss(electricity, METHOD="INT.PER", normalize=FALSE)

## Example of multivalued, one per series argument
## The AR.LPC.CEPS dissimilarity allows the specification of the ARIMA model for each series
## Create three sample time series and a mts object
x <- arima.sim(model=list(ar=c(0.4,-0.1)), n =100, n.start=100)
y <- arima.sim(model=list(ar=c(0.9)), n =100, n.start=100)
z <- arima.sim(model=list(ar=c(0.5, 0.2)), n =100, n.start=100)
seriests <- rbind(x,y,z)

## If we want to provide the ARIMA order for each series
## and use it with AR.LPC.CEPS, we create a matrix with the row-wise orders
orderx <- c(2,0,0)
ordery <- c(1,0,0)
orderz <- c(2,0,0)
orders = rbind(orderx, ordery, orderz)

diss( seriests, METHOD="AR.LPC.CEPS", k=30, order= orders )

##other examples
diss( seriests, METHOD="MINDIST.SAX", w=10, alpha=4 )
diss( seriests, METHOD="PDC" )

```

---

diss.ACF

*Autocorrelation-based Dissimilarity*


---

**Description**

Computes the dissimilarity between two time series as the distance between their estimated simple (ACF) or partial (PACF) autocorrelation coefficients.

**Usage**

```

diss.ACF(x, y, p = NULL, omega=NULL, lag.max=50)
diss.PACF(x, y, p = NULL, omega=NULL, lag.max=50)

```

**Arguments**

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
p	If not NULL, sets the weight for the geometric decaying of the autocorrelation coefficients. Ranging from 0 to 1.
lag.max	Maximum number of simple or partial autocorrelation coefficients to be considered.
omega	If not NULL, completely specifies the weighting matrix for the autocorrelation coefficients. p is ignored if omega is used.

**Details**

Performs the weighted Euclidean distance between the simple autocorrelation ( `dist.ACF` ) or partial autocorrelation ( `dist.PACF` ) coefficients. If neither `p` nor `omega` are specified, uniform weighting is used. If `p` is specified, geometric wights decaying with the lag in the form  $p(1 - p)^i$  are applied. If `omega` ( $\Omega$ ) is specified,

$$d(x, y) = \{(\hat{\rho}_x - \hat{\rho}_y)^t \Omega (\hat{\rho}_x - \hat{\rho}_y)\}^{\frac{1}{2}}$$

with  $\hat{\rho}_x$  and  $\hat{\rho}_y$  the respective (partial) autocorrelation coefficient vectors.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Galeano, P. and Peña, D. (2000). Multivariate analysis in vector time series. *Resenhas*, **4** (4), 383–403.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.COR](#)

**Examples**

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
z <- sin(seq(0, pi, length.out=100))
## Compute the distance and check for coherent results
diss.PACF(x, y)
diss.ACF(x, z)
diss.PACF(y, z)
#create a dist object for its use with clustering functions like pam or hclust
diss( rbind(x,y,z), "ACF", p=0.05)
```

---

diss.AR.LPC.CEPS

*Dissimilarity Based on LPC Cepstral Coefficients*


---

### Description

Computes the dissimilarity between two time series in terms of their Linear Predictive Coding (LPC) ARIMA processes.

### Usage

```
diss.AR.LPC.CEPS(x, y, k = 50, order.x=NULL, order.y=NULL,
  seasonal.x=list(order=c(0, 0, 0), period=NA),
  seasonal.y=list(order=c(0, 0, 0), period=NA),
  permissive=TRUE)
```

### Arguments

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
k	Number of cepstral coefficients to be considered.
order.x	Numeric matrix. Specifies the ARIMA models to be fitted for the series x. When using <code>diss</code> wrapper, use <code>order</code> argument instead. See details.
order.y	Numeric matrix. Specifies the ARIMA ARIMA models to be fitted for the series y. When using <code>diss</code> wrapper, use <code>order</code> argument instead. See details.
seasonal.x	A list of <code>arima</code> seasonal elements for series x. When using <code>diss</code> wrapper, use <code>seasonal</code> argument instead. See details.
seasonal.y	A list of <code>arima</code> seasonal elements for series x. When using <code>diss</code> wrapper, use <code>seasonal</code> argument instead. See details.
permissive	Specifies whether to force an AR order of 1 if no order is found. Ignored if neither <code>order.x</code> or <code>order.y</code> are NULL

### Details

If `order.x` or `order.y` are NULL, their respective series will be fitted automatically using a AR model. `order.x` and `order.y` contain the three components of the ARIMA model: the AR order, the degree of differencing and the MA order, specified as in the function `arima`.

`seasonal.x` and `seasonal.y` are lists with two components: 'order' and 'period'. See `seasonal` parameter of `arima`, except that specification using a numeric vector of length 3 is not allowed.

If using `diss` function with "AR.LPC.CEPS" method, the argument `order` must be used instead of `order.x` and `order.y`. `order` is a matrix with one row per series, specified as in `arima`. If `order` is NULL, automatic fitting imposing a AR model is performed. The argument `seasonal` is used instead of `seasonal.x` and `seasonal.y`. `seasonal` is a list of elements, one per series in the same order that the series are input. Each element of `seasonal` must have the same format as the one in `arima`.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Kalpakis, K., Gada D. and Puttagunta, V. (2001) Distance measures for effective clustering of arima time-series. *Proceedings 2001 IEEE International Conference on Data Mining*, 273–280.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. *Journal of Statistical Software*, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.AR.PIC](#), [diss.AR.MAH](#), [diss](#)

**Examples**

```
## Create three sample time series
x <- arima.sim(model=list(ar=c(0.4,-0.1)), n =100, n.start=100)
y <- arima.sim(model=list(ar=c(0.9)), n =100, n.start=100)
z <- arima.sim(model=list(ar=c(0.5, 0.2)), n =100, n.start=100)
## Compute the distance and check for coherent results
diss.AR.LPC.CEPS(x, y, 25) #impose an AR automatically selected for both series
#impose an ARIMA(2,0,0) for series x and an AR automatically selected for z
diss.AR.LPC.CEPS(x, z, 25, order.x = c(2,0,0), order.y = NULL )
diss.AR.LPC.CEPS(y, z, 25)
#create a dist object for its use with clustering functions like pam or hclust

diss( rbind(x,y,z), METHOD="AR.LPC.CEPS", k=20, order=rbind(c(2,0,0), c(1,0,0), c(2,0,0)),
      seasonal=list( list(order=c(1,0,0), period=1), list(order=c(2,0,0), period=3),
                    list(order=c(1,0,0), period=1)) )
```

---

diss.AR.MAH

*Model-based Dissimilarity Proposed by Maharaj (1996, 2000)*

---

**Description**

Computes the dissimilarity between two time series by testing whether both series are or not generated by the same ARMA model.

**Usage**

```
diss.AR.MAH(x, y, dependence=FALSE, permissive=TRUE)
```

**Arguments**

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
dependence	Boolean for considering dependence between observations of the series at the same point in time.
permissive	Boolean for continuing with the method even if no valid order is selected by AIC.

**Details**

Assuming that the time series  $x$  and  $y$  belong to the class of invertible and stationary ARMA processes, this dissimilarity measure is based on checking the equality of their underlying ARMA models by following the testing procedures proposed by Maharaj (1996,2000). The ARMA structures are approximated by truncated  $AR(\infty)$  models with a common order  $k = \max(k_x, k_y)$ , where  $k_x$  and  $k_y$  are determined by the AIC criterion. The AR coefficients are automatically fitted. The dissimilarity can be evaluated by using the value of the test statistic or alternatively the associated p-value. If dependence is FALSE, the dissimilarity measure is constructed by following the procedure introduced by Maharaj (1996), which is designed to compare independent time series. Otherwise, a more general testing procedure is used (Maharaj, 2000), which assumes that both models are correlated at the same time points but uncorrelated across observations (Maharaj, 2000). When permissive argument is TRUE, if the automatic fitting of the AR order fails, the method shows a warning and then forces an AR of order 1. If permissive is FALSE the method produces an error if no AR order is found by AIC.

**Value**

statistic	The statistic of the homogeneity test.
p_value	The p-value of the homogeneity test.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

- Maharaj, E.A. (1996) A significance test for classifying ARMA models. *J. Statist. Comput. Simulation*, **54(4)**, 305–331.
- Maharaj E.A. (2000) Clusters of time series. *J. Classification*, **17(2)**, 297–314.
- Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. *Journal of Statistical Software*, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.AR.PIC](#), [diss.AR.LPC](#), [CEPS](#)

**Examples**

```
## Create three sample time series
x <- arima.sim(model=list(ar=c(0.4,-0.1)), n =100, n.start=100)
y <- arima.sim(model=list(ar=c(0.9)), n =100, n.start=100)
z <- arima.sim(model=list(ar=c(0.5, 0.2)), n =100, n.start=100)
## Compute the distance and check for coherent results
diss.AR.MAH(x, y)
diss.AR.MAH(x, z)
diss.AR.MAH(y, z)

#create a dist object for its use with clustering functions like pam or hclust
diss( rbind(x,y,z), "AR.MAH")$statistic
```

diss.AR.PIC

*Model-based Dissimilarity Measure Proposed by Piccolo (1990)***Description**

Computes the distance between two time series as the Euclidean distance between the truncated AR operators approximating their ARMA structures.

**Usage**

```
diss.AR.PIC(x, y, order.x=NULL, order.y=NULL, permissive=TRUE)
```

**Arguments**

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
order.x	Specifies the ARIMA model to be fitted for the series x. When using diss wrapper, use order argument instead. See details.
order.y	Specifies the ARIMA model to be fitted for the series y. When using diss wrapper, use order argument instead. See details.
permissive	Specifies whether to force an AR order of 1 if no order is found. Ignored if neither order.x or order.y are NULL

**Details**

If order.x or order.y are NULL, their respective series will be fitted automatically using a AR model. If permissive is TRUE and no AR order is found automatically, an AR order of 1 will be imposed, if this case fails, then no order can be found and the function produces an error. order.x and order.y contain the three components of the ARIMA model: the AR order, the degree of differencing and the MA order, specified as in the function [arima](#).

If using diss function with "AR.PIC" method, the argument order must be used instead of order.x and order.y. orders is a matrix with one row per ARIMA, specified as in [arima](#). If order is NULL, automatic fitting imposing a AR model is performed.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Piccolo, D. (1990) A distance measure for classifying arima models. *J. Time Series Anal.*, **11(2)**, 153–164.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. *Journal of Statistical Software*, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.AR.MAH](#), [diss.AR.LPC.CEPS](#), [diss](#), [arima](#), [ar](#)

**Examples**

```
## Create three sample time series
x <- arima.sim(model=list(ar=c(0.4,-0.1)), n =100, n.start=100)
y <- arima.sim(model=list(ar=c(0.9)), n =100, n.start=100)
z <- arima.sim(model=list(ar=c(0.5, 0.2)), n =100, n.start=100)
## Compute the distance and check for coherent results
#ARIMA(2,0,0) for x and ARIMA(1,0,0) for y
diss.AR.PIC( x, y, order.x = c(2,0,0), order.y = c(1,0,0) )
diss.AR.PIC( x, z, order.x = c(2,0,0), order.y = c(2,0,0) )
# AR for y (automatically selected) and ARIMA(2,0,0) for z
diss.AR.PIC( y, z, order.x=NULL, order.y=c(2,0,0) )
#create a dist object for its use with clustering functions like pam or hclust
diss( rbind(x,y,z), METHOD="AR.PIC", order=rbind(c(2,0,0), c(1,0,0), c(2,0,0)) )
```

---

diss.CDM

*Compression-based Dissimilarity measure*


---

**Description**

Computes the dissimilarity based on the sizes of the compressed time series.

**Usage**

```
diss.CDM(x, y, type = "min")
```

**Arguments**

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
type	Character string, the type of compression. May be abbreviated to a single letter, defaults to the first of the alternatives.

**Details**

The compression based dissimilarity is calculated:

$$d(x, y) = C(xy)/(C(x) + C(y))$$

where  $C(x)$ ,  $C(y)$  are the sizes in bytes of the compressed series  $x$  and  $y$ .  $C(xy)$  is the size in bytes of the series  $x$  and  $y$  concatenated. The algorithm used for compressing the series is chosen with `type`. `type` can be "gzip", "bzip2" or "xz", see `memCompress`. "min" selects the best separately for  $x$ ,  $y$  and the concatenation. Since the compression methods are character-based, a symbolic representation can be used, see details for an example using SAX as the symbolic representation. The series are transformed to a text representation prior to compression using `as.character`, so small numeric differences may produce significantly different text representations. While this dissimilarity is asymptotically symmetric, for short series the differences between `diss.CDM(x, y)` and `diss.CDM(y, x)` may be noticeable.

**Value**

The computed dissimilarity.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Keogh, E., Lonardi, S., & Ratanamahatana, C. A. (2004). Towards parameter-free data mining. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 206-215).

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

`memCompress`, `diss`, `diss.NCD`, `PAA`, `convert.to.SAX.symbol`

**Examples**

```
n = 50
x <- rnorm(n) #generate sample series, white noise and a wiener process
y <- cumsum(rnorm(n))

diss.CDM(x, y)
```

```

z <- rnorm(n)
w <- cumsum(rnorm(n))
series = rbind(x, y, z, w)
diss(series, "CDM", type="bzip2")

#####
####symbolic representation prior to compression, using SAX####
####simpler symbolization, such as round() could also be used####
#####
#normalization function, required for SAX
z.normalize = function(x) {
  (x - mean(x)) / sd(x)
}

sx <- convert.to.SAX.symbol( z.normalize(x), alpha=4 )
sy <- convert.to.SAX.symbol( z.normalize(y), alpha=4 )
sz <- convert.to.SAX.symbol( z.normalize(z), alpha=4 )
sw <- convert.to.SAX.symbol( z.normalize(w), alpha=4 )

diss(rbind(sx, sy, sz, sw), "CDM", type="bzip2")

```

---

diss.CID

*Complexity-Invariant Distance Measure For Time Series*


---

### Description

Computes the distance based on the Euclidean distance corrected by the complexity estimation of the series.

### Usage

```
diss.CID(x, y)
```

### Arguments

x                    Numeric vector containing the first of the two time series.  
y                    Numeric vector containing the second of the two time series.

### Details

This distance is defined

$$CID(x, y) = ED(x, y) \times CF(x, y)$$

where  $CF(x, y)$  is a complexity correction factor defined as:

$$\max(CE(x), CE(y)) / \min(CE(x), CE(y))$$

and  $CE(x)$  is a complexity estimate of a time series  $x$ . `diss.CID` therefore increases the distance between series with different complexities. If the series have the same complexity estimate, the distance defenerates Euclidean distance. The complexity is defined in `diss.CID` as:

$$CE(x) = \sqrt{\sum_{t=1} (x_{t+1} - x_t)^2}$$

### Value

The computed dissimilarity.

### Author(s)

Pablo Montero Manso, José Antonio Vilar.

### References

Batista, G. E., Wang, X., & Keogh, E. J. (2011). A Complexity-Invariant Distance Measure for Time Series. In *SDM* (Vol. 31, p. 32).

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. *Journal of Statistical Software*, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

### See Also

[diss](#), [diss.CORT](#)

### Examples

```
n = 100
x <- rnorm(n) #generate sample series, white noise and a wiener process
y <- cumsum(rnorm(n))

diss.CID(x, y)

z <- rnorm(n)
w <- cumsum(rnorm(n))
series = rbind(x, y, z, w)
diss(series, "CID")
```

---

diss.COR

*Correlation-based Dissimilarity*

---

### Description

Computes dissimilarities based on the estimated Pearson's correlation of two given time series.

**Usage**

```
diss.COR(x, y, beta = NULL)
```

**Arguments**

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
beta	If not NULL, specifies the regulation of the convergence in the second method.

**Details**

Two different measures of dissimilarity between two time series based on the estimated Pearson's correlation can be computed. If beta is not specified, the value  $d_1 = \sqrt{2(1-\rho)}$  is computed, where  $(\rho)$  denotes the Pearson's correlation between series x and y. If beta is specified, the function  $d_2 = \sqrt{\left(\frac{1-\rho}{1+\rho}\right)^\beta}$  is used, where  $\beta$  is beta.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Golay, X., Kollias, S., Stoll, G., Meier, D., Valavanis, A., and Boesiger, P. (2005) A new correlation-based fuzzy logic clustering algorithm for FMRI. *Magnetic Resonance in Medicine*, **40.2**, 249–260.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.PACF](#), [diss.ACF](#), [diss](#)

**Examples**

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
z <- sin(seq(0, pi, length.out=100))
## Compute the distance and check for coherent results
diss.COR(x, y)
diss.COR(x, z)
#create a dist object for its use with clustering functions like pam or hclust
## Not run:
diss( rbind(x,y,z), "COR")

## End(Not run)
```

---

diss.CORT	<i>Dissimilarity Index Combining Temporal Correlation and Raw Values Behaviors</i>
-----------	--

---

### Description

Computes an adaptive dissimilarity index between two time series that covers both dissimilarity on raw values and dissimilarity on temporal correlation behaviors.

### Usage

```
diss.CORT(x, y, k = 2, deltamethod="Euclid")
```

### Arguments

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
k	Parameter controlling the weight of the dissimilarity between dynamic behaviors (See Details).
deltamethod	Defines the method for the raw data discrepancy. Either "Euclid", "Frechet" or "DTW".

### Details

The dissimilarity between time series x and y is given by:

$$d(x, y) = \Phi[CORT(x, y)]\delta(x, y)$$

where:

CORT(x,y) measures the proximity between the dynamic behaviors of x and y by means of the first order temporal correlation coefficient defined by:

$$CORT(x, y) = \frac{\sum_{t=1} (x_{t+1} - x_t)(y_{t+1} - y_t)}{\sqrt{\sum_{t=1} (x_{t+1} - x_t)^2} \sqrt{\sum_{t=1} (y_{t+1} - y_t)^2}}$$

$\Phi[u]$  is an adaptive tuning function taking the form:

$$\frac{2}{1 + e^{ku}}$$

with  $k \geq 0$  so that both  $\Phi$  and k modulate the weight that CORT(x,y) has on d(x,y).

$\delta(x, y)$  denotes a dissimilarity measure between the raw values of series x and y, such as the Euclidean distance, the Frechet distance or the Dynamic Time Warping distance. Note that  $d(x, y) = \delta(x, y)$  if  $k=0$ .

More details of the procedure can be seen in Chouakria-Douzal and Nagabhushan (2007).

deltamethod ( $\delta$ ) can be either Euclidean (deltamethod = "Euclid"), Frechet (deltamethod = "Frechet") or Dynamic Time Warping (deltamethod = "DTW") distances. When calling from dis.CORT, DTW uses Manhattan as local distance.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Chouakria-Douzal, A. and Nagabhushan P. N. (2007) Adaptive dissimilarity index for measuring time series proximity. *Adv. Data Anal. Classif.*, **1(1)**, 5–21.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.COR](#), [diss.DTWARP](#), [diss.FRECHET](#), [distFrechet](#), [dtw](#).

**Examples**

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
z <- sin(seq(0, pi, length.out=100))
## Compute the distance and check for coherent results
diss.CORT(x, y, 2)
diss.CORT(x, z, 2)
diss.CORT(y, z, 2)
#create a dist object for its use with clustering functions like pam or hclust
## Not run:
diss( rbind(x,y,z), "CORT", k=3, deltamethod="DTW")

## End(Not run)
```

---

diss.DTWARP

*Dynamic Time Warping Distance*

---

**Description**

Computes Dynamic Time Warping distance between time series.

**Usage**

```
diss.DTWARP(x, y, ...)
```

**Arguments**

x                    Numeric vector containing the first of the two time series.  
y                    Numeric vector containing the second of the two time series.  
...                  Additional parameters for the function. See `link[dtw]{dtw}`.

**Details**

This is basically a wrapper for `dtw` of the `pdcc` package, intended for an easier discovery of the functionalities used in `TSclust`.

**Value**

The computed distance.

**See Also**

`diss`, `link[dtw]{dtw}`

**Examples**

```
## Create three sample time series  
x <- cumsum(rnorm(100))  
y <- cumsum(rnorm(100))  
diss.DTWARP(x, y)
```

---

diss.DWT

*Dissimilarity for Time Series Based on Wavelet Feature Extraction*

---

**Description**

Performs an unsupervised feature extraction using orthogonal wavelets on the series and returns the Euclidean distance between the wavelet approximations in an appropriate scale.

**Usage**

```
diss.DWT(series)
```

**Arguments**

series              Numeric matrix with row order time series

## Details

This method differs from other dissimilarities in that pairwise dissimilarities depend on the whole dataset that is given to `diss.DWT`, hence, there is no pairwise version of the function defined, only accepts whole datasets. The set of original series is replaced by their wavelet approximation coefficients in an appropriate scale, and the dissimilarity between two series is computed as the Euclidean distance between these coefficients. The appropriate scale is automatically determined by using an algorithm addressed to obtain an efficient reduction of the dimensionality but preserving as much information from the original data as possible. The algorithm is introduced by Zhang, Ho, Zhang, and Lin (2006).

## Value

Returns an object of type `dist` with the pairwise distances.

## Author(s)

Pablo Montero Manso, José Antonio Vilar.

## References

Zhang, H., Ho, T. B., Zhang, Y., and Lin, M. (2006) Unsupervised feature extraction for time series clustering using orthogonal wavelet transform. *INFORMATICA-LJUBLJANA*-, **30(3)**, 305.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

## See Also

[wavDWT](#)

## Examples

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
z <- sin(seq(0, pi, length.out=100))

#compute the distance
diss.DWT(rbind(x, y, z))
```

---

diss.EUCL

*Euclidean Distance*

---

## Description

Computes Euclidean distance between time series.

**Usage**

```
diss.EUCL(x, y)
```

**Arguments**

x                    Numeric vector containing the first of the two time series.  
y                    Numeric vector containing the second of the two time series.

**Value**

The computed distance.

**See Also**

[diss](#), [dist](#)

**Examples**

```
## Create three sample time series  
x <- cumsum(rnorm(100))  
y <- cumsum(rnorm(100))  
diss.EUCL(x, y)
```

---

diss.FRECHET	<i>Frechet Distance</i>
--------------	-------------------------

---

**Description**

Computes the Frechet distance between time series.

**Usage**

```
diss.FRECHET(x, y, ...)
```

**Arguments**

x                    Numeric vector containing the first of the two time series.  
y                    Numeric vector containing the second of the two time series.  
...                  Additional parameters for the function. See [link\[longitudinalData\]{distFrechet}](#).

**Details**

This is basically a wrapper for `distFrechet` of the [longitudinalData](#) package, intended for an easier discovery of the functionalities used in `TSclust`.

**Value**

The computed distance.

**See Also**

`diss`, `link[longitudinalData]{distFrechet}`

**Examples**

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
diss.FRECHET(x, y)
```

---

diss.INT.PER

*Integrated Periodogram Based Dissimilarity*


---

**Description**

Computes the dissimilarity between two time series in terms of the distance between their integrated periodograms.

**Usage**

```
diss.INT.PER(x, y, normalize=TRUE)
```

**Arguments**

`x` Numeric vector containing the first of the two time series.  
`y` Numeric vector containing the second of the two time series.  
`normalize` If TRUE the normalized version is computed.

**Details**

The distance is computed as:

$$d(x, y) = \int_{-\pi}^{\pi} |F_x(\lambda) - F_y(\lambda)| d\lambda,$$

where  $F_x(\lambda_j) = C_x^{-1} \sum_{i=1}^j I_x(\lambda_i)$  and  $F_y(\lambda_j) = C_y^{-1} \sum_{i=1}^j I_y(\lambda_i)$ , with  $C_x = \sum_i I_x(\lambda_i)$  and  $C_y = \sum_i I_y(\lambda_i)$  in the normalized version.  $C_x = 1$  and  $C_y = 1$  in the non-normalized version.  $I_x(\lambda_k)$  and  $I_y(\lambda_k)$  denote the periodograms of `x` and `y`, respectively.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

## References

Casado de Lucas, D. (2010) Classification techniques for time series and functional data.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

## See Also

[diss.PER](#)

## Examples

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
z <- sin(seq(0, pi, length.out=100))
## Compute the distance and check for coherent results
diss.INT.PER(x, y, normalize=TRUE)
diss.INT.PER(x, y, normalize=TRUE)
diss.INT.PER(x, y, normalize=TRUE)
## Not run:
diss( rbind(x,y,z), "INT.PER", normalize=FALSE )

## End(Not run)
```

---

diss.MINDIST.SAX

*Symbolic Aggregate Aproximation related functions*

---

## Description

diss.MINDIST.SAX computes a dissimilarity that lower bounds the Euclidean on the discretized, dimensionality reduced series. Function PAA produces the dimension reduction. Function `convert.to.SAX.symbol` produces the discretization.

## Usage

```
diss.MINDIST.SAX(x, y, w, alpha=4, plot=FALSE)
PAA(x, w)
convert.to.SAX.symbol(x, alpha)
MINDIST.SAX(x, y, alpha, n)
SAX.plot(series, w, alpha, col.ser=rainbow(ncol(series)))
```

## Arguments

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
w	The amount of equal sized frames that the series will be reduced to.

alpha	The size of the alphabet, the amount of symbols used to represents the values of the series.
plot	If TRUE, plot a graphic of the reduced series, with their corresponding symbols.
n	The original size of the series.
series	A ts or mts object with the series to plot.
col.ser	Colors for the series. One per series.

### Details

SAX is a symbolic representation of continuous time series.

w must be an integer but it does not need to divide the length of the series. If w divides the length of the series, the `diss.MINDIST.SAX` plot uses this to show the size of the frames.

PAA performs the Piecewise Aggregate Approximation of the series, reducing it to w elements, called frames. Each frame is composed by  $n/w$  observations of the original series, averaged. Observations are weighted when w does not divide n.

`convert.to.SAX.symbol` performs SAX discretization: Discretizes the series x to an alphabet of size alpha, x should be z-normalized in this case. The  $N(0, 1)$  distribution is divided in alpha equal probability parts, if an observation falls into the *i*th part (starting from minus infinity), it is assigned the *i* symbol.

MINDIST.SAX calculates the MINDIST dissimilarity between symbolic representations.

`diss.MINDIST.SAX` combines the previous procedures to compute a dissimilarity between series. The series are z-normalized at first. Then the dimensionality is reduced usin PAA to produce series of length w. The series are discretized to an alphabet of size alpha using `convert.to.SAX.symbol`. Finally the dissimilarity value is produced using MINDIST.SAX.

`SAX.plot` produces a plot of the SAX representation of the given series.

### Value

The computed dissimilarity.

### Author(s)

Pablo Montero Manso, José Antonio Vilar.

### References

Lin, J., Keogh, E., Lonardi, S. & Chiu, B. (2003) A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.

Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, **3(3)**, 263-286.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**[diss](#)**Examples**

```

set.seed(12349)
n = 100
x <- rnorm(n) #generate sample series, white noise and a wiener process
y <- cumsum(rnorm(n))
w <- 20 #amount of equal-sized frames to divide the series, parameters for PAA
alpha <- 4 #size of the alphabet, parameter for SAX

#normalize
x <- (x - mean(x)) /sd(x)
y <- (y - mean(y)) /sd(y)

paax <- PAA(x, w) #generate PAA reductions
paay <- PAA(y, w)

plot(x, type="l", main="PAA reduction of series x") #plot an example of PAA reduction
p <- rep(paax,each=length(x)/length(paax)) #just for plotting the PAA
lines(p, col="red")

#repeat the example with y
plot(y, type="l", main="PAA reduction of series y")
py <- rep(paay,each=length(y)/length(paay))
lines(py, col="blue")

#convert to SAX representation
SAXx <- convert.to.SAX.symbol( paax, alpha)
SAXy <- convert.to.SAX.symbol( paay, alpha)

#CALC THE SAX DISTANCE
MINDIST.SAX(SAXx, SAXy, alpha, n)

#this whole process can be computed using diss.MINDIST.SAX
diss.MINDIST.SAX(x, y, w, alpha, plot=TRUE)

z <- rnorm(n)^2

diss(rbind(x,y,z), "MINDIST.SAX", w, alpha)

SAX.plot( as.ts(cbind(x,y,z)), w=w, alpha=alpha)

```

**Description**

Computes the distance based on the sizes of the compressed time series.

**Usage**

```
diss.NCD(x, y, type = "min")
```

**Arguments**

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
type	Character string, the type of compression. May be abbreviated to a single letter, defaults to the first of the alternatives.

**Details**

The compression based dissimilarity is calculated:

$$d(x, y) = C(xy) - \max(C(x), C(y)) / \min(C(x), C(y))$$

where  $C(x)$ ,  $C(y)$  are the sizes in bytes of the compressed series  $x$  and  $y$ .  $C(xy)$  is the size in bytes of the series  $x$  and  $y$  concatenated. The algorithm used for compressing the series is chosen with `type`. `type` can be "gzip", "bzip2" or "xz", see `memCompress`. "min" selects the best separately for  $x$ ,  $y$  and the concatenation. Since the compression methods are character-based, a symbolic representation can be used, see details for an example using SAX as the symbolic representation. The series are transformed to a text representation prior to compression using `as.character`, so small numeric differences may produce significantly different text representations. While this dissimilarity is asymptotically symmetric, for short series the differences between `diss.NCD(x, y)` and `diss.NCD(y, x)` may be noticeable.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

- Cilibrasi, R., & Vitányi, P. M. (2005). Clustering by compression. *Information Theory, IEEE Transactions on*, **51**(4), 1523-1545.
- Keogh, E., Lonardi, S., & Ratanamahatana, C. A. (2004). Towards parameter-free data mining. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 206-215).
- Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. *Journal of Statistical Software*, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[memCompress](#), [diss](#)

**Examples**

```
n = 50
x <- rnorm(n) #generate sample series, white noise and a wiener process
y <- cumsum(rnorm(n))

diss.NCD(x, y)

z <- rnorm(n)
w <- cumsum(rnorm(n))
series = rbind(x, y, z, w)
diss(series, "NCD", type="bzip2")

#####
####symbolic representation prior to compression, using SAX####
####simpler symbolization, such as round() could also be used###
#####
#normalization function, required for SAX
z.normalize = function(x) {
  (x - mean(x)) / sd(x)
}

sx <- convert.to.SAX.symbol( z.normalize(x), alpha=4 )
sy <- convert.to.SAX.symbol( z.normalize(y), alpha=4 )
sz <- convert.to.SAX.symbol( z.normalize(z), alpha=4 )
sw <- convert.to.SAX.symbol( z.normalize(w), alpha=4 )

diss(rbind(sx, sy, sz, sw), "NCD", type="bzip2")
```

---

diss.PDC

*Permutation Distribution Distance*


---

**Description**

Computes the Permutation Distribution distance between time series.

**Usage**

```
diss.PDC(x, y, ...)
```

**Arguments**

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
...	Additional parameters for the function. See <code>link[pdc]{pdcDist}</code> .

**Details**

This is basically a wrapper for `pdDist` of the `pd` package, intended for an easier discovery of the functionalities used in `TSclust`.

**Value**

The computed distance.

**See Also**

`diss`, `link[pd]{pdDist}`

**Examples**

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
diss.PDC(x, y)
```

---

diss.PER

*Periodogram Based Dissimilarity*


---

**Description**

Computes the distance between two time series based on their periodograms.

**Usage**

```
diss.PER(x, y, logarithm=FALSE, normalize=FALSE)
```

**Arguments**

<code>x</code>	Numeric vector containing the first of the two time series.
<code>y</code>	Numeric vector containing the second of the two time series.
<code>logarithm</code>	Boolean. If TRUE logarithm of the periodogram coefficients will be taken.
<code>normalize</code>	Boolean. If TRUE, the periodograms will be normalized by the variance of their respective series.

**Details**

Computes the Euclidean distance between the periodogram coefficients of the series `x` and `y`. Additional transformations can be performed on the coefficients depending on the values of `logarithm` and `normalize`.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Caiado, J., Crato, N. and Peña, D. (2006) A periodogram-based metric for time series classification. *Comput. Statist. Data Anal.*, **50(10)**, 2668–2684.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. *Journal of Statistical Software*, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

link{diss.INT.PER}

**Examples**

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
z <- sin(seq(0, pi, length.out=100))
## Compute the distance and check for coherent results
diss.PER(x, y)
diss.PER(x, z)
diss.PER(y, z)
diss.PER(x, y, TRUE, TRUE)
diss.PER(x, z, TRUE, TRUE)
diss.PER(y, z, TRUE, TRUE)
#create a dist object for its use with clustering functions like pam or hclust
diss( rbind(x,y,z), "PER", logarithm=TRUE, normalize=TRUE)
```

---

diss.PRED

---

*Dissimilarity Measure Based on Nonparametric Forecast*


---

**Description**

Computes the dissimilarity between two time series as the L1 distance between the kernel estimators of their forecast densities at a pre-specified horizon.

**Usage**

```
diss.PRED(x, y, h, B=500, logarithm.x=FALSE, logarithm.y=FALSE,
differences.x=0, differences.y=0, plot=FALSE)
```

**Arguments**

<code>x</code>	Numeric vector containing the first of the two time series.
<code>y</code>	Numeric vector containing the second of the two time series.
<code>h</code>	The horizon of interest, i.e the number of steps-ahead where the prediction is evaluated.
<code>B</code>	The amount of bootstrap resamples.
<code>logarithm.x</code>	Boolean. Specifies whether to transform series <code>x</code> by taking logarithms or not. When using <code>diss</code> wrapper, use <code>logarithms</code> argument instead. See details.
<code>logarithm.y</code>	Boolean. Specifies whether to transform series <code>y</code> by taking logarithms or not. When using <code>diss</code> wrapper, use <code>logarithms</code> argument instead. See details.
<code>differences.x</code>	Specifies the amount of differences to apply to series <code>x</code> . When using <code>diss</code> wrapper, use <code>differences</code> argument instead. See details.
<code>differences.y</code>	Specifies the amount of differences to apply to series <code>y</code> . When using <code>diss</code> wrapper, use <code>differences</code> argument instead. See details.
<code>plot</code>	If TRUE, plot the resulting forecast densities.

**Details**

The dissimilarity between the time series `x` and `y` is given by

$$d(x, y) = \int |f_{x,h}(u) - f_{y,h}(u)| du$$

where  $f_{x,h}$  and  $f_{y,h}$  are kernel density estimators of the forecast densities  $h$ -steps ahead of `x` and `y`, respectively. The horizon of interest `h` is pre-specified by the user. The kernel density estimators are based on `B` bootstrap replicates obtained by using a resampling procedure that mimics the generating processes, which are assumed to follow an arbitrary autoregressive structure (parametric or non-parametric). The procedure is completely detailed in Vilar et al. (2010). This function has high computational cost due to the bootstrapping procedure.

The procedure uses a bootstrap method that requires stationary time series. In order to support a wider range of time series, the method allows some transformations on the series before proceeding with the bootstrap resampling. This transformations are inverted before calculating the densities. The transformations allowed are logarithm and differenciation. The parameters `logarithm.x`, `logarithm.y`, `differences.x`, `differences.y` can be specified with this purpose.

If using `diss` function with "PRED" method, the argument `logarithms` must be used instead of `logarithm.x` and `logarithm.y`. `logarithms` is a boolean vector specifying if the logarithm transform should be taken for each one of the series. The argument `differences`, a numeric vector specifying the amount of differences to apply the series, is used instead of `differences.x` and `differences.y`. The plot is also different, showing all the densities in the same plot.

**Value**

`diss.PRED` returns a list with the following components.

<code>L1dist</code>	The computed distance.
---------------------	------------------------

dens.x	A 2-column matrix with the density of prediction of series x. First column is the base (x) and the second column is the value (y) of the density.
dens.y	A 2-column matrix with the density of prediction of series y. First column is the base (x) and the second column is the value (y) of the density.

When used from the `diss` wrapper function, it returns a list with the following components.

<code>dist</code>	A <code>dist</code> object with the pairwise L1 distances between series.
<code>densities</code>	A list of 2-column matrices containing the densities of each series, in the same format as 'dens.x' or 'dens.y' of <code>diss.PRED</code> .

### Author(s)

José Antonio Vilar, Pablo Montero Manso.

### References

Alonso, A.M., Berrendero, J.R., Hernandez, A. and Justel, A. (2006) Time series clustering based on forecast densities. *Comput. Statist. Data Anal.*, **51**,762–776.

Vilar, J.A., Alonso, A. M. and Vilar, J.M. (2010) Non-linear time series clustering based on non-parametric forecast densities. *Comput. Statist. Data Anal.*, **54** (11), 2850–2865.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. *Journal of Statistical Software*, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

### See Also

[diss](#)

### Examples

```
x <- (rnorm(100))
x <- x + abs(min(x)) + 1 #shift to produce values greater than 0, for a correct logarithm transform
y <- (rnorm(100))
z <- sin(seq(0, pi, length.out=100))
## Compute the distance and check for coherent results
diss.PRED(x, y, h=6, logarithm.x=FALSE, logarithm.y=FALSE, differences.x=1, differences.y=0)
#create a dist object for its use with clustering functions like pam or hclust
diss( rbind(x,y,z), METHOD="PRED", h=3, B=200,
      logarithms=c(TRUE,FALSE, FALSE), differences=c(1,1,2) )
```

diss.SPEC.GLK

*Dissimilarity based on the Generalized Likelihood Ratio Test***Description**

The dissimilarity between two time series is computed by using an adaptation of the generalized likelihood ratio test to check the equality of two log-spectra.

**Usage**

```
diss.SPEC.GLK(x, y, plot=FALSE)
```

**Arguments**

`x` Numeric vector containing the first of the two time series.  
`y` Numeric vector containing the second of the two time series.  
`plot` If TRUE, the smoothed spectral densities of the two series are plotted.

**Details**

The dissimilarity between two series `x` and `y` is measured in terms of the value of a test statistic to check the equality of their log-spectra,  $m_X(\lambda)$  and  $m_Y(\lambda)$  respectively. The test statistic is constructed by using the generalized likelihood ratio test criterion (Fan and Zhang, 2004). Specifically, the test statistic takes the form:

$$d(x, y) = \sum_{k=1}^T [Z_k - \hat{\mu}(\lambda_k) - 2 \log(1 + e^{\{Z_k - \hat{\mu}(\lambda_k)\}})] - \sum_{k=1}^T [Z_k - 2 \log(1 + e^{Z_k})],$$

where  $I_x(\lambda_k)$  and  $I_y(\lambda_k)$  are the periodograms of `x` and `y`,  $Z_k = \log(I_x(\lambda_k)) - \log(I_y(\lambda_k))$ , and  $\hat{\mu}(\lambda_k)$  is the local maximum log-likelihood estimator of  $\mu(\lambda_k) = m_x(\lambda_k) - m_y(\lambda_k)$  computed by local linear fitting.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Fan, J. and Zhang, W. (2004) Generalised likelihood ratio tests for spectral density. *Biometrika*, 195–209.

Pértega, S. and Vilar, J.A. (2010) Comparing several parametric and nonparametric approaches to time series clustering: A simulation study. *J. Classification*, **27(3)**, 333–362.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

### See Also

[diss.SPEC.ISD](#), [diss.SPEC.LLR](#)

### Examples

```
## Create two sample time series
x <- cumsum(rnorm(50))
y <- cumsum(rnorm(50))
z <- sin(seq(0, pi, length.out=50))
## Compute the distance and check for coherent results
diss.SPEC.GLK(x, y, plot=TRUE)
#create a dist object for its use with clustering functions like pam or hclust
## Not run:
diss( rbind(x,y,z), "SPEC.GLK" )

## End(Not run)
```

---

diss.SPEC.ISD	<i>Dissimilarity Based on the Integrated Squared Difference between the Log-Spectra</i>
---------------	---

---

### Description

Computes the dissimilarity between two time series in terms of the integrated squared difference between non-parametric estimators of their log-spectra.

### Usage

```
diss.SPEC.ISD(x, y, plot=FALSE, n=length(x))
```

### Arguments

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
plot	If TRUE, plot the smoothed spectral densities of the two series.
n	The number of points to use for the linear interpolation. A value of n=0 uses numerical integration instead of linear interpolation. See details.

**Details**

$$d(x, y) = \int (\hat{m}_x(\lambda) - \hat{m}_y(\lambda))^2 d\lambda,$$

where  $\hat{m}_x(\lambda)$  and  $\hat{m}_y(\lambda)$  are local linear smoothers of the log-periodograms, obtained using the maximum local likelihood criterion.

By default, for performance reasons, the spectral densities are estimated using linear interpolation using  $n$  points. If  $n$  is 0, no linear interpolation is performed, and `integrate` is used to calculate the integral, using as many points as `integrate` sees fit.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Pértega, S. and Vilar, J.A. (2010) Comparing several parametric and nonparametric approaches to time series clustering: A simulation study. *J. Classification*, **27(3)**, 333–362.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.SPEC.GLK](#), [diss.SPEC.LLR](#)

**Examples**

```
## Create two sample time series
x <- cumsum(rnorm(50))
y <- cumsum(rnorm(50))
z <- sin(seq(0, pi, length.out=50))
## Compute the distance and check for coherent results
diss.SPEC.ISD(x, y, plot=TRUE)
#create a dist object for its use with clustering functions like pam or hclust
## Not run:
diss.SPEC.ISD(x, y, plot=TRUE, n=0)#try integrate instead of interpolation
diss( rbind(x,y,z), "SPEC.ISD" )

## End(Not run)
```

---

diss.SPEC.LLR	<i>General Spectral Dissimilarity Measure Using Local-Linear Estimation of the Log-Spectra</i>
---------------	--

---

### Description

Computes a general dissimilarity measure based on the ratio of local linear spectral estimators.

### Usage

```
diss.SPEC.LLR(x, y, alpha=0.5, method="DLS", plot=FALSE, n=length(x))
```

### Arguments

x	Numeric vector containing the first of the two time series.
y	Numeric vector containing the second of the two time series.
alpha	Power for the ratio of densities in the Chernoff information measure. Between 0 and 1.
method	"DLS" for least squares estimation of the spectral density and "LK" for maximum likelihood estimation.
plot	if TRUE, plot the smoothed spectral densities of the two series.
n	The number of points to use for the linear interpolation. A value of n=0 uses numerical integration instead of linear interpolation. See details.

### Details

$$d_W = \int_{-\pi}^{\pi} W' \left( \frac{f_x(\lambda)}{f_y(\lambda)} \right) d\lambda$$

where:

- $f_x$  and  $f_y$  are nonparametric approximations of spectral densities of x and y respectively.
- $W'(x) = W(x) + W(1/x)$  with  $W(x) = \log(\alpha x + (1 - \alpha)x) - \alpha \log(x)$ , so that  $W(\cdot)$  is a divergence function depending on  $\alpha$ .

This dissimilarity measure corresponds to the limiting spectral approximation of the Chernoff information measure in the time domain (see Kakizawa et al., 1998). The spectral densities are approximated by using local linear fitting by generalized least squared if method="DLS" or by maximum likelihood if method="LK" (in this case, higher computational cost is required).

By default, for performance reasons, the spectral densities are estimated using linear interpolation using n points. If n is 0, no linear interpolation is performed, and integrate is used to calculate the integral, using as many points as integrate sees fit. If the dissimilarity will be calculated for more than two series, calling SPEC.LLR from the diss wrapper function is preferred, since it saves some computations.

**Value**

The computed distance.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Vilar, J.A. and Pértega, S. (2004) Discriminant and cluster analysis for gaussian stationary processes: local linear fitting approach. *J. Nonparametr. Stat.*, **16(3-4)** 443–462.

Kakizawa, Y., Shumway, R. H. and Taniguchi M. (1998) Discrimination and clustering for multivariate time series. *J. Amer. Statist. Assoc.*, **93(441)**, 328– 340.

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.SPEC.GLK](#), [diss.SPEC.ISD](#), [integrate](#)

**Examples**

```
## Create three sample time series
x <- cumsum(rnorm(50))
y <- cumsum(rnorm(50))
z <- sin(seq(0, pi, length.out=50))
## Compute the distance and check for coherent results
diss.SPEC.LLR(x, y, plot=TRUE)
diss.SPEC.LLR(x, z, n=0) #try integrate instead of interpolation
diss.SPEC.LLR(y, z, method="LK", n=0) #maximum likelihood with integration
#create a dist object for its use with clustering functions like pam or hclust
diss(rbind(x,y,z), METHOD="SPEC.LLR", method="DLS", alpha=0.5, n=50)
```

---

electricity

*Hourly Electricity Prices in the Spanish Market*

---

**Description**

Partial realizations of time series of hourly electricity prices (Cent/kWh) in the Spanish market.

**Usage**

```
data(electricity)
```

**Format**

A matrix with 365 observations of the hourly electricity cost readings.

**Details**

The dataset consists of hourly electricity prices in the Spanish market during weekdays of the period December 31, 2007 - May 25, 2009. Available at <http://www.ome1.es>.

**References**

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

---

interest.rates

*Long-Term Interest Rates from 1995 to 2012*

---

**Description**

Partial realizations of time series of long-term interest rates (10-year bonds) for several countries.

**Usage**

```
data(interest.rates)
```

**Format**

A ts object with 215 observations of the monthly long-term interest rates (10-year bonds) from January 1995 to November 2012 of several countries.

**References**

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

---

loo1nn.cv

*Clustering Evaluation Index Based on Leave-one-out One-nearest-neighbor Evaluation*

---

**Description**

Computes the leave-one-out one-nearest-neighbor cross-validation of an arbitrary distance matrix.

**Usage**

```
loo1nn.cv(d, G)
```

**Arguments**

- d                    A dist object.
- G                    Integer vector with the labels of the true cluster solution. Each element of the vector specifies the cluster 'id' that the element belongs to.

**Details**

Computes the proportion of successful clusters that the given distance matrix produces using leave-one-out one-nearest-neighbor cross-validation. Distance ties are solved by majority vote. A tie while voting produces a warning and is solved by selecting a candidate cluster at random.

**Value**

The computed proportion.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**See Also**

[cluster.evaluation](#), [loo1nn](#), [knn.cv](#),

**Examples**

```
data(synthetic.tseries)

#create the ground thruth cluster
G <- rep(1:6, each = 3)

#obtain candidate distance matrix (dist object)
dACF <- diss(synthetic.tseries, "ACF")

#calculate the cross-validation
loo1nn.cv(dACF, G)
```

---

paired.tseries

*Pairs of Time Series from Different Domains*

---

**Description**

Dataset formed by pairs of time series from different domains. Series were selected from the UCR Time Series Archive.

**Usage**

```
data(paired.tseries)
```

**Format**

A mts object with 36 series of length 1000.

**Details**

Each pair of series in the dataset (Series 1 and 2, Series 3 and 4, etc.) comes from the same domain, so this pairing could constitute a possible ground truth solution.

**Note**

abbreviate can be used on the colnames.

**Source**

[http://www.cs.ucr.edu/~eamonn/SIGKDD2004/All\\_datasets/](http://www.cs.ucr.edu/~eamonn/SIGKDD2004/All_datasets/)

**References**

Keogh, E., Lonardi, S., & Ratanamahatana, C. A. (2004). Towards parameter-free data mining. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 206-215).

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**Examples**

```
data(paired.tseries)
#Create the true solution, the pairs
true_cluster <- rep(1:18, each=2)
#test a dissimilarity metric and a cluster algorithm
intperdist <- diss(paired.tseries, "INT.PER") #create the distance matrix
#use hierarchical clustering and divide the tree in 18 clusters
intperclust <- cutree(hclust(intperdist), k=18 )
#use a cluster similarity index to rate the solution
cluster.evaluation(true_cluster, intperclust)

#### other evaluation criterion used in this dataset consist in counting the correct pairs
#### formed during agglomerative hierarchical cluster (see references)
true_pairs = (-matrix(1:36, ncol=2, byrow=TRUE))
hcintper <- hclust(intperdist, "complete")
#count within the hierarchical cluster the pairs
sum(match(data.frame(t(true_pairs)), data.frame(t(hcintper$merge)), nomatch=0) > 0) / 18
```

---

`pvalues.clust`*Clustering Algorithm Based on p-values.*

---

**Description**

Clustering algorithm based on p-values. Each group in the cluster solution is formed by series with associated p-values greater than a pre-specified level of significance.

**Usage**

```
pvalues.clust(pvalues, significance)
```

**Arguments**

<code>pvalues</code>	A dist object containing the p-values from testing the equality of each pair of time series under study.
<code>significance</code>	The significance level.

**Details**

Each element (i,j) in `pvalues` corresponds to the p-value obtained from checking whether or not the *i*-th and *j*-th series come from the same generating model. The clustering algorithm will only group together those series whose associated p-values are greater than the pre-specified significance level. The algorithm was originally developed for its use with the p-values obtained with `diss.AR.MAH` (see Maharaj, 2000), but it can be applied to any similar test.

**Value**

An integer vector of length *n*, the number of observations, giving for each observation the number (id) of the cluster to which it belongs.

**Author(s)**

Pablo Montero Manso, José Antonio Vilar.

**References**

Maharaj E.A. (2000) Clusters of time series. *J. Classification*, **17**(2), 297–314.  
Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**See Also**

[diss.AR.MAH](#)

**Examples**

```
## Create three sample time series
x <- cumsum(rnorm(100))
y <- cumsum(rnorm(100))
z <- sin(seq(0, pi, length.out=100))
##

## Compute the distance and check for coherent results
dd <- diss( rbind(x,y,z), "AR.MAH")
pvalues.clust( dd$p_value, 0.05 )
```

synthetic.tseries

*Synthetic Time Series for Clustering Performace Comparisons.***Description**

This dataset features three repetitions of several models of time series.

**Usage**

```
data(synthetic.tseries)
```

**Details**

The dataset is a mts object, formed by several repetitions of each of the following models.

M1	AR	$X_t = 0.6X_{t-1} + \varepsilon_t$
M2	Bilinear	$X_t = (0.3 - 0.2\varepsilon_{t-1}) X_{t-1} + 1.0 + \varepsilon_t$
M3	EXPAR	$X_t = (0.9 \exp(-X_{t-1}^2) - 0.6) X_{t-1} + 1.0 + \varepsilon_t$
M4	SETAR	$X_t = (0.3X_{t-1} + 1.0) I(X_{t-1} \geq 0.2) - (0.3X_{t-1} - 1.0) I(X_{t-1} < 0.2) + \varepsilon_t$
M5	NLAR	$X_t = 0.7  X_{t-1}  (2 +  X_{t-1} )^{-1} + \varepsilon_t$
M6	STAR	$X_t = 0.8X_{t-1} - 0.8X_{t-1} (1 + \exp(-10X_{t-1}))^{-1} + \varepsilon_t$

Three simulations of each model are included. This dataset can be used for comparing the performance of different dissimilarity measures between time series or clustering algorithms.

**References**

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

**Examples**

```
data(synthetic.tseries)
```

```

#Create the true solution, for this dataset, there are three series of each model
true_cluster <- c(1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6)
#test a dissimilarity metric and a cluster algorithm
intperdist <- diss( synthetic.tseries, "INT.PER") #create the distance matrix
#use hierarchical clustering and divide the tree in 6 clusters
intperclust <- cutree( hclust(intperdist), 6 )
#use a cluster similarity index to rate the solution
cluster.evaluation( true_cluster, intperclust)

#test another dissimilarity metric and a cluster algorithm
acfdist <- diss( synthetic.tseries, "ACF", p=0.05)
acfcluster <- pam( acfdist, 6 )$clustering #use pam clustering to form 6 clusters
cluster.evaluation( true_cluster, acfcluster)
## Not run:
#test another dissimilarity metric and a cluster algorithm
chernoffdist <- diss( synthetic.tseries, "SPEC.LLR")
chernoffclust <- pam( chernoffdist, 6 )$clustering
cluster.evaluation( true_cluster, chernoffclust)

## End(Not run)

```

---

TSclust

*Package for Time Series Clustering.*


---

## Description

This package contains several measures of dissimilarity between time series, some examples of time series datasets, specific clustering algorithms, and dimension reduction algorithms. Dissimilarities begin with `diss.*`, and a wrapper function `diss` is available. Cluster evaluation methods include `cluster.evaluation` and `loo1nn.cv`. A clustering algorithm based on pairwise p-values is implemented in `pvalues.clust`. The package should be used along with other existing clustering packages and function such as `hclust`, packages `cluster`, ...

## Author(s)

Pablo Montero Manso, José Antonio Vilar.

## References

Montero, P and Vilar, J.A. (2014) *TSclust: An R Package for Time Series Clustering*. Journal of Statistical Software, 62(1), 1-43. <http://www.jstatsoft.org/v62/i01/>.

## See Also

[diss](#)

**Examples**

```
#the available dissimilarities can be found in the diss help, page (?diss)
#and their individual pages from there.

### The most common use case begins with a set of time series we want to cluster.
### This package includes several example datasets.
###
data(interest.rates)
###transformation of the interest rates
trans.inter.rates <- log(interest.rates[2:215,]) - log(interest.rates[1:214,])

##use the dist function of the proxy package to easily create the dist object
#applying ACF with geometric decaying to each pair of time series
tsdist <- diss( t(trans.inter.rates) , "ACF", p=0.05)

names(tsdist) <- colnames(interest.rates)

#perform hierachical clustering to the dist object
hc <- hclust(tsdist)

#show the results
plot(hc)

mahdist <- diss( t(trans.inter.rates) , "AR.MAH", p=0.05)$p_value

pvalues.clust(mahdist, 0.05)
```

# Index

## \*Topic `\textasciitildekw1`

- `cluster.evaluation`, 2
- `diss`, 4
- `diss.ACF`, 6
- `diss.AR.LPC.CEPS`, 8
- `diss.AR.MAH`, 9
- `diss.AR.PIC`, 11
- `diss.CDM`, 12
- `diss.CID`, 14
- `diss.COR`, 15
- `diss.CORT`, 17
- `diss.DTWARP`, 18
- `diss.DWT`, 19
- `diss.EUCL`, 20
- `diss.FRECHET`, 21
- `diss.INT.PER`, 22
- `diss.MINDIST.SAX`, 23
- `diss.NCD`, 25
- `diss.PDC`, 27
- `diss.PER`, 28
- `diss.PRED`, 29
- `diss.SPEC.GLK`, 32
- `diss.SPEC.ISD`, 33
- `diss.SPEC.LLR`, 35
- `loo1nn.cv`, 37
- `pvalues.clust`, 40

## \*Topic `\textasciitildekw2`

- `cluster.evaluation`, 2
- `diss`, 4
- `diss.ACF`, 6
- `diss.AR.LPC.CEPS`, 8
- `diss.AR.MAH`, 9
- `diss.AR.PIC`, 11
- `diss.CDM`, 12
- `diss.CID`, 14
- `diss.COR`, 15
- `diss.CORT`, 17
- `diss.DTWARP`, 18
- `diss.DWT`, 19

- `diss.EUCL`, 20
- `diss.FRECHET`, 21
- `diss.INT.PER`, 22
- `diss.MINDIST.SAX`, 23
- `diss.NCD`, 25
- `diss.PDC`, 27
- `diss.PER`, 28
- `diss.PRED`, 29
- `diss.SPEC.GLK`, 32
- `diss.SPEC.ISD`, 33
- `diss.SPEC.LLR`, 35
- `loo1nn.cv`, 37
- `pvalues.clust`, 40

## \*Topic `datasets`

- `paired.tseries`, 38

`ar`, 12

`arma`, 8, 11, 12

`cluster.evaluation`, 2, 38

`cluster.stats`, 3

`clValid`, 3

`convert.to.SAX.symbol`, 13

`convert.to.SAX.symbol`  
(`diss.MINDIST.SAX`), 23

`diss`, 4, 9, 12, 13, 15, 16, 19, 21, 22, 25, 27,  
28, 31, 42

`diss.ACF`, 4, 6, 16

`diss.AR.LPC.CEPS`, 4, 8, 10, 12

`diss.AR.MAH`, 4, 9, 9, 12, 40

`diss.AR.PIC`, 4, 9, 10, 11

`diss.CDM`, 5, 12

`diss.CID`, 5, 14

`diss.COR`, 5, 7, 15, 18

`diss.CORT`, 5, 15, 17

`diss.DTWARP`, 5, 18, 18

`diss.DWT`, 5, 19

`diss.EUCL`, 5, 20

`diss.FRECHET`, 5, 18, 21

diss.INT.PER, [5](#), [22](#)  
diss.MINDIST.SAX, [5](#), [23](#)  
diss.NCD, [5](#), [13](#), [25](#)  
diss.PACF, [5](#), [16](#)  
diss.PACF (diss.ACF), [6](#)  
diss.PDC, [27](#)  
diss.PER, [5](#), [23](#), [28](#)  
diss.PRED, [5](#), [29](#)  
diss.SAX (diss.MINDIST.SAX), [23](#)  
diss.SPEC.GLK, [5](#), [32](#), [34](#), [36](#)  
diss.SPEC.ISD, [5](#), [33](#), [33](#), [36](#)  
diss.SPEC.LLR, [5](#), [33](#), [34](#), [35](#)  
dist, [5](#), [21](#)  
distFrechet, [18](#)  
dtw, [18](#)

electricity, [36](#)

integrate, [36](#)  
interest.rates, [37](#)

knn.cv, [38](#)

longitudinalData, [21](#)  
loo1nn, [38](#)  
loo1nn.cv, [37](#)

memCompress, [13](#), [26](#), [27](#)  
MINDIST.SAX (diss.MINDIST.SAX), [23](#)

PAA, [13](#)  
PAA (diss.MINDIST.SAX), [23](#)  
paired.tseries, [38](#)  
pdc, [5](#), [19](#), [28](#)  
pdcDist, [5](#)  
pvalues.clust, [40](#)

SAX.plot (diss.MINDIST.SAX), [23](#)  
std.ext, [3](#)  
synthetic.tseries, [41](#)

TSclust, [42](#)  
TSclust-package (TSclust), [42](#)

wavDWT, [20](#)