Package ‘dtwclust’

July 30, 2015

Type Package

Title Time series clustering with Dynamic Time Warping distance

Version 0.1.0

Date 2015-07-30

Depends flexclust, proxy, dtw

Imports methods, caTools, ggplot2, reshape2, modeltools

Suggests TSdist

Author Alexis Sarda-Espinosa

Maintainer Alexis Sarda <alexis.sarda@gmail.com>

Description Ttime series clustering using different techniques related to the Dynamic Time Warping distance and its corresponding lower bounds. Additionally, an implementation of k-Shape clustering is available.

URL https://github.com/asardaes/dtwclust

License GPL-3

LazyData TRUE

R topics documented:

`dtwclust-package` *Time series clustering under Dynamic Time Warping (DTW) distance.*

Description

Perform time series clustering using different techniques related to the DTW distance and its corresponding lower bounds (LB). Additionally, an implementation of k-Shape clustering is available.

Details

This package tries to consolidate the different procedures available to perform clustering of time series under DTW. Right now only univariate time series are supported. Similarly, time series should have equal lengths for partitional methods.

Please see the documentation for `dtwclust`, which serves as the main entry point.

Other packages that are particularly leveraged here are the `flexclust` package for partitional clustering, the `proxy` package for distance matrix calculations, and the `dtw` package for the core DTW calculations.

Four distances are registered via `pr_DB`: `"LB_Keogh", "LB_Improved", "SBD"` and `"DTW2"`. See `lb_keogh`, `lb_improved` and `SBD` for more details on the first 3. The last one is done with `dtw` using `L2` norm, but it differs from the result you would obtain if you specify `L2` as `dist.method`: with `DTW2`, pointwise distances (the local cost matrix) are calculated with `L1` norm, *each* element of the matrix is squared and the result is fed into `dtw`, which finds the optimum warping path. The square root of the resulting distance is *then* computed.

Author(s)

Alexis Sarda-Espinosa

References

Begum N, Ulanova L, Wang J and Keogh E (2015). “Accelerating Dynamic Time Warping Clustering with a Novel Admissible Pruning Strategy.” In *Conference on Knowledge Discovery and Data Mining*, series KDD ’15. ISBN 978-1-4503-3664-2/15/08, `http://doi.org/http://dx.doi.org/10.1145/2783258.2783286`.

Giorgino T (2009). “Computing and Visualizing Dynamic Time Warping Alignments in R: The ’dtw’ Package.” *Journal of Statistical Software*, **31**(7), pp. 1-24. `http://www.jstatsoft.org/v31/i07/`.

Ratanamahatana A and Keogh E (2004). “Everything you know about dynamic time warping is wrong.” In *3rd Workshop on Mining Temporal and Sequential Data, in conjunction with 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD-2004), Seattle, WA*.

Keogh E and Ratanamahatana CA (2005). “Exact indexing of dynamic time warping.” *Knowledge and information systems*, **7**(3), pp. 358-386.

Lemire D (2009). “Faster retrieval with a two-pass dynamic-time-warping lower bound .” *Pattern Recognition*, **42**(9), pp. 2169 - 2180. ISSN 0031-3203, `http://doi.org/http://dx.doi.org/10.1016/j.patcog.2008.11.030`, `http://www.sciencedirect.com/science/article/pii/S0031320308004925`.

Liao TW (2005). “Clustering of time series data - a survey.” *Pattern recognition*, **38**(11), pp. 1857-1874.

Paparrizos J and Gravano L (2015). “k-Shape: Efficient and Accurate Clustering of Time Series.” In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, series

SIGMOD '15, pp. 1855-1870. ISBN 978-1-4503-2758-9, `http://doi.org/10.1145/2723372.2737793`.

Petitjean F, Ketterlin A and Gancarski P (2011). "A global averaging method for dynamic time warping, with applications to clustering." *Pattern Recognition*, **44**(3), pp. 678 - 693. ISSN 0031-3203, `http://doi.org/http://dx.doi.org/10.1016/j.patcog.2010.09.013`, `http://www.sciencedirect.com/science/article/pii/S003132031000453X`.

Sakoe H and Chiba S (1978). "Dynamic programming algorithm optimization for spoken word recognition." *Acoustics, Speech and Signal Processing, IEEE Transactions on*, **26**(1), pp. 43-49. ISSN 0096-3518, `http://doi.org/10.1109/TASSP.1978.1163055`.

Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P and Keogh E (2013). "Experimental comparison of representation methods and distance measures for time series data." *Data Mining and Knowledge Discovery*, **26**(2), pp. 275-309. ISSN 1384-5810, `http://doi.org/10.1007/s10618-012-0250-5`, `http://dx.doi.org/10.1007/s10618-012-0250-5`.

See Also

`dtwclust`, `kcca`, `dist`, `dtw`

`DBA` *DTW Barycenter Averaging*

Description

A global averaging method for time series under DTW (Petitjean, Ketterlin and Gancarski, 2011).

Usage

```
DBA(X, center = NULL, max.iter = 25, error.check = TRUE, trace = FALSE)
```

Arguments

`X`	A data matrix where each row is a time series. Optionally, a list where each element is a time series.
`center`	Optionally, a time series to use as reference. It must be a numeric vector.Defaults to a random series of `X` if `NULL`.
`max.iter`	Maximum number of iterations allowed.
`error.check`	Should inconsistencies in the data be checked?
`trace`	If `TRUE`, the current iteration is printed to screen.

Details

This function tries to find the optimum average series between a group time series in DTW space. Refer to the cited article for specific details on the algorithm.

If a given series reference is provided in `center`, the algorithm should always converge to the same result provided the rows `X` keep the same values, although their order may change.

Value

The average time series.

References

Petitjean F, Ketterlin A and Gancarski P (2011). “A global averaging method for dynamic time warping, with applications to clustering.” *Pattern Recognition*, **44**(3), pp. 678 - 693. ISSN 0031-3203, http://doi.org/http://dx.doi.org/10.1016/j.patcog.2010.09.013, http://www.sciencedirect.com/science/article/pii/S003132031000453X.

Examples

```
# Sample data
data(uciCT)

# Obtain an average for the first 5 time series
dtw.avg <- DBA(CharTraj[1:5], CharTraj[[1]], trace = TRUE)
plot(dtw.avg, type="l")

# Change the provided order
dtw.avg2 <- DBA(CharTraj[5:1], CharTraj[[1]], trace = TRUE)
all(dtw.avg == dtw.avg2)
```

`dtwclust` *Time series clustering under DTW*

Description

This function uses the DTW distance and related lower bounds to cluster time series. For now, all series must be univariate and, in the case of partitional methods, have equal lengths.

Usage

```
dtwclust(data = NULL, type = "partitional", k = 2, method = "average",
  distance = "dtw", centroid = "pam", window.size = NULL, norm = "L1",
  dc = NULL, control = NULL, save.data = FALSE, seed = NULL,
  trace = FALSE, ...)
```

Arguments

`data`	A list where each element is a time series, or a numerical matrix where each row is a time series. Only the latter is supported in case of `type = "partitional"`.
`type`	What type of clustering method to use, `partitional`, `hierarchical` or `tadpole`.
`k`	Numer of desired clusters in partitional methods.
`method`	Which linkage method to use in hierarchical methods. See `hclust`.
`distance`	One of the supported distance definitions (see Distance section). Ignored for `type = "tadpole"`.
`centroid`	Either a supported string or an appropriate function to calculate centroids when using partitional methods (see Centroid section).
`window.size`	Window constraint for DTW and LB calculations. **See Sakoe-Chiba section**.
`norm`	Pointwise distance for DTW and LB. Either `L1` for Manhattan distance or `L2` for Euclidean. Ignored for `distance = "DTW"` (which always uses `L1`) and `distance = "DTW2"` (which always uses `L2`).

`dc`	Cutoff distance for TADPole algorithm.
`control`	Parameters for partitional clustering algorithms. See `flexclustControl`.
`save.data`	Return a copy of the data in the returned object? Ignored for hierarchical clustering.
`seed`	Random seed for reproducibility of partitional algorithms.
`trace`	Boolean flag. If true, more output regarding the progress is printed to screen.
`...`	Additional arguments to pass to `dist` or a custom distance function.

Details

Partitional algorithms are implemented via `kcca`. Hierarchical algorithms use the `hclust` function. The `tadpole` algorithm uses the `TADPole` function.

In case of partitional algorithms, `data` should be in the form of a matrix. In the other cases, it may be a matrix or a list, but the matrix will be coerced to a list. A matrix input requires that all time series have equal lengths. If the lengths vary slightly between time series, reinterpolating them to a common length is most likely an acceptable approach (Ratanamahatana and Keogh, 2004). If this is not the case, then clustering them directly is probably ill-advised. See the examples.

Value

An object with formal class `dtwclust-class` if `type = "partitional" | "tadpole"`. Otherwise an object with class `hclust` as returned by `hclust`.

Distance

If a custom distance function is provided, it will receive the data as the first argument. For partitional algorithms, the second argument will be the cluster centers (i.e. other time series) in the form of a matrix where each row is a center series. If hierarchical algorithms are used, the function will also receive the elements of `...`.

For partitional algorithms, the function *could* make use of the `window.size` and `norm` parameters, which *should* be detected thanks to `R`'s lexical scoping, however this cannot be guaranteed.

The function should return a distance matrix, ideally of class `crossdist`. In case of partitional algorithms, the time series in the data should be along the rows, and the cluster centers along the columns of the distance matrix.

The other option is to provide a string. For partitional algorithms, it can be a supported distance of `kccaFamily`. For hierarchical, it can be any distance function registered in `dist`. In the latter case, all extra parameters should be provided in `...`.

Additionally, with either type of algorithm, it can be one of the following custom implementations:

- `"dtw"`: DTW with L1 norm and optionally a Sakoe-Chiba constraint.
- `"dtw2"`: DTW with L2 norm and optionally a Sakoe-Chiba constraint.
- `"dtw_lb"`: DTW with L1 or L2 norm and optionally a Sakoe-Chiba constraint. Some computations are avoided by first estimating the distance matrix with Lemire's lower bound and then iteratively refining with DTW. See `dtw_lb`.
- `"lbk"`: Keogh's lower bound with either L1 or L2 norm for the Sakoe-Chiba constraint.
- `"lbi"`: Lemire's lower bound with either L1 or L2 norm for the Sakoe-Chiba constraint.
- `"sbd"`: Shape-based distance. Each series is z-normalized in this case. As a result, the cluster centers (for partitional methods) are also z-normalized. See `SBD` for more details.

Centroid

In the case of partitional algorithms, a suitable function should calculate the cluster centers. In this case, the centers are themselves time series.

If a custom function is provided, it will receive a matrix as only argument. Each row will be a time series that belongs to a given cluster. The function should return a numeric vector with the center time series.

The other option is to provide a character string. The following options are available:

- `"mean"`: The average along each dimension. In other words, the average of all x_i^j among the j series that belong to the same cluster for all time points t_i.
- `"median"`: The median along each dimension. Similar to mean.
- `"shape"`: Shape averaging. See `shape_extraction` for more details.
- `"dba"`: DTW Barycenter Averaging. See `DBA` for more details.
- `"pam"`: Partition around medoids. This basically means that the cluster centers are always one of the time series in the data. In this case, the distance matrix is pre-computed once using all time series in the data and then re-used at each iteration.

Sakoe-Chiba Constraint

A global constraint to speed up the DTW calculation is the Sakoe-Chiba band (Sakoe and Chiba, 1978). To use it, a window width must be defined via `window.size`.

Because of the way the different functions being used here are implemented, there is a subtle but critical mismatch in the way the window size is defined for DTW and the lower bounds (LB). The DTW calculation with `dtw` expects an *even* `window.size` that represents the distance between the diagonal and one of the edges of the window. The LB calculation expects an *odd* window.size that represents the whole window width to be used in the running max and min. The outcome of said running functions are centered with respect to the window width.

Therefore, if, for example, the DTW is calculated with a window of 10, the corresponding LB should be calculated with `2*10 + 1 = 21`.

The functions that rely on both the LB and DTW take care of this discrepancy, but the user should be careful if things are tested manually. Namely, `dtw_lb` and `TADPole` expect the DTW `window.size`, as well as `DTW` and `DTW2`. The lower bounds expect the `window.size` of the running max/min. See the examples in `lb_keogh` and `lb_improved`.

Note

Notice that the lower bounds are defined only for time series of equal lengths. `DTW` and `DTW2` don't require this, but they are much slower to compute.

The lower bounds are **not** symmetrical, and `DTW` is only symmetrical if series are of equal lengths.

Specifying `distance = "sbd"` and `centroid = "shape"` is equivalent to the k-Shape algorithm (Papparizos and Gravano, 2015). See `SBD` and `shape_extraction` for more info.

Author(s)

Alexis Sarda-Espinosa

References

Sakoe H and Chiba S (1978). "Dynamic programming algorithm optimization for spoken word recognition." *Acoustics, Speech and Signal Processing, IEEE Transactions on*, **26**(1), pp. 43-49. ISSN 0096-3518, `http://doi.org/10.1109/TASSP.1978.1163055`.

Ratanamahatana A and Keogh E (2004). "Everything you know about dynamic time warping is wrong." In *3rd Workshop on Mining Temporal and Sequential Data, in conjunction with 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD-2004), Seattle, WA*.

Paparrizos J and Gravano L (2015). "k-Shape: Efficient and Accurate Clustering of Time Series." In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, series SIGMOD '15, pp. 1855-1870. ISBN 978-1-4503-2758-9, `http://doi.org/10.1145/2723372.2737793`.

Examples

```
# Load data
data(uciCT)

# Reinterpolate to same length and coerce as matrix
data <- t(sapply(CharTraj, reinterpolate, newLength = 205))

# Simple partitional clustering with L2 distance and PAM
kc.l2 <- dtwclust(data, k = 20, distance = "kmeans", centroid = "pam", seed = 3247, trace = TRUE)
cat("Rand index for L2+PAM:", randIndex(kc.l2, CharTrajLabels), "\n\n")

# TADPole clustering (takes around 5 seconds)
kc.tadp <- dtwclust(data, type = "tadpole", k = 20, window.size = 20, dc = 1.5, save.data = TRUE)
cat("Rand index for TADPole:", randIndex(kc.tadp, CharTrajLabels), "\n\n")
plot(kc.tadp)

# Hierarchical clustering based on shabe-based distance
hc.sbd <- dtwclust(data, type = "hierarchical", distance = "sbd")
cl.sbd <- cutree(hc.sbd, 20)
cat("Rand index for HC+SBD:", randIndex(cl.sbd, CharTrajLabels), "\n\n")

## Not run:
# Use full DTW and PAM (takes around two minutes)
kc.dtw <- dtwclust(data, k = 20, seed = 3251, trace = TRUE)

# Use full DTW with DBA centroids (takes around five minutes)
kc.dba <- dtwclust(data, k = 20, centroid = "dba", seed = 3251, trace = TRUE)

## End(Not run)
```

`dtwclust-class` *Class definition for* `dtwclust`

Description

Formal S4 class to know how to handle data for plotting.

Details

It contains the following specific slots:

- `type`: A string indicating one of the supported clustering types of `dtwclust`.
- `distance`: A string indicating one of the supported distances of `dtwclust`.
- `centroid`: A string indicating one of the supported centroids of `dtwclust`.

Additionally, the class inherits from `kccasimple-class`, so all related slots and methods are also supported.

`dtw_lb` *DTW calculation guided by Lemire's lower bound (LB_Improved)*

Description

Calculation of a distance matrix with the Dynamic Time Warping (DTW) distance guided by Lemire's lower bound (LB).

Usage

```
dtw_lb(x, y = NULL, window.size = NULL, norm = "L1", error.check = TRUE)
```

Arguments

`x`	A matrix where rows are time series, or a list of time series.
`y`	An object similar to `x`.
`window.size`	The window size to use with the DTW calculation. **See details**.
`norm`	Pointwise distance. Either `L1` for Manhattan distance or `L2` for Euclidean.
`error.check`	Should inconsistencies in the data be checked?

Details

This function first calculates an initial estimate of a distance matrix between two sets of time series using Lemire's improved lower bound. Afterwards, it uses the estimate to calculate the true DTW distances between *only* the nearest neighbors of each series in `x` found in `y`. If only `x` is provided, the distance matrix is calculated between all its time series. This could be useful in case one is interested in only the nearest neighbor of one or more series among a dataset.

Because of the way the different functions being used here are implemented, there is a subtle but critical mismatch in the way the window size is defined for DTW and the LB. The DTW calculation with `dtw` expects an *even* `window.size` that represents the distance between the diagonal and one of the edges of the window. The LB calculation expects an *odd* window.size that represents the whole window width to be used in the running max and min. The outcome of said running functions are centered with respect to the window width.

Therefore, if, for example, the DTW is calculated with a window of 10, the corresponding LB should be calculated with `2*10 + 1 = 21`.

This function expects the `window.size` for `DTW` and takes care of this discrepancy automatically.

Value

The distance matrix with class `crossdist`.

Note

This function uses a lower bound that is only defined for time series of equal lengths.

References

Lemire D (2009). "Faster retrieval with a two-pass dynamic-time-warping lower bound ." *Pattern Recognition*, **42**(9), pp. 2169 - 2180. ISSN 0031-3203, http://doi.org/http://dx.doi.org/10.1016/j.patcog.2008.11.030, http://www.sciencedirect.com/science/article/pii/S0031320308004925.

See Also

`lb_improved`

Examples

```
# Load data
data(uciCT)

# Reinterpolate to same length
data <- lapply(CharTraj, reinterpolate, newLength = 205)

# Calculate the DTW distance between a certain subset aided with the lower bound
system.time(d <- dtw_lb(data[1:5], data[6:100], window.size = 20))

# Nearest neighbors
NN1 <- apply(d, 1, which.min)

# Calculate the DTW distances between all elements (about seven times slower)
system.time(d2 <- proxy::dist(data[1:5], data[6:100], method = "DTW",
                              window.type = "slantedband", window.size = 20))

# Nearest neighbors
NN2 <- apply(d2, 1, which.min)

all(NN1 == NN2)
```

`lb_improved` *Lemire's improved DTW lower bound*

Description

This function calculates a lower bound (LB) on the Dynamic Time Warp (DTW) distance between two time series. It uses a Sakoe-Chiba constraint.

Usage

```
lb_improved(x, y, window.size, norm = "L1")
```

Arguments

`x`	A time series.
`y`	A time series with the same length as `x`.
`window.size`	Window size for envelop calculation. **See details**.
`norm`	Pointwise distance. Either `L1` for Manhattan distance or `L2` for Euclidean.

Details

The lower bound is defined for time series of equal length only.

Because of the way the different functions being used here are implemented, there is a subtle but critical mismatch in the way the window size is defined for DTW and the LB. The LB calculation expects an *odd* `window.size` that represents the whole window width to be used in the running max and min. The outcome of said running functions are centered with respect to the window. The DTW calculation with `dtw` expects a window.size that represents the distance between the diagonal and one of the edges of the window.

Therefore, if, for example, the LB is calculated with a window of 21, the corresponding DTW distance should be calculated with `21 %/% 2 = 10`.

This function expects the `window.size` for the running max/min.

Value

The improved lower bound for the DTW distance.

References

Lemire D (2009). "Faster retrieval with a two-pass dynamic-time-warping lower bound ." *Pattern Recognition*, **42**(9), pp. 2169 - 2180. ISSN 0031-3203, `http://doi.org/http://dx.doi.org/10.1016/j.patcog.2008.11.030`, `http://www.sciencedirect.com/science/article/pii/S0031320308004925`.

Examples

```
# Sample data
data(uciCT)

# Lower bound distance between two series
d.lbi <- lb_improved(CharTraj[[1]], CharTraj[[2]], window.size = 11)

# Corresponding true DTW distance (accounting for window.size discrepancy)
d.dtw <- dtw(CharTraj[[1]], CharTraj[[2]], window.type = "slantedband", window.size = 5)$distance

d.lbi <= d.dtw
```

`lb_keogh` *Keogh's DTW lower bound*

Description

This function calculates a lower bound (LB) on the Dynamic Time Warp (DTW) distance between two time series. It uses a Sakoe-Chiba constraint.

Usage

```
lb_keogh(x, y, window.size, norm = "L1")
```

Arguments

`x`	A time series.
`y`	A time series with the same length as `x`.
`window.size`	Window size for envelop calculation. **See details**.
`norm`	Pointwise distance. Either `L1` for Manhattan distance or `L2` for Euclidean.

Details

The lower bound is defined for time series of equal length only.

Because of the way the different functions being used here are implemented, there is a subtle but critical mismatch in the way the window size is defined for DTW and the LB. The LB calculation expects an *odd* `window.size` that represents the whole window width to be used in the running max and min. The outcome of said running functions are centered with respect to the window. The DTW calculation with `dtw` expects a window.size that represents the distance between the diagonal and one of the edges of the window.

Therefore, if, for example, the LB is calculated with a window of 21, the corresponding DTW distance should be calculated with `21 %/% 2 = 10`.

This function expects the `window.size` for the running max/min.

Value

A list with:

- `d`: The lower bound of the DTW distance.
- `upper.env`: The time series of the upper envelope.
- `lower.env`: The time series of the lower envelope.

References

Keogh E and Ratanamahatana CA (2005). “Exact indexing of dynamic time warping.” *Knowledge and information systems*, **7**(3), pp. 358-386.

Examples

```
# Sample data
data(uciCT)

# Lower bound distance between two series
d.lbk <- lb_keogh(CharTraj[[1]], CharTraj[[2]], window.size = 11)$d

# Corresponding true DTW distance (accounting for window.size discrepancy)
d.dtw <- dtw(CharTraj[[1]], CharTraj[[2]], window.type = "slantedband", window.size = 5)$distance

d.lbk <= d.dtw
```

NCCc *Cross-correlation with coefficient normalization*

Description

This function uses FFT to compute the cross-correlation sequence between two series. They need not be of equal length.

Usage

```
NCCc(x, y)
```

Arguments

x	A time series.
y	Another time series.

Value

The cross-correlation sequence with length `length(x) + length(y) - 1`.

References

Paparrizos J and Gravano L (2015). "k-Shape: Efficient and Accurate Clustering of Time Series." In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, series SIGMOD '15, pp. 1855-1870. ISBN 978-1-4503-2758-9, http://doi.org/10.1145/2723372.2737793.

See Also

SBD

plot-dtwclust *Plot the result of* `dtwclust`

Description

Plots the time series of each cluster along with the obtained centroid. It uses `ggplot2` plotting system.

Usage

```
## S4 method for signature dtwclust,missing
plot(x, y, clus = seq_len(x@k), data = NULL,
  ...)
```

Arguments

`x`	An object of class `dtwclust-class` as returned by `dtwclust`.
`y`	Ignored.
`clus`	Which clusters to plot.
`data`	The data in the same format as it was provided to `dtwclust`.
`...`	Further arguments to pass to `geom_line` for the plotting of the *cluster centers*. Default values are: `linetype = "dashed", size = 1.5, colour = "black",alpha = 0.5`.

Details

The flag `save.data` must be set to `TRUE` when running `dtwclust` to be able to use this.

Optionally, you can manually provide the clustering result as well as the data in `data`.

See Also

`dtwclust-class`, `dtwclust`

`reinterpolate` *Wrapper for simple linear reinterpolation*

Description

This function is just a wrapper for the native function `approx` to do simple linear reinterpolation.

Usage

```
reinterpolate(ts, newLength)
```

Arguments

`ts`	A time series.
`newLength`	Desired length of the output series.

Value

Reinterpolated time series

SBD *Shape-based distance*

Description

Distance based on coefficient-normalized cross-correlation as proposed by Papparizos and Gravano, 2015, for the k-Shape clustering algorithm.

Usage

```
SBD(x, y, znorm = TRUE)
```

Arguments

x	A time series.
y	Another time series.
znorm	Should each series be z-normalized before calculating the distance?

Details

This function works best if the series are *z-normalized*. If not, at least they should have corresponding amplitudes, since the values of the signal **do** affect the outcome.

If x and y do **not** have the same length, it would be best if the longer sequence is provided in y, because it will be shifted to match x. Anything before the matching point is discarded and the series is padded with trailing zeros as needed.

The output values lie between 0 and 2, with 0 indicating perfect similarity.

Value

A list with:

- dist: The distance between x and y.
- yshift: A shifted version of y so that it optimally mathces x.

References

Paparrizos J and Gravano L (2015). “k-Shape: Efficient and Accurate Clustering of Time Series.” In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, series SIGMOD '15, pp. 1855-1870. ISBN 978-1-4503-2758-9, http://doi.org/10.1145/2723372.2737793.

See Also

NCCc, shape_extraction

`shape_extraction` *Shape average of several time series*

Description

Time-series shape extraction based on optimal alignments as proposed by Papparizos and Gravano, 2015, for the k-Shape clustering algorithm.

Usage

```
shape_extraction(X, cz = NULL, znorm = TRUE)
```

Arguments

`X`	Numeric matrix where each row is a time series.
`cz`	Center to use as basis. Should already be *normalized*. Calculation uses all `X` if `cz = NULL`.
`znorm`	Boolean flag. Should z-scores be calculated for `X` before processing?

Details

This works only if the signals are *z-normalized*, since the output will also have this normalization.

This centroid computation is casted as an optimization problem called maximization of Rayleigh Quotient. See the cited article for more details.

Value

Centroid time series.

References

Paparrizos J and Gravano L (2015). “k-Shape: Efficient and Accurate Clustering of Time Series.” In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, series SIGMOD ’15, pp. 1855-1870. ISBN 978-1-4503-2758-9, `http://doi.org/10.1145/2723372.2737793`.

See Also

`SBD`, `zscore`

Examples

```
# Sample data
data(uciCT)

# Subset of interest, normalized
X <- t(sapply(CharTraj[1:5], zscore))

# Obtain centroid series
C <- shape_extraction(X, znorm = FALSE)

# Result
```

```
matplot(t(X), type = "l", col = 1:5)
points(C)
```

TADPole *TADPole clustering*

Description

Time-series Anytime Density Peaks Clustering as proposed by Begum et al., 2015.

Usage

```
TADPole(data, window.size = NULL, k = 2, dc, error.check = TRUE)
```

Arguments

`data`	The data matrix where each row is a time series. Optionally a list with each time series.
`window.size`	Window size constraint for DTW.
`k`	The number of desired clusters.
`dc`	The cutoff distance.
`error.check`	Should the data be checked for inconsistencies?

Details

This function can be called either directly or through `dtwclust`.

TADPole clustering adopts a relatively new clustering framework and adapts it to time series clustering with DTW. See the cited article for the details of the algorithm.

Because of the way the algorithm works, it can be considered a kind of Partitioning Around Medoids (PAM). This means that the cluster centers are always elements of the data.

The algorithm first uses the DTW's upper and lower bounds to find series with many close neighbors (in DTW space). Anything below the cutoff distance (`dc`) is considered a neighbor. Aided with this information, the algorithm then tries to prune as many DTW calculations as possible in order to accelerate the clustering procedure. The series that lie in dense areas (i.e. that have lots of neighbors) are taken as cluster centers.

The algorithm relies on the DTW bounds, which are only defined for time series of equal lengths.

Because of the way the different functions being used here are implemented, there is a subtle but critical mismatch in the way the window size is defined for DTW and the lower bounds (LB). The DTW calculation with `dtw` expects an *even* `window.size` that represents the distance between the diagonal and one of the edges of the window. The LB calculation expects an *odd* window.size that represents the whole window width to be used in the running max and min. The outcome of said running functions are centered with respect to the window width.

Therefore, if, for example, the DTW is calculated with a window of 10, the corresponding LB should be calculated with `2*10 + 1 = 21`.

This function expects the `window.size` for `DTW` and takes care of this discrepancy automatically.

Value

A list with:

- `cl`: Cluster indices.
- `centers`: Indices of the centers.
- `distCalcPercentage`: Percentage of distance calculations that were actually performed.

References

Begum N, Ulanova L, Wang J and Keogh E (2015). "Accelerating Dynamic Time Warping Clustering with a Novel Admissible Pruning Strategy." In *Conference on Knowledge Discovery and Data Mining*, series KDD '15. ISBN 978-1-4503-3664-2/15/08, `http://doi.org/http://dx.doi.org/10.1145/2783258.2783286`.

`uciCT` *Subset of character trajectories data set*

Description

Subset: only 5 examples of X velocity. See details.

Format

A list with 100 elements. Each element is a time series. Labels included as factor vector.

Details

Quoting the source:

"Multiple, labelled samples of pen tip trajectories recorded whilst writing individual characters. All samples are from the same writer, for the purposes of primitive extraction. Only characters with a single pen-down segment were considered."

The subset included here (`CharTraj`) has only 5 examples of the X velocity for each character. A vector with labels is also loaded in `CharTrajLabels`.

Source

`https://archive.ics.uci.edu/ml/datasets/Character+Trajectories`

`zscore` *Wrapper for z-normalization*

Description

Wrapper for function `scale` that returns zeros instead of `NaN`.

Usage

```
zscore(x, ...)
```

Arguments

`x`	Data to normalize.
`...`	Further arguments to pass to `scale`.

Value

Normalized data.

Index