Package 'dtwclust'

July 28, 2015

Type Package

Title Time series clustering with Dynamic Time Warping distance

Version 0.1

Date 2015-07-27

Depends flexclust, proxy, dtw

Imports methods, caTools, ggplot2, reshape2, modeltools

Author Alexis Sarda Espinosa

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

Description

Perform time series clustering using different techniques related to the DTW distance and its corresponding lower bounds. Additionally, an implementation of kShape clustering is available.

License GPL-3

LazyData TRUE

R topics documented:

`dtwclust-package` *DTW*

Description

Something

Details

Something more

`DBA` *DTW Barycenter Averaging*

Description

See Petitjean 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. 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.

Value

The average time series.

dtwclust *Time series clustering with DTW*

Description

This function uses the DTW distance and related lower bounds to cluster time series. For now, all series must have equal length.

Usage

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

Arguments

data	Numeric matrix where each row is a time series.
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.
distance	One of the supported distance measurements (see details). It can also be the name of a family to use with function kcca if type == "partitional", or a supported distance of dist if type == "hierarchical".
centroid	Either a supported string (see details) or an appropriate function to calculate centroids when using partitional methods.
window.size	Window constraint for DTW and LB calculations.
norm	Pointwise distance. Either L1 for Manhattan distance or L2 for Euclidean.
dc	Cutoff distance for TADPole algorithm.
control	Parameters for partitional clustering algorithm. See flexclustControl.
save.data	Return a copy of the data in the returned object?
seed	Random seed for reproducibility
trace	Boolean flag. If true, more output regarding the progress is printed to screen.
...	Additional arguments to pass to dist.

Value

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

Author(s)

Alexis Sarda

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*

Description

Calculation of a distance matrix with the Dynamic Time Warp (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 of *only* the nearest neighbors for each series in x.

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`.

The function takes care of this discrepancy if needed, but you should be careful if you are testing things manually.

Value

The distance matrix with class `crossdist`.

`lb_improved` *Lemire's improved 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

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`.

The internal functions take care of this discrepancy if needed, but you should be careful if you are testing things manually.

Value

The improved lower bound for the DTW distance.

`lb_keogh` *Keogh's 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

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`.

The internal functions take care of this discrepancy if needed, but you should be careful if you are testing things manually.

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.

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 series.
y	Another series.

Value

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

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 provided.

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.

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

See Paparrizos 2015.

Usage

```
SBD(x, y)
```

Arguments

`x`	A time series.
`y`	Another time series.

Details

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

Value

A list with:

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

shape_extraction *Shape average of several time series*

Description

See Paparrizos 2015.

Usage

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

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 best (perhaps only) if the signals have similar amplitudes, which is why *z-normalization* is recommended.

Value

Centroid time series.

TADPole *TADPole clustering*

Description

See Begum et al. 2015

Usage

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

Arguments

data	The data matrix. 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 inconsistencies in the data be checked?

Value

A list with:

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

---

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