

# Package ‘deltaGseg’

January 17, 2025

**Type** Package

**Title** deltaGseg

**Version** 1.47.0

**Date** 2015-12-29

**Author** Diana Low, Efthymios Motakis

**Maintainer** Diana Low <lowdiana@gmail.com>

**Description** Identifying distinct subpopulations through multiscale time series analysis

**License** GPL-2

**Depends** R (>= 2.15.1), methods, ggplot2, changepoint, wavethresh, tseries, pvclust, fBasics, grid, reshape, scales

**Suggests** knitr

**VignetteBuilder** knitr

**biocViews** Proteomics, TimeCourse, Visualization, Clustering

**Collate** AllClasses.R AllGenerics.R show-methods.R plot-methods.R pvclust-mods.R helpers.R

**git\_url** <https://git.bioconductor.org/packages/deltaGseg>

**git\_branch** devel

**git\_last\_commit** 6042fb0

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2025-01-17

## Contents

chooseBreaks . . . . .	3
chooseBreaks-methods . . . . .	4
clusterPV . . . . .	4
clusterPV-methods . . . . .	5
clusterSegments . . . . .	6
clusterSegments-methods . . . . .	7

deltaGseg . . . . .	8
denoiseSegments . . . . .	9
denoiseSegments-methods . . . . .	10
diagnosticPlots . . . . .	11
diagnosticPlots-methods . . . . .	12
getAVD . . . . .	12
getAVD-methods . . . . .	13
getBreaks . . . . .	13
getBreaks-methods . . . . .	14
getIntervals . . . . .	15
getIntervals-methods . . . . .	16
getSegments . . . . .	16
getSegments-methods . . . . .	17
getTNames . . . . .	18
getTNames-methods . . . . .	19
getTraj . . . . .	19
getTraj-methods . . . . .	20
parseTraj . . . . .	21
plot-methods . . . . .	22
plotDiff . . . . .	22
plotDiff-methods . . . . .	23
pvals . . . . .	24
SegSeriesTrajectories-class . . . . .	24
SegTrajectories-class . . . . .	26
show-methods . . . . .	28
simtraj . . . . .	28
simtraj.tr . . . . .	29
simtraj.tr2 . . . . .	29
splitTraj . . . . .	30
splitTraj-methods . . . . .	31
traj1 . . . . .	31
traj1.denoise . . . . .	32
traj1.ss . . . . .	32
traj1.tr . . . . .	33
Trajectories-class . . . . .	33
transformSeries . . . . .	34
transformSeries-methods . . . . .	36
TransTrajectories-class . . . . .	36

---

chooseBreaks                      *Choose a given number of breakpoints in a set of trajectories.*

---

### Description

This function automatically chooses a subset of breakpoints from all the estimated breakpoints of function `splitTraj`

### Usage

```
chooseBreaks(breakpoints, numbreaks)
```

### Arguments

breakpoints	Numeric list, output from <code>splitTraj</code> .
numbreaks	Integer. Number of breakpoints to be returned per trajectory. Breakpoints chosen will be evenly spaced from those defined by <code>splitTraj</code> .

### Details

None.

### Value

Returns a numeric list of breakpoints, one list per trajectory.

### Author(s)

Diana H.P. Low, Efthimios Motakis

### See Also

[splitTraj](#)

### Examples

```
data(deltaGseg)
all_breakpoints<-splitTraj(traj1) #default splits=15 (i.e. 16 segments).
all_breakpoints
chooseBreaks(all_breakpoints,numbreaks=3)
```

chooseBreaks-methods *chooseBreaks*

---

**Description**

Choose evenly spaced breakpoints from a list of breakpoints.

**Methods**

signature(breakpoints="list", numbreaks="numeric") Returns a sublist of breakpoints of length numbreaks.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

---

clusterPV *clusterPV*

---

**Description**

Wrapper for modified pvclust function.

**Usage**

```
clusterPV(object, bootstrap=500)
```

**Arguments**

object            An object of class 'SegTrajectories'.  
bootstrap        Integer. Number of bootstraps to run.

**Details**

This is a wrapper to call the pvclust function that has been modified to suit our deltaGseg computation.

**Value**

Returns an object of class "pvclust". For use in [clusterSegments](#) when running the pvclust option.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

Shimodaira, H. (2004). Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling, *Annals of Statistics* 32, 2616-2641.

**See Also**

[clusterSegments](#)]

**Examples**

```
data(deltaGseg)
clusterPV(traj1.denoise)
```

---

clusterPV-methods      *clusterPV*

---

**Description**

Returns a pvclust values for object of class "SegTrajectories")

**Methods**

signature(object = "SegSeriesTrajectories") Returns an object of class "pvclust".

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**See Also**

[transformSeries](#)

**Examples**

```
#data(deltaGseg)
#clusterPV(td2)
```

---

clusterSegments	<i>clusterSegments</i>
-----------------	------------------------

---

### Description

The function does hierarchical clustering of the segmented (and joined) series by `hclust` and performs one of the "intervention" methods (see the respective parameter below) to identify subpopulations. The hierarchical clustering is performed by Euclidean distances using "average" linkage method.

### Usage

```
clusterSegments(object, intervention = "groups", pv=NULL, graphics=NULL)
```

### Arguments

<code>object</code>	An object of class 'SegTrajectories'
<code>intervention</code>	intervention: Character. One of "groups", "heights" or "pvclust". Option <code>pvclust</code> performs simple hierarchical clustering by <code>hclust</code> and then assesses the uncertainty in the clustering by the bootstrap probability values computed via multi-scale bootstrap resampling. Option "groups" asks the user to input the number of subpopulations we want to identify (interactive). Option "heights" asks the user to interactively set a threshold T (a horizontal line on the tree plot) that defines the number of subpopulations.
<code>pv</code>	Supplied p-values for <code>intervention=pvclust</code> . See <a href="#">clusterPV</a>
<code>graphics</code>	Character vector. Optional parameter defining the colors for plotting (each color indicates a different subpopulation). Must be at least the length of total number of subpopulations defined.

### Details

The algorithm offers several alternatives for subpopulation estimation that, ultimately, they lead to similar solutions. The user should first visualize the segmented data to get a rough idea of the possible subpopulations (plots generated by the algorithm). Option "pvclust" computes the hierarchical clustering tree with the p-values. The subpopulations are defined interactively by the user (point and click based on the R function `identify()`; see `help(identify)`). Alternatively, option "groups" asks the user to input the number of subpopulations or define them interactively in option "height" (point and click at the desired height in the tree). The final plot shows the number of estimated subpopulations.

### Value

An object of class `SegSeriesTrajectories`.

### Author(s)

Diana H.P. Low, Efthimios Motakis

**References**

Shimodaira, H. (2004) Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling, *Annals of Statistics*, 32, 2616-2641.

**Examples**

```
## Not run:  
## interactive!  
data(deltaGseg)  
traj1.ss<-clusterSegments(traj1.denoise, intervention = "groups") #define clusters by number of groups formed.  
  
## End(Not run)
```

---

clusterSegments-methods  
*clusterSegments*

---

**Description**

Performs the function `clusterSegments` on an object of class "SegTrajectories".

**Methods**

`signature(object = "SegTrajectories")` Returns an object of class "SegSeriesTrajectories"

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**Examples**

```
#data(deltaGseg)  
#st<-clusterSegments(dt)
```

---

deltaGseg

*deltaGseg*

---

## Description

deltaGseg

## Details

Package: deltaGseg  
Type: Package  
Version: 0.99.0  
Date: 2013-02-04  
License: GPL-2

## Author(s)

Diana H.P. Low, Efthimios Motakis

Maintainer: Diana H.P. Low <dlow@imcb.a-star.edu.sg>

## References

- Zhou W., Motakis E., Fuentes G., Verma C.S. (2012) Macrostate identification from biomolecular simulations through time series analysis. *J Chem Inf Model.* 2012 Sep 24;52(9):2319-24. Epub 2012 Sep 5.
- Dickey, D.A. and W.A. Fuller (1979), Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, 74, p. 427-431.
- Nason, G.P. (2008) *Wavelet methods in Statistics with R*. Springer, New York.
- Auger, I. E.; Lawrence, C. E. Algorithms for the optimal identification of segment neighborhoods. *Bull. Math.Biol.* 1989, 51(1), 39-54.
- Shimodaira, H. (2004) Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling, *Annals of Statistics*, 32, 2616-2641.
- D'Agostino R.B., Pearson E.S. (1973); Tests for Departure from Normality, *Biometrika* 60, 613-22.



---

denoiseSegments      *Wavelet denoising of trajectory series (segments)*

---

### Description

This function computes Augmented Dickey-Fuller test for weak-stationarity and carries out segmentation and wavelets denoising.

### Usage

```
denoiseSegments(object, seg_method="BinSeg", maxQ=15, fn=1, factor=0.8, thresh_level=TRUE, minobs=200)
```

### Arguments

object	An object of class "Trajectories" or "TransTrajectories". (the output of functions <a href="#">parseTraj</a> or <a href="#">transformSeries</a> , respectively).
seg_method	Character. One of "SegNeigh" or "BinSeg". By default it performs the Segment Neighborhood (SegNeigh) method to find multiple changes in mean for data that is assumed to be normally distributed. The value returned is the result of finding the optimal location of up to Q changepoints using the log of the likelihood ratio statistic. Once all changepoint locations have been calculated, the optimal number of changepoints is decided using $k \cdot \text{pen}$ as the penalty function where $k$ is the number of changepoints tested ( $k$ in $(1, Q)$ ). In very large series, the memory demanding "SegNeigh" can be replaced by "BinSeg" (Binary Segmentation). The segmentation is performed only if the Augmented Dickey-Fuller test P-value is significant at $\alpha=0.05$ . If not, an error message appears indicating the need of series splitting or differentiation (see <a href="#">transformSeries</a> ).
maxQ	Integer. The maximum number of Q changepoints to be estimated.
fn	Integer. It specifies the degree of smoothness of the wavelet that you want to use in the decomposition. It takes values from 1 (coarsest/hard smoothing, i.e. Haar's step function) to 10 (finest/soft smoothing). Put simply, the fitted (wavelet denoised/estimated) data of segment $q$ with $\text{fn}=1$ have lower variance than the fitted data of $q$ under $\text{fn}=10$ . The former data will resemble a step function over time while the latter will be much closer to the original data.
factor	Numeric. Between 0.6 and 1 used for re-scaling the denoising threshold.
thresh_level	Logical. If FALSE then a global threshold is computed on and applied to all scale levels. If TRUE a threshold is computed and applied separately to each scale level (for serious residuals autocorrelation).
minobs	Integer. The minimum number of observations a segment should consist of to be accepted

### Value

An object of class "SegTrajectories"

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

Dickey, D.A. and W.A. Fuller (1979), Distribution of the Estimators for Autoregressive Time Series with a Unit Root, Journal of the American Statistical Association, 74, p. 427-431.

Nason, G.P. (2008) Wavelet methods in Statistics with R. Springer, New York.

Auger, I. E.; Lawrence, C. E. Algorithms for the optimal identification of segment neighborhoods. Bull. Math.Biol. 1989, 51(1), 39-54.

**See Also**

[transformSeries](#)

**Examples**

```
data(deltaGseg)
traj1.denoise<-denoiseSegments(traj1.tr, seg_method="BinSeg",maxQ=15,fn=1,factor=0.8,thresh_level=TRUE,minobs=2
```

---

denoiseSegments-methods

*denoiseSegments*

---

**Description**

Performs the function `denoiseSegments` on an object of either class "Trajectories" or "TransTrajectories" (`classUnion="TrajORTransTraj"`).

**Methods**

`signature(object = "Trajectories")` Returns an object of class "SegTrajectories"

`signature(object = "TransTrajectories")` Returns an object of class "SegTrajectories"

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**Examples**

```
#data(deltaGseg)
#dt<-denoiseSegments(tt)
```

---

diagnosticPlots      *diagnosticPlots*

---

## Description

This function generates the diagnostic plots of the wavelet denoising model residuals. The assumptions are that the residuals autocorrelation is not significant and that the residuals distribution is approximately normal or, at least, symmetric around 0. We provide plots and test to verify these assumptions (depends on R package fBasics).

## Usage

```
diagnosticPlots(object, norm.test="KS",single.series = FALSE)
```

## Arguments

object	An object of class "SegSeriesTrajectories".
norm.test	Character. One of "KS", "Shapiro", "Agost". Test for residuals normality accepting "KS" (Kolmogorov Smirnov test with Lilliefors correction), "Shapiro" (Shapiro test for normality) and "Agost" (D'Agostino test for normality using the skewness and kurtosis of the data ; also gives the skewness and kurtosis p-values for the hypothesis that the respective estimated measures differ from the theoretical values under the normal distribution).
single.series	Logical. If FALSE (default) the residuals of each series are independently analyzed.

## Details

The function outputs the standard autocorrelation plots for viewing the residuals autocorrelation, histograms for checking the normality assumptions and the respective P-values to test the normality assumption.

## Value

A series of plots with printed P-values for the autocorrelation and normality tests.

## Author(s)

Diana H.P. Low, Efthimios Motakis

## References

D'Agostino R.B., Pearson E.S. (1973); Tests for Departure from Normality, *Biometrika* 60, 613-22.

## Examples

```
data(deltaGseg)
diagnosticPlots(traj1.ss,norm.test="KS",single.series=TRUE)
```

---

 diagnosticPlots-methods

*diagnosticPlots*


---

### Description

Performs the function diagnosticPlots on an object of class "SegSeriesTrajectories".

### Methods

signature(object = "SegSeriesTrajectories") Returns histogram and acf plots of series

### Author(s)

Diana H.P. Low, Efthimios Motakis

### References

None

### Examples

```
#data(deltaGseg)
#diagnosticPlots(st)
```

---

 getAVD

*getAVD*


---

### Description

Accessor of Trajectories or TransTrajectories object to retrieve adf p-values.

### Usage

```
getAVD(object)
```

### Arguments

object            An object of class 'Trajectories' or 'TransTrajectories'.

### Value

Returns a vector (of pvalues) for each trajectory defined in the object.

### Note

None.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**See Also**

[parseTraj](#), [transformSeries](#)

**Examples**

```
data(deltaGseg)
getAVD(traj1)
```

---

getAVD-methods	getAVD
----------------	--------

---

**Description**

Returns numeric vector from the slot @avd in an object of class "Trajectories", or @tavd in an object of class "TransTrajectories."

**Methods**

signature(object = "Trajectories") Numeric vector of length equal to series length.  
signature(object = "TransTrajectories") Numeric vector of length equal to series length.

**See Also**

[getAVD](#), [parseTraj](#), [transformSeries](#)

---

getBreaks	<i>getBreaks</i>
-----------	------------------

---

**Description**

Accessor of TransTrajectories object to retrieve computed breakpoints.

**Usage**

```
getBreaks(object)
```

**Arguments**

object            An object of class 'TransTrajectories'.

**Details**

None.

**Value**

Returns a list (of numerical breakpoint values) for each trajectory defined in an object of class 'Trajectories'.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**See Also**

[splitTraj,transformSeries](#)

**Examples**

```
data(deltaGseg)
getBreaks(traj1.tr)
```

---

*getBreaks-methods*      *getBreaks*

---

**Description**

Returns a list from the @breakpoints slot for object of class "TransTrajectories")

**Methods**

signature(object = "SegSeriesTrajectories") List of breakpoints per series after transformation of series data using transformSeries

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**See Also**

[transformSeries](#)

### Examples

```
#data(deltaGseg)
#getBreaks(traj1.tr)
```

---

`getIntervals`                      *getIntervals*

---

### Description

Helper function to retrieve subpopulations and computes the intervals for each subpopulation after segmentation and clustering.

### Usage

```
getIntervals(object)
```

### Arguments

`object`                      An object of class 'SegSeriesTrajectories'.

### Details

None.

### Value

Returns a list of subpopulations and the intervals.

### Author(s)

Diana H.P. Low, Efthimios Motakis

### References

None

### See Also

[clusterSegments](#)

### Examples

```
data(deltaGseg)
getIntervals(traj1.ss)
```

---

getIntervals-methods    *getIntervals*

---

**Description**

Returns a list of subpopulations and the intervals of the segmented series.)

**Methods**

signature(object = "SegSeriesTrajectories") Returns a list of subpopulations and the intervals of the segmented series from segmentationWithinSeries

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**See Also**

[transformSeries](#)

**Examples**

```
#data(deltaGseg)
#getIntervals(traj1.ss)
```

---

getSegments                    *getSegments*

---

**Description**

Accessor of SegTrajectories or SegSeriesTrajectories object to retrieve segment matrix.

**Usage**

```
getSegments(object)
```

**Arguments**

object                    Object of class "SegTrajectories" or "SegSeriesTrajectories"

**Value**

A matrix with segment information including quantiles.



**Note**

None.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None.

**See Also**

[denoiseSegments](#), [clusterSegments](#)

**Examples**

```
data(deltaGseg)
segments<-getSegments(traj1.denoise)
```

---

*getSegments-methods*    *getSegments*

---

**Description**

Returns a matrix from the @smatrix slot (for object of class "SegTrajectories") or @ssmatrix slot (for object of class "SegSeriesTrajectories")

**Methods**

signature(object = "SegSeriesTrajectories") Matrix is a result of the function [clusterSegments](#)  
signature(object = "SegTrajectories") Matrix is a result of the function [denoiseSegments](#)

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**Examples**

```
#data(deltaGseg)
#getSegments(traj1.denoise)
```

---

`getTNames`*getTNames*

---

**Description**

Accessor of Trajectories and TransTrajectories object to retrieve filename(s) used in trajectory computation.

**Usage**

```
getTNames(object)
```

**Arguments**

`object` An object of class 'Trajectories' or 'TransTrajectories'.

**Details**

None.

**Value**

Returns character vector of filenames.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None.

**Examples**

```
data(deltaGseg)
getTNames(traj1)
```

---

getTNames-methods	<i>Retrieve filenames of trajectories</i>
-------------------	---

---

**Description**

Returns a vector from the @filenames or @tfilenames slot.

**Details**

Returns a vector of filename(s) values.

**Methods**

signature(object = "Trajectories") Returns the original filenames of series read by parseTraj.

signature(object = "TransTrajectories") Returns filenames generated by the transformSeries function if the original series has been split into subseries. Subseries names are denoted by an underscore after the original names, eg. FILE1\_1, FILE1\_2.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**Examples**

```
#data(deltaGseg)
#getTNames(traj1)
```

---

getTraj	<i>getTraj</i>
---------	----------------

---

**Description**

Accessor of Trajectories or TransTrajectories object to retrieve trajectories.

**Usage**

```
getTraj(object)
```

**Arguments**

object            An object of class "Trajectories" or "TransTrajectories"

**Details**

None.

**Value**

List of length equal to number of trajectories, each containing matrix with 2 columns. 1: time points, 2: free energy values.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None.

**Examples**

```
data(deltaGseg)
alltrajectories<-getTraj(traj1)
```

---

getTraj-methods      *getTrajectories*

---

**Description**

Returns a list of matrices from the @trajlist slot (for object of class "Trajectories") or @ttrajlist slot (for object of class "TransTrajectories")

**Methods**

signature(object = "Trajectories") Returns the original trajectories used for computation.

signature(object = "TransTrajectories") Returns the transformed trajectories generated by the function [transformSeries](#)

**Author(s)**

Diana H.P. Low, Efthimios Motakis

---

parseTraj	<i>Reads in files containing trajectory data</i>
-----------	--

---

**Description**

Reads in files with 2-column, space-separated numerical values containing 1:time points, 2:trajectory(free binding energies).

**Usage**

```
parseTraj(path = getwd(), files = NULL, fromfile=TRUE)
```

**Arguments**

path	Directory containing trajectory files.
files	Character vector of filenames to read. If not provided, will read all files in given directory and treat them as a set. Can also be used to read in variables if given as list.
fromfile	Logical. If set to FALSE, the files parameter will be used to read in variables.

**Details**

This is an initialization function for the deltaGseg package. It reads the trajectory files (input) and reports the a short description of the file, the Augmented Dickey-Fuller test p-values for each trajectory in the set and the data plot. The input files should be in tab delimited form with 2 columns: the first column contains the time points 1, 2, ..., T and the second the free binding energies at each time point.

**Value**

A 'Trajectories' object.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None.

**See Also**

[Trajectories-class](#)

**Examples**

```

traj1<-parseTraj(path=system.file("extdata",package="deltaGseg"),files=c("D_GBTOT1","D_GBTOT2","D_GBTOT3"))
traj1 #prints summary of traj1 object

# using parseTraj for existing variables ##
subtraj<-getTraj(traj1)[[1]] #extracts first trajectory in the above series
traj2<-parseTraj(files=list(subtraj),fromfile=FALSE)
traj2

```

---

plot-methods

*Plot Trajectories-related objects*


---

**Description**

Plot "Trajectories" objects and customizes output.

**Methods**

```
signature(x = "Trajectories") plot(object,name='all',breakpoints=NULL)
```

name: Character. Name of sub-series, or if all, plots the whole series.

breakpoints: List. Supply breakpoints generated by [splitTraj](#).

```
signature(x = "TransTrajectories") plot(object,labelling=TRUE)
```

labelling: Logical. Writes labels. May be turned off to prevent overcrowding of plot.

```
signature(x = "SegTrajectories") plot(object)
```

```
signature(x = "SegSeriesTrajectories") plot(object)
```

---

plotDiff

*plotDiff*


---

**Description**

Plots (sub)series before and after transformation.

**Usage**

```
plotDiff(object,name=NULL)
```

**Arguments**

object            An object of class 'TransTrajectories'.

name             Character. Name of (sub)series.

**Details**

None.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**See Also**

[transformSeries](#)

**Examples**

```
data(deltaGseg)
plotDiff(simtraj.tr2,name="1_2")
```

---

*plotDiff-methods*      *plotDiff*

---

**Description**

Plots differentiated (sub)series in object of class "TransTrajectories")

**Methods**

signature(object = "TransTrajectories") Plots (sub)series before and after transformation.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**See Also**

[transformSeries](#)

**Examples**

```
#data(deltaGseg)
#plotDiff(traj1.tr, 'D_GBTOT3_1')
```

---

pvals	<i>pvalues data</i>
-------	---------------------

---

**Description**

pvals is the bootstrapped pvalues obtained via `pvclust` for `clusterSegments`

**Usage**

```
data(deltaGseg)
```

**Format**

```
class pvclust
```

**Source**

```
internal
```

---

```
SegSeriesTrajectories-class
  Class "SegSeriesTrajectories"
```

---

**Description**

Objects of this class is a product of the `clusterSegments` function.

**Objects from the Class**

Objects can be created by calls of the form `new("SegSeriesTrajectories", ...)`.

**Slots**

**ssmatrix:** Object of class "data.frame" The output is a data.frame with the following information (in columns): the observed data ("observed"), the estimated, wavelet denoised data ("estimated"), the residuals of the estimation ("residuals"), the estimated subpopulations IDs ("subpopulations"), the series IDs/filenames ("seriesID")

**ssparams:** Object of class "character" Parameters used to run the clustering algorithm, `clusterSegments`.

**sparams:** Object of class "character" Parameters used to run the segmentation algorithm, `denoiseSegments`.

**smatrix:** Object of class "matrix" A matrix containing the preliminary results from segmentation/denoising for each (sub)series generated by `transformSeries`. Each list element contains the following information in matrix form (in columns): the observed data (1st column), the estimated, wavelet denoised data (2nd column), the residuals of the estimation (3rd column), the starting/ending time points of each segment (4th/5th columns), the estimated segment IDs (6th column), the quantiles of the estimated data [minimum, 5%, 10%, 15%, ..., 95%, maximum] (from columns 7th to 27th) and the series IDs (28th column)



path: Object of class "character" Inherited from [Trajectories-class](#)  
 filenames: Object of class "character" Inherited from [Trajectories-class](#)  
 trajlist: Object of class "list" Trajectories. Inherited from [Trajectories-class](#)  
 avd: Object of class "numeric" adf p-values. Inherited from [Trajectories-class](#)  
 tmethod: Object of class "character" Transformation method. Inherited from [TransTrajectories-class](#)  
 breakpoints: Object of class "list" breakpoints, if used.  
 tavid: Object of class "numeric" adf p-values after transformation, if used. Inherited from [TransTrajectories-class](#)  
 ttrajlist: Object of class "list" Transformed trajectories. Inherited from [TransTrajectories-class](#)  
 tfilenames: Object of class "character" Transformed trajectories names. Inherited from [TransTrajectories-class](#)  
 ct: Object of class "numeric" Grouping information.

### Extends

Class "[SegTrajectories](#)", directly. Class "[TransTrajectories](#)", directly. Class "[Trajectories](#)", directly.

### Accessors

In the code snippets below, x is a SegSeriesTrajectories object.

[getTNames\(x\)](#): Retrieves filenames from slot filenames or tfilenames depending on whether the series has been transformed.  
[getTraj\(x\)](#): Retrieves trajectories list from slot trajlist or ttrajlist depending on whether the series has been transformed.  
[getBreaks\(x\)](#): Retrives breakpoints (if any) from slot breakpoints  
[getAVD\(x\)](#): Retrieves adf p-values from slot avd or tavid depending on whether the series has been transformed.  
[getSegments\(x\)](#): Retrieves clustered segmentation data from slot smatrix produced by [clusterSegments](#)

### Other methods

In the code snippets below, x is a SegSeriesTrajectories object.

[diagnosticPlots\(x\)](#): Generates the diagnostic plots of the wavelet denoising model residuals  
[getIntervals\(x\)](#): Helper function to retrieve subpopulations and computes the intervals for each subpopulation after segmentation and clustering.  
[plot\(x\)](#): Plots the final clustered segmentation data after [clusterSegments](#)  
[show\(x\)](#): Displays summary of object, including inherited classes. This helps in recalling the analysis path taken to produce the current results.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**See Also**

[SegTrajectories](#), [clusterSegments](#)

**Examples**

```
showClass("SegSeriesTrajectories")
```

---

SegTrajectories-class *Class* "SegTrajectories"

---

**Description**

Objects of this class is a product of the `denoiseSegments` function.

**Objects from the Class**

Objects can be created by calls of the form `new("SegTrajectories", ...)`.

**Slots**

**sparams:** Object of class "character" Parameters used to run the segmentation algorithm.

**smatrix:** Object of class "matrix" A matrix containing the preliminary results from segmentation/denoising for each (sub)series generated by `transformSeries`. Each list element contains the following information in matrix form (in columns): the observed data (1st column), the estimated, wavelet denoised data (2nd column), the residuals of the estimation (3rd column), the starting/ending time points of each segment (4th/5th columns), the estimated segment IDs (6th column), the quantiles of the estimated data [minimum, 5%, 10%, 15%, ..., 95%, maximum] (from columns 7th to 27th) and the series IDs (28th column)

**path:** Object of class "character" Inherited from [Trajectories-class](#)

**filenames:** Object of class "character" Inherited from [Trajectories-class](#)

**trajlist:** Object of class "list" Trajectories. Inherited from [Trajectories-class](#)

**avd:** Object of class "numeric" adf p-values. Inherited from [Trajectories-class](#)

**tmethod:** Object of class "character" Transformation method. Inherited from [TransTrajectories-class](#)

**breakpoints:** Object of class "list" breakpoints, if used.

**tavd:** Object of class "numeric" adf p-values after transformation, if used. Inherited from [TransTrajectories-class](#)

**tttrajlist:** Object of class "list" Transformed trajectories. Inherited from [TransTrajectories-class](#)

**tfilenames:** Object of class "character" Transformed trajectories names. Inherited from [TransTrajectories-class](#)

**Extends**

Class "[TransTrajectories](#)", directly. Class "[Trajectories](#)", directly.

**Accessors**

In the code snippets below, x is a SegTrajectories object.

`getTNames(x)`: Retrieves filenames from slot `filenames` or `tfilenames` depending on whether the series has been transformed.

`getTraj(x)`: Retrieves trajectories list from slot `trajlist` or `ttrajlist` depending on whether the series has been transformed.

`getBreaks(x)`: Retrieves breakpoints (if any) from slot `breakpoints`

`getAVD(x)`: Retrieves adf p-values from slot `avd` or `tavd` depending on whether the series has been transformed.

`getSegments(x)`: Retrieves initial segmentation data from slot `smatrix` produced by [denoiseSegments](#)

**Other methods**

In the code snippets below, x is a SegTrajectories object.

`clusterPV(x,bootstrap=500)`: Computes p-values to be used with `method="pvclust"` in `clusterSegments(x)`

`clusterSegments(x)`: clustering of segmented trajectories into similar groups.

`plot(x)`: Plots the initial segmentation data after denoising by [denoiseSegments](#)

`show(x)`: Displays summary of object, including inherited classes. This helps in recalling the analysis path taken to produce the current results.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**See Also**

[denoiseSegments](#)

**Examples**

```
showClass("SegTrajectories")
```

---

show-methods            *~~ Methods for Function show ~~*

---

**Description**

~~ Methods for function show ~~

**Methods:**

signature(object = "SegSeriesTrajectories")

signature(object = "SegTrajectories")

signature(object = "Trajectories")

signature(object = "TransTrajectories")

---

simtraj                    *Sample trajectory series*

---

**Description**

simulated trajectory for appendix example

**Usage**

data(deltaGseg)

**Format**

class Trajectories

**Source**

internal

---

simtraj.tr

*Sample trajectory series*

---

**Description**

simulated trajectory for appendix example

**Usage**

data(deltaGseg)

**Format**

class Trajectories

**Source**

internal

---

simtraj.tr2

*Sample trajectory series*

---

**Description**

simulated trajectory for appendix example

**Usage**

data(deltaGseg)

**Format**

class Trajectories

**Source**

internal

---

`splitTraj`*Identify Breakpoints in a Trajectory*

---

**Description**

`splitTraj` determines the breakpoints to split a given trajectory into a user specified number of segments. This analysis is performed for very long series (more than 20,000 time points) in order to avoid any memory allocation problems in R. Alternatively, it can be used for manual splitting of the series (see `transformSeries` with `method="override_splitting"`)

**Usage**

```
splitTraj(object, segsplits = rep(5,length(object@filenames)))
```

**Arguments**

<code>object</code>	An object of class "Trajectories".
<code>segsplits</code>	Numeric vector. The number of breakpoints. The length of <code>segsplits</code> must equal the number of trajectory series in the Trajectories object. Each value specifies the number of splits we want to impose in our long series in order to make it shorter.

**Details**

The output of the function is the estimated points that split the series into smaller sub-series. Typically, the plotted series and the estimated splits are further inspected using plots.

**Value**

A numeric list of length equal to number of trajectory series, containing the breakpoints for each series.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**Examples**

```
data(deltaGseg)
splitTraj(traj1)
```

---

splitTraj-methods	<i>splitTraj</i>
-------------------	------------------

---

**Description**

Performs the function `splitTraj`

**Methods**

`signature(object = "Trajectories")` Returns a list of breakpoints identified in the trajectory series.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**References**

None

**Examples**

```
#data(deltaGseg)
#all_breakpoints<-splitTraj(traj1)
```

---

traj1	<i>Sample trajectory series</i>
-------	---------------------------------

---

**Description**

traj1 is a sample trajectory series from the data files provided in the deltaGseg package

**Usage**

```
data(deltaGseg)
```

**Format**

class Trajectories

**Source**

internal

---

traj1.denoise	<i>Sample trajectory series</i>
---------------	---------------------------------

---

**Description**

traj1.tr is the denoised trajectory after using [denoiseSegments](#)

**Usage**

```
data(deltaGseg)
```

**Format**

```
class SegTrajectories
```

**Source**

```
internal
```

---

traj1.ss	<i>Sample trajectory series</i>
----------	---------------------------------

---

**Description**

traj1.ss is the clustered series after using [clusterSegments](#)

**Usage**

```
data(deltaGseg)
```

**Format**

```
class SegSeriesTrajectories
```

**Source**

```
internal
```



---

traj1.tr	<i>Sample trajectory series</i>
----------	---------------------------------

---

**Description**

traj1.tr is the transformed trajectory after using `transformSeries`

**Usage**

```
data(deltaGseg)
```

**Format**

```
class TransTrajectories
```

**Source**

```
internal
```

---

Trajectories-class	<i>Class "Trajectories"</i>
--------------------	-----------------------------

---

**Description**

Objects of this class is a product of the initialization function, `parseTraj`.

**Objects from the Class**

Objects can be created by calls of the form `new("Trajectories", ...)`.

**Slots**

**path:** Object of class "character" Directory where files were read from.

**filenames:** Object of class "character" Name of files read.

**trajlist:** Object of class "list" Trajectories.

**avd:** Object of class "numeric" adf p-values.

**Accessors**

In the code snippets below, `x` is a Trajectories object.

`getTNames(x)`: Retrieves filenames from slot `filenames`.

`getTraj(x)`: Retrieves trajectories list from slot `trajlist`.

`getBreaks(x)`: Retrives breakpoints (if any) from slot `breakpoints`

`getAVD(x)`: Retrieves adf p-values from slot `avd`.

**Other methods**

In the code snippets below, x is a Trajectories object.

`splitTraj(x)`: Computes likely breakpoints for the series.

`transformSeries(x)`: Apply transformation functions for the series if series is not stationary, or to split long series after determining breakpoints with `splitTraj`.

`plot(x, name='all')`: Plots the trajectory series either individually, or combined.

`show(x)`: Displays summary of object, including inherited classes. This helps in recalling the analysis path taken to produce the current results.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

**See Also**

`parseTraj`

**Examples**

```
showClass("Trajectories")
```

---

<code>transformSeries</code>	<i>Transforms a Non-Stationary Series into a Weakly-Stationary one</i>
------------------------------	--

---

**Description**

It transforms non-stationary series into weakly stationary (sub)series with three alternative methods (see parameter "methods").

**Usage**

```
transformSeries(object, method = "splitting", breakpoints = 1)
```

**Arguments**

<code>object</code>	An object of class "Trajectories".
<code>method</code>	Character. One of "differentiation", "override_splitting", "splitting". See details.
<code>breakpoints</code>	Integer (for method="splitting") or numeric vector (for method="override_splitting"). See details.

## Details

(i) `method="differentiation"`: first differences  $B[t]-B[t-1]$  are calculated and the first differentiated series is used for further analysis (segmentation and clustering). This option is needed in special cases, only when the series exhibits a trend-like behavior that cannot be removed by splitting. The first differentiations with remove the trend completely (see Appendix in the manual).

(ii) `method="splitting"`: the series are split by automatic data segmentation to subseries. This option divides a non-stationary series to a number of subseries that are weakly stationary;

(iii) `method="override_splitting"`: the series are split into subseries by user-defined cut-offs obtained from the numerical output of the `splitTraj` function. This option is for long stationary series that cannot be analyzed due to memory limitations. It can be also used for manual splitting when "splitting" option is not satisfactory. Typically, "splitting" and "override\_splitting" generate new data files of subseries.

Determining breakpoints for method (i) "splitting": an integer specifying the number of split points. This number (a single value applied to all series) denotes the number of subseries that the original series should be divided into. (ii) "override\_splitting": the parameter takes the exact values (time points coordinates) of split points (a list of length equal to the number of series; see manual). One can derive and manually insert these split points after inspecting the output of the `splitTraj` function (see manual). The user should select a few splits so that the original series is not divided into too many subseries (difficult to process because many new files are generated). Alternatively, function `chooseBreaks` automatically chooses a subset of breakpoints (not recommended to keep those without inspection)

## Value

An object of class "TransTrajectories".

## Author(s)

Diana H.P. Low, Efthimios Motakis

## References

Dickey, D.A. and W.A. Fuller (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association* 74, 427-431.

## See Also

[parseTraj](#), [splitTraj](#), [chooseBreaks](#)

## Examples

```
data(deltaGseg)
trans_series<-transformSeries(traj1,method='splitting',breakpoints=1)
```

transformSeries-methods

*transformSeries*

---

### Description

Returns a matrix from the @smatrix slot (for object of class "SegTrajectories") or @ssmatrix slot (for object of class "SegSeriesTrajectories")

### Methods

signature(object = "Trajectories") Returns an object of class "TransTrajectories"

### Author(s)

Diana H.P. Low, Efthimios Motakis

### References

None

### Examples

```
#data(deltaGseg)
#breakpoints<-chooseBreaks(all_breakpoints,3)
#tt<-transformSeries(traj1,breakpoints=breakpoints)
```

---

TransTrajectories-class

*Class "TransTrajectories"*

---

### Description

Object of this class is a product of the transformSeries function.

### Objects from the Class

Objects can be created by calls of the form new("TransTrajectories", ...).

**Slots**

**tmethod:** Object of class "character" Transformation method.  
**breakpoints:** Object of class "list" breakpoints, if used.  
**tavd:** Object of class "numeric" adf p-values after transformation.  
**ttrajlist:** Object of class "list" Transformed trajectories.  
**difftraj:** Object of class "list" Differentiated trajectories. These trajectories may be produced if `transformSeries` was used with `method="differentiation"`. The original trajectory will be kept in `ttrajlist` its differentiated version (used only for computation, not presentation) will be stored in this slot. The plotting function `plotDiff` enables the user to compare the original and differentiated versions of the subseries.  
**tfilenames:** Object of class "character" Transformed trajectories names.  
**path:** Object of class "character" Inherited from `Trajectories-class`  
**filenames:** Object of class "character" Inherited from `Trajectories-class`  
**trajlist:** Object of class "list" Trajectories. Inherited from `Trajectories-class`  
**avd:** Object of class "numeric" adf p-values. Inherited from `Trajectories-class`

**Extends**

Class "`Trajectories`", directly.

**Accessors**

In the code snippets below, x is a TransTrajectories object.

`getTNames(x)`: Retrieves filenames from slot `tfilenames`.  
`getTraj(x)`: Retrieves trajectories list from slot `ttrajlist`.  
`getBreaks(x)`: Retrives breakpoints (if any) from slot `breakpoints`  
`getAVD(x)`: Retrieves adf p-values from slot `tavd`.

**Other methods**

In the code snippets below, x is a TransTrajectories object.

`denoiseSegments(x)`: denoising and initial segmentation of trajectory series.  
`plotDiff(x, name='diff_object_name')`: Plots the original and differentiated subseries (one at a time) if `method="differentiation"` was used in `transformSeries`  
`plot(x)`: Plots the transformed series after `transformSeries`  
`show(x)`: Displays summary of object, including inherited classes. This helps in recalling the analysis path taken to produce the current results.

**Author(s)**

Diana H.P. Low, Efthimios Motakis

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

```
showClass("TransTrajectories")
```

# Index

## \* classes

SegSeriesTrajectories-class, [24](#)  
SegTrajectories-class, [26](#)  
Trajectories-class, [33](#)  
TransTrajectories-class, [36](#)

## \* datasets

pvals, [24](#)  
simtraj, [28](#)  
simtraj.tr, [29](#)  
simtraj.tr2, [29](#)  
traj1, [31](#)  
traj1.denoise, [32](#)  
traj1.ss, [32](#)  
traj1.tr, [33](#)

## \* methods

chooseBreaks-methods, [4](#)  
clusterPV-methods, [5](#)  
clusterSegments-methods, [7](#)  
denoiseSegments-methods, [10](#)  
diagnosticPlots-methods, [12](#)  
getAVD-methods, [13](#)  
getBreaks-methods, [14](#)  
getIntervals-methods, [16](#)  
getSegments-methods, [17](#)  
getTNames-methods, [19](#)  
getTraj-methods, [20](#)  
plot-methods, [22](#)  
plotDiff-methods, [23](#)  
show-methods, [28](#)  
splitTraj-methods, [31](#)  
transformSeries-methods, [36](#)

## \* package

deltaGseg, [8](#)

chooseBreaks, [3](#), [35](#)  
chooseBreaks, list, numeric-method  
(chooseBreaks-methods), [4](#)  
chooseBreaks-methods, [4](#)  
clusterPV, [4](#), [6](#), [27](#)

clusterPV, SegTrajectories-method  
(clusterPV-methods), [5](#)  
clusterPV-methods, [5](#)  
clusterSegments, [4](#), [5](#), [6](#), [15](#), [17](#), [24–27](#), [32](#)  
clusterSegments, SegTrajectories-method  
(clusterSegments-methods), [7](#)  
clusterSegments-methods, [7](#)

deltaGseg, [8](#)  
denoiseSegments, [9](#), [17](#), [24](#), [27](#), [32](#), [37](#)  
denoiseSegments, Trajectories-method  
(denoiseSegments-methods), [10](#)  
denoiseSegments, TransTrajectories-method  
(denoiseSegments-methods), [10](#)  
denoiseSegments-methods, [10](#)  
diagnosticPlots, [11](#), [25](#)  
diagnosticPlots, SegSeriesTrajectories-method  
(diagnosticPlots-methods), [12](#)  
diagnosticPlots-methods, [12](#)

getAVD, [12](#), [13](#)  
getAVD, Trajectories-method  
(getAVD-methods), [13](#)  
getAVD, TransTrajectories-method  
(getAVD-methods), [13](#)  
getAVD-methods, [13](#)  
getBreaks, [13](#)  
getBreaks, TransTrajectories-method  
(getBreaks-methods), [14](#)  
getBreaks-methods, [14](#)  
getIntervals, [15](#), [25](#)  
getIntervals, SegSeriesTrajectories-method  
(getIntervals-methods), [16](#)  
getIntervals-methods, [16](#)  
getSegments, [16](#)  
getSegments, SegSeriesTrajectories-method  
(getSegments-methods), [17](#)  
getSegments, SegTrajectories-method  
(getSegments-methods), [17](#)  
getSegments-methods, [17](#)

- getTNames, 18
- getTNames, Trajectories-method
  - (getTNames-methods), 19
- getTNames, TransTrajectories-method
  - (getTNames-methods), 19
- getTNames-methods, 19
- getTraj, 19
- getTraj, Trajectories-method
  - (getTraj-methods), 20
- getTraj, TransTrajectories-method
  - (getTraj-methods), 20
- getTraj-methods, 20
  
- parseTraj, 9, 13, 21, 34, 35
- plot (plot-methods), 22
- plot, SegSeriesTrajectories-method
  - (plot-methods), 22
- plot, SegTrajectories-method
  - (plot-methods), 22
- plot, Trajectories-method
  - (plot-methods), 22
- plot, TransTrajectories-method
  - (plot-methods), 22
- plot-methods, 22
- plotDiff, 22, 37
- plotDiff, TransTrajectories-method
  - (plotDiff-methods), 23
- plotDiff-methods, 23
- pvals, 24
- pvclust, 24
  
- SegSeriesTrajectories-class, 24
- SegTrajectories, 25, 26
- SegTrajectories-class, 26
- show, SegSeriesTrajectories-method
  - (show-methods), 28
- show, SegTrajectories-method
  - (show-methods), 28
- show, Trajectories-method
  - (show-methods), 28
- show, TransTrajectories-method
  - (show-methods), 28
- show-methods, 28
- simtraj, 28
- simtraj.tr, 29
- simtraj.tr2, 29
- splitTraj, 3, 14, 22, 30, 34, 35
- splitTraj, Trajectories-method
  - (splitTraj-methods), 31
  
- splitTraj-methods, 31
  
- traj1, 31
- traj1.denoise, 32
- traj1.ss, 32
- traj1.tr, 33
- Trajectories, 25, 27, 37
- Trajectories-class, 25, 26, 33, 37
- transformSeries, 5, 9, 10, 13, 14, 16, 20, 23,  
24, 26, 33, 34, 34, 37, 38
- transformSeries, Trajectories-method
  - (transformSeries-methods), 36
- transformSeries-methods, 36
- TransTrajectories, 25, 27
- TransTrajectories-class, 25, 26, 36