Package ‘segclust2d’

April 24, 2024

Type Package
Title Bivariate Segmentation/Clustering Methods and Tools
Version 0.3.3
URL https://github.com/rpatin/segclust2d
BugReports https://github.com/rpatin/segclust2d/issues


License GPL-3
LazyData TRUE
RoxygenNote 7.3.1
Depends R (>= 3.3.0)
Imports RColorBrewer (> 1.1-2), dplyr (> = 1.0.0), plyr (> = 1.8.4), reshape2 (> = 1.4.1), ggplot2 (> = 2.1.0), magrittr, Rcpp, zoo, grDevices, graphics, stats, utils, scales, rlang, methods, cli
Suggests knitr, rmarkdown, testthat, dygraphs (> = 1.1.1-1), xts (> = 0.9-7), leaflet (> = 1.0.1), sp (> = 1.2-3), adehabitatLT, depmixS4, moveHMM (> = 1.2), htmltools, move, devtools, spelling
LinkingTo Rcpp, RcppArmadillo
VignetteBuilder knitr
Encoding UTF-8
 Language en-US
 NeedsCompilation yes
 Author Remi Patin [aut, cre], Marie-Pierre Etienne [aut], Emilie Lebarbier [aut], Simon Benhamou [aut]
 Maintainer Remi Patin <remi.patin@normale.fr>
 Repository CRAN
 Date/Publication 2024-04-24 08:00:03 UTC

R topics documented:

add_covariates ................................................. 4
angular_speed .................................................. 5
apply_rowSums .................................................. 5
apply_subsampling .............................................. 6
argcheck_diag.var ............................................. 7
argcheck_Kmax ................................................... 7
argcheck_lmin .................................................... 8
argcheck_ncluster ............................................. 8
argcheck_order.var ............................................ 9
argcheck_ordering ............................................ 9
argcheck_scale.variable ..................................... 10
argcheck_seg.var ............................................. 10
argcheck_segclust ............................................ 11
argcheck_segmentation ....................................... 11
argcheck_type_coord ......................................... 12
arma_repmat ..................................................... 12
augment .......................................................... 13
bisig_plot ....................................................... 13
calc_BIC .......................................................... 14
calc_dist .......................................................... 14
calc_speed ........................................................ 15
calc_stat_states ............................................... 16
check_repetition .............................................. 16
chooseseg_lavielle ........................................... 17
choose_kmax ..................................................... 18
colsums_sapply ................................................. 18
cumsum_cpp ...................................................... 19
DynProg .......................................................... 19
DynProg_algo_cpp .............................................. 20
EM.algo_simultanee ........................................... 20
EM.algo_simultanee_Cpp ...................................... 21
EM.init_simultanee ........................................... 21
Estep_simultanee ............................................... 22
find_mu_sd ....................................................... 23
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gmean_simultanee</td>
<td>23</td>
</tr>
<tr>
<td>Gmixt_algo_cpp</td>
<td>24</td>
</tr>
<tr>
<td>Gmixt_simultanee</td>
<td>24</td>
</tr>
<tr>
<td>Gmixt_simultanee_fullcpp</td>
<td>25</td>
</tr>
<tr>
<td>hybrid_simultanee</td>
<td>25</td>
</tr>
<tr>
<td>initialisePhi</td>
<td>26</td>
</tr>
<tr>
<td>likelihood</td>
<td>27</td>
</tr>
<tr>
<td>logdens_simultanee_cpp</td>
<td>27</td>
</tr>
<tr>
<td>map_segm</td>
<td>28</td>
</tr>
<tr>
<td>matrixRupt</td>
<td>29</td>
</tr>
<tr>
<td>Mstep_simultanee</td>
<td>29</td>
</tr>
<tr>
<td>Mstep_simultanee_cpp</td>
<td>30</td>
</tr>
<tr>
<td>neighborsbis</td>
<td>31</td>
</tr>
<tr>
<td>plot_segm</td>
<td>32</td>
</tr>
<tr>
<td>plot_states</td>
<td>33</td>
</tr>
<tr>
<td>prepare_HMM</td>
<td>34</td>
</tr>
<tr>
<td>prepare_shiftfit</td>
<td>35</td>
</tr>
<tr>
<td>prep_segm</td>
<td>36</td>
</tr>
<tr>
<td>prep_segm_HMM</td>
<td>36</td>
</tr>
<tr>
<td>prep_segm_shiftfit</td>
<td>37</td>
</tr>
<tr>
<td>relabel_states</td>
<td>37</td>
</tr>
<tr>
<td>remat</td>
<td>38</td>
</tr>
<tr>
<td>ruptAsMat</td>
<td>38</td>
</tr>
<tr>
<td>segclust</td>
<td>39</td>
</tr>
<tr>
<td>segclust2d</td>
<td>40</td>
</tr>
<tr>
<td>segclust_internal</td>
<td>41</td>
</tr>
<tr>
<td>segmap_list</td>
<td>42</td>
</tr>
<tr>
<td>segmentation</td>
<td>43</td>
</tr>
<tr>
<td>segmentation-class</td>
<td>44</td>
</tr>
<tr>
<td>simulmode</td>
<td>48</td>
</tr>
<tr>
<td>simulshift</td>
<td>49</td>
</tr>
<tr>
<td>spatial_angle</td>
<td>49</td>
</tr>
<tr>
<td>stat_segm</td>
<td>50</td>
</tr>
<tr>
<td>stat_segm_HMM</td>
<td>51</td>
</tr>
<tr>
<td>stat_segm_shiftfit</td>
<td>51</td>
</tr>
<tr>
<td>subsample_rename</td>
<td>52</td>
</tr>
<tr>
<td>test_data</td>
<td>52</td>
</tr>
<tr>
<td>wrap_dynprog_cpp</td>
<td>52</td>
</tr>
</tbody>
</table>

**Index**  
54
Description

Add several covariates to movement observations. add_covariates adds several covariates to a data frame with movement information. It adds: distance between location, spatial angle, speed, smoothed speed, persistence and rotation velocity (calculated with spatial angle).

Usage

add_covariates(x, ...)

## S3 method for class 'Move'
add_covariates(x, coord.names = c("x", "y"), ...)

## S3 method for class 'ltraj'
add_covariates(x, coord.names = c("x", "y"), ...)

## S3 method for class 'data.frame'
add_covariates(
  x,
  coord.names = c("x", "y"),
  smoothed = FALSE,
  timecol = "dateTime",
  units = "hour",
  radius = NULL,
  ...
)

Arguments

x movement data
...
additional arguments
coord.names names of coordinates column in x
smoothed whether speed are smoothed or not
timecol names of POSIXct time column
units units for time calculation. Default "hour"
radius for spatial angle calculations

Value

data.frame with additional covariates
angular_speed

Examples

```r
## Not run: add_covariates(move_object, coord.names = c("x","y"), smoothed = T)
## Not run:
data(simulmode)
simple_data <- simulmode[,c("dateTime","x","y")]
full_data <- add_covariates(simple_data, coord.names = c("x","y"),
timecol = "dateTime",smoothed = TRUE, units ="min")
## End(Not run)
```

#### angular_speed

*Calculate angular speed along a path*

Description

angular_speed calculate turning angle between locations, taking a dataframe as input.

Usage

```r
angular_speed(x, coord.names = c("x", "y"))
```

Arguments

- **x** data.frame with locations
- **coord.names** names of coordinates column in x

Value

vector of turning angle.

Author(s)

Remi Patin, Simon Benhamou.

apply_rowSums

*apply_rowSums*

Description

Internal function for Expectation-Maximization (EM) algorithm.

Usage

```r
apply_rowSums(rupt, x)
```
apply_subsampling

Arguments
rupt current estimated breaks in signal
x bivariate signal

apply_subsampling Internal function for subsampling

Description
if subsample = FALSE do nothing.

Usage
apply_subsampling(x, is_segclust, subsample, subsample_over, subsample_by)

Arguments
x data.frame to be subsampled
is_segclust TRUE or FALSE whether the function was called from 'segclust()' or 'segmentation()'
subsample if FALSE disable subsampling
subsample_over maximum number of row accepted
subsample_by subsampling parameters

Details
else if subsample_by is missing, subsample only if nrow(x) > subsample_over, then it subsample with the minimum needed to get a data.frame smaller than subsample_over
if subsample_by is provided, use it to subsample.

Value
a data.frame
argcheck_diag.var  

Check for argument 'diag.var'

Description

Check whether argument 'diag.var' was provided. If not, propose default value.

Usage

argcheck_diag.var(diag.var, seg.var)

Arguments

diag.var  
  names of the variables on which statistics are calculated.
seg.var  
  for behavioral segmentation: names of the variables used for segmentation (either one or two names).

Value

a vector of character string

argcheck_Kmax  

Check for argument 'Kmax'

Description

Check whether argument 'Kmax' was provided and is adequate before subsampling. Propose adequate value if Kmax is not provided.

Usage

argcheck_Kmax(Kmax, lmin, datalength)

Arguments

Kmax  
  maximum number of segments.
lmin  
  minimum length of segments.
datalength  
  length of data provided

Value

an integer
argcheck_lmin  

**Description**
Check whether argument 'lmin' was provided and is adequate before subsampling

**Usage**
argcheck_lmin(lmin, is_segclust)

**Arguments**
- `lmin`: minimum length of segments.
- `is_segclust`: TRUE if function is called from `segclust`; FALSE otherwise, if function is called from `segmentation`.

**Value**
a NULL object

argcheck_ncluster  

**Description**
Check whether argument 'ncluster' was provided and is adequate

**Usage**
argcheck_ncluster(ncluster, Kmax)

**Arguments**
- `ncluster`: number of clusters into which segments should be grouped. Can be a vector if one want to test several number of clusters.
- `Kmax`: maximum number of segments.

**Value**
a NULL object
argcheck_order.var

Check for argument 'order.var'

Description
Check whether argument 'order.var' was provided. If not, propose default value.

Usage
argcheck_order.var(order.var, diag.var)

Arguments
order.var names of the variable with which states are ordered.
diag.var names of the variables on which statistics are calculated.

Value
a vector of character string

argcheck_ordering

Check for argument 'order'

Description
Check whether argument 'order' was provided for plot.segmentation and segmap. If not, propose default value.

Usage
argcheck_ordering(order, seg.type, order.var)

Arguments
order TRUE or FALSE depending on whether cluster be ordered
seg.type types of the segmentation
order.var name of the variable to order the cluster

Value
a boolean
argcheck_scale.var

Check for argument 'scale.variable'

Description
Check whether argument 'scale.variable' was provided. If not, propose default value.

Usage
argcheck_scale.var(scale.variable, is_segclust)

Arguments
scale.variable minimum length of segments.
is_segclust TRUE if function is called from segclust; FALSE otherwise, if function is called from segmentation.

Value
a boolean

argcheck_seg.var
Check for argument 'seg.var'

Description
Check whether argument 'seg.var' was adequately provided. If provided, also check for its length (1 or 2) and for the existence of corresponding column names in x If unprovided, use default value (segmentation only) and tests if column names exists.

Usage
argcheck_seg.var(x, seg.var, is_segclust)

Arguments
x data used for segmentation. Supported: data.frame, Move object, ltraj object
seg.var for behavioral segmentation: names of the variables used for segmentation (either one or two names).
is_segclust TRUE if function is called from segclust; FALSE otherwise, if function is called from segmentation.

Value
a list with a data.frame and a vector with two character strings
argcheck_segclust

Check for argument 'ncluster' and 'nseg'

Description

Check whether argument 'ncluster' and 'nseg' were provided. If not, propose default value based on BIC.

Usage

argcheck_segclust(ncluster, nseg, ncluster.BIC, Kopt.BIC)

Arguments

- ncluster: number of cluster
- nseg: number of segment
- ncluster.BIC: optimal number of cluster selected by BIC
- Kopt.BIC: optimal number of segment selected by BIC for each number of cluster

Value

a list with two integers

argcheck_segmentation

Check for argument 'nseg'

Description

Check whether argument 'nseg' was provided. If not, propose default value based on Lavielle’s criterium

Usage

argcheck_segmentation(nseg, Kopt.lavielle)

Arguments

- nseg: number of segment
- Kopt.lavielle: optimal number of segment selected with Lavielle’s criterium

Value

an integer
argcheck_type_coord \hspace{1cm} \textit{Check for deprecated 'type' and 'coord.names' argument}

\section*{Description}

Check whether argument 'type' and 'coord.names' were provided and communicate adequately if need be.

\section*{Usage}

\begin{Verbatim}
argcheck_type_coord(...)
\end{Verbatim}

\section*{Arguments}

\begin{itemize}
  \item ... additional parameters transmitted from \texttt{segmentation} or \texttt{segclust}
\end{itemize}

\section*{Value}

a NULL object

\section*{arma_repmat \hspace{1cm} arma_repmat}

\section*{Description}

C++ Armadillo version for repmat function. Repeat a matrix in bloc.

\section*{Usage}

\begin{Verbatim}
arma_repmat(A, n, m)
\end{Verbatim}

\section*{Arguments}

\begin{itemize}
  \item \texttt{A} \hspace{1cm} matrix
  \item \texttt{n} \hspace{1cm} number of repetition in line
  \item \texttt{m} \hspace{1cm} number of repetition in column
\end{itemize}
augment

Generic function for augment

Description

see broom::augment for more informations

Usage

augment(x, ...)

Arguments

x object to be augmented
...
additional arguments

bisig_plot

bisig_plot draws the plots of the bivariate signal on the same plot (scale free)

Description

bisig_plot draws the plots of the bivariate signal on the same plot (scale free)

Usage

bisig_plot(x, rupt = NULL, mu = NULL, pop = NULL, merge.seg = FALSE)

Arguments

x the signal to be plotted
rupt optional, if given add vertical lines at change points (rupt should a vector)
mu optional the mean of each class of segment,
pop optional the cluster to whom each segment belongs to,
merge.seg should segment be merged ?

Value

no value
**calc_BIC**  
*Calculate BIC*

**Description**

BIC calculates BIC given log-likelihood, number of segment and number of class.

**Usage**

```
calc_BIC(likelihood, ncluster, nseg, n)
```

**Arguments**

- `likelihood`: log-likelihood
- `ncluster`: number of cluster
- `nseg`: number of segment
- `n`: number of observations

**Value**

A data.frame with BIC, number of cluster and number of segment.

---

**calc_dist**  
*Calculate distance between locations*

**Description**

`calc_dist` calculate distance between locations, taking a dataframe as input. Distance can also be smoothed over the two steps before and after the each point.

**Usage**

```
calc_dist(x, coord.names = c("x", "y"), smoothed = FALSE)
```

**Arguments**

- `x`: data.frame with locations
- `coord.names`: names of coordinates column in `x`
- `smoothed`: whether distance are smoothed or not

**Value**

A vector of distance.
calc_speed

Author(s)
Remi Patin

Examples
## Not run: calc_dist(df,coord.names = c("x","y"), smoothed = T)

---

**calc_speed**

Calculate speed along a path

Description
calc_dist calculate speed between locations, taking a dataframe as input. Speed can also be smoothed over the two steps before and after the each point.

Usage
calc_speed(
  x,
  coord.names = c("x", "y"),
  timecol = "dateTime",
  smoothed = FALSE,
  units = "hour"
)

Arguments
- **x**: data.frame with locations
- **coord.names**: names of coordinates column in x
- **timecol**: names of POSIXct time column
- **smoothed**: whether speed are smoothed or not
- **units**: units for time calculation. Default "hour"

Value
vector of distance

Author(s)
Remi Patin

Examples
## Not run: calc_speed(df,coord.names = c("x","y"), timecol = "dateTime",
  smoothed = T)
## End(Not run)
calc_stat_states      Calculate state statistics

Description

calc_stat_states calculates statistics of a given segmentation: mean and variance of the different states.

Usage

```
calc_stat_states(data, df.segm, diag.var, order.var = NULL)
```

Arguments

- **data**: the data.frame with the different variable
- **df.segm**: output of prep_segm function
- **diag.var**: names of the variables on which statistics are calculated
- **order.var**: names of the variable with which states are ordered

Value

- a data.frame with mean and variance of the different states

Examples

```
## Not run: calc_stat_states(data, diag.var = c("dist","angle"),
order.var='dist', type='hmm',hmm.model=mod1.hmm)
## End(Not run)
```

check_repetition      Check for repetition in the series

Description

check_repetition checks whether the series have identical or near-identical repetition larger than lmin. If that is the case, throw an error, the algorithm cannot yet handle these repetition, because variance on the segment would be null.

Usage

```
check_repetition(x, lmin, rounding = FALSE, magnitude = 3)
```
**chooseseg_lavielle**

**Arguments**

- **x** the bivariate series to be tested
- **lmin** minimum length of segment
- **rounding** whether or not series are rounded
- **magnitude** number of magnitude of standard deviation below which values are rounded. i.e if magnitude = 3, difference smaller than one thousandth of the standard deviation are rounded to the same value.

**Value**

a boolean, TRUE if there is any repetition larger or equal to lmin.

**Examples**

```r
set.seed(42)
dat <- rbind(base::sample(seq(1,10), size= 100, replace = TRUE),
            base::sample(seq(1,10), size= 100, replace = TRUE))
check_repetition(dat, lmin = 3)
check_repetition(dat, lmin = 5)
```

---

**chooseseg_lavielle**  

*Internal Function for choosing optimal number of segment*

**Description**

Choosing optimal number of segment using Marc Lavielle’s method. From Emilie Lebarbier. Method based on identifying breaks in the slope of the contrast.

**Usage**

```r
chooseseg_lavielle(J, S = 0.75, ...)
```

**Arguments**

- **J** likelihood for each number of segment
- **S** threshold for choosing the number of segment. See adehabitatLT::chooseseg
- **...** additional arguments

**Value**

a list with optimal number of segment and full data.frame of the calculus
choose_kmax  

*Finding best segmentation with a different threshold S*

**Description**

Choosing optimal number of segment using Marc Lavielle’s method. From Emilie Lebarbier. Method based on identifying breaks in the slope of the contrast.

**Usage**

```r
choose_kmax(x, S = 0.75)
```

**Arguments**

- `x`: segmentation-class object
- `S`: threshold for choosing the number of segment. See adehabitatLT::chooseseg

**Value**

the optimal number of segment given threshold S.

**Examples**

```r
## Not run:
res.seg <- segmentation(df, coord.names = c("x","y"), Kmax = 30, lmin = 10)
# find the optimal number of segment according to Lavielle’s criterium with a
# different threshold.
choose_kmax(res.seg, S = 0.60)
## End(Not run)
```

---

colsums_sapply  

**Description**

Internal function for Expectation-Maximization (EM) algorithm.

**Usage**

```r
colsums_sapply(i, rupt, x, mu, tau)
```
cumsum_cpp

Arguments

- **i**: number of signal
- **rupt**: current estimated breaks in signal
- **x**: bivariate signal
- **mu**: mean parameter for each signal
- **tau**: tau

Description

C++ function for cumulative sum (replacing R cumsum)

Usage

cumsum_cpp(x)

Arguments

- **x**: Numerical Vector

DynProg

**DynProg** computes the change points given a cost matrix matD and a maximum number of segments Kmax

Description

DynProg computes the change points given a cost matrix matD and a maximum number of segments Kmax

Usage

DynProg(matD, Kmax)

Arguments

- **matD**: the cost Matrix os size n x n
- **Kmax**: the maximal number of segments

Value

a list with J.est a vector with Kmax value, the Kth is the minimum contrast for a model with K segments (-J.est is the log-likelihood) and with t.test a matrix, line K are the coordinates of the change points for a model with K segments
**DynProg_algo_cpp**

**Description**
This function finds the best segmentation given a Cost Matrix using a dynamic programming algorithm. C++ implementation of DynProg.

**Usage**
`DynProg_algo_cpp(matD, Kmax)`

**Arguments**
- `matD`: Cost Matrix
- `Kmax`: number of segments

---

**EM.algo_simultanee**

**Description**
EM.algo.simultanee calculates the MLE of phi for given change-point instants.

**Usage**
`EM.algo_simultanee(x, rupt, P, phi, eps = 1e-06, sameSigma = FALSE)`

**Arguments**
- `x`: bivariate signal
- `rupt`: the sequence of change points
- `P`: number of clusters
- `phi`: starting value for the parameter
- `eps`: eps
- `sameSigma`: TRUE if segments have the same variance

**Value**
a list with phi, the MLE, tau = (taukj) the probability for segment k to belong to class, lvinc = lvinc, empty = empty, dv = dv
EM.algo_simultanee_Cpp

EM.algo_simultanee_Cpp calculates the MLE of phi for given change-point instants and for a fixed number of clusters

Description

EM.algo_simultanee calculates the MLE of phi for given change-point instants and for a fixed number of clusters

Usage

EM.algo_simultanee_Cpp(x, rupt, P, phi, eps = 1e-06, sameSigma = FALSE)

Arguments

x: bivariate signal
rupt: the sequence of change points
P: number of clusters
phi: starting value for the parameter
eps: eps
sameSigma: TRUE if segments have the same variance

Value

a list with phi, the MLE, tau = (taukj) the probability for segment k to belong to class, lvinc = lvinc, empty = empty, dv = dv

EM.init_simultanee

EM.init_simultanee proposes an initial value for the EM algorithm based on a hierarchical clustering algorithm (ascending)

Description

EM.init_simultanee proposes an initial value for the EM algorithm based on a hierarchical clustering algorithm (ascending)

Usage

EM.init_simultanee(x, rupt, K, P)
Arguments

- \( x \) - the bivariate signal
- \( \text{rupt} \) - the change point instants, data.frame
- \( K \) - number of segments
- \( P \) - number of clusters

Value

\( \phi_0 \) - candidate for the EM algorithm

Description

Estep_simultanee computes posterior probabilities and incomplete-data log-likelihood for mixture models

Usage

\[
\text{Estep_simultanee}(\text{logdensity}, \phi, \text{eps} = 1e-09)
\]

Arguments

- \( \text{logdensity} \) - is a \( K \times P \) matrix containing the conditional log-densities for each segment
- \( \phi \) - a list containing the parameters of the mixture
- \( \text{eps} \) - eps

Value

A list with \( \tau \) a \( K \times P \) matrix, \( \tau_{kj} \) is the posterior probability for segment \( k \) to belong to class \( j \) and \( \text{lvinc} \), the incomplete log likelihood \( P(X=x) \)
**find_mu_sd**  
*Find mean and standard deviation of segments*

**Description**

`find_mu_sd` calculates statistics of a given segmentation: mean and variance of the different states.

**Usage**

`find_mu_sd(df.states, diag.var)`

**Arguments**

- `df.states`: a list of data.frame
- `diag.var`: names of the variables on which statistics are calculated

**Value**

A data.frame with mean and variance of the different states

---

**Gmean_simultanee**  
*Gmean_simultanee calculates the cost matrix for a segmentation model with changes in the mean and variance for all signals*

**Description**

`Gmean_simultanee` calculates the cost matrix for a segmentation model with changes in the mean and variance for all signals.

**Usage**

`Gmean_simultanee(Don, lmin, sameVar = FALSE)`

**Arguments**

- `Don`: the bivariate signal
- `lmin`: minimum size for a segment, default value is 2
- `sameVar`: whether variance is the same for each segment.

**Value**

The cost matrix $G(i,j)$ which contains the variance of the data between point $(i+1)$ to point $j$. 
**Gmixt_algo_cpp**

**Description**
Internal C++ algorithm for computing the cost matrix.

**Usage**
```
Gmixt_algo_cpp(zi, lgi, P, mvec, wk, svec, prop)
```

**Arguments**
- **zi**: vector of observations
- **lgi**: vector of indices
- **P**: number of class
- **mvec**: vector of means for each class
- **wk**: temporary vector for calculations
- **svec**: vector of standard deviations for each class
- **prop**: mixture vector

**Gmixt_simultanee**

**Description**
Calculates the cost matrix for a segmentation/clustering model

**Usage**
```
Gmixt_simultanee(Don, lmin, phi)
```

**Arguments**
- **Don**: the bivariate signal
- **lmin**: the minimum size for a segment
- **phi**: the parameters of the mixture

**Value**
- a matrix
Gmixt_simultanee_fullcpp

Description
C++ function replacing Gmixt_simultanee

Usage
Gmixt_simantanee_fullcpp(Don, lmin, prop, mu, s)

Arguments
Don  Bivariate Signal
lmin minimum length of segments
prop mixture parameters
mu  mean parameters
s  standard deviation parameters

hybrid_simultanee

Description
It is an algorithm which combines dynamic programming and the EM algorithm to calculate the MLE of phi and T, which are the mixture parameters and the change point instants. This algorithm is run for a given number of clusters, and estimates the parameters for a segmentation/clustering model with P clusters and 1:Kmax segments

Usage
hybrid_simultanee(
  x,
  P,
  Kmax,
  lmin = 3,
  sameSigma = TRUE,
  sameVar.init = FALSE,
  eps = 1e-06,
  lissage = TRUE,
  pureR = FALSE,
  ...
)
)
Arguments

- `x` the two-dimensional signal, one line per dimension
- `P` the number of classes
- `Kmax` the maximal number of segments
- `lmin` minimum length of segment
- `sameSigma` should segment have the same variance
- `sameVar.init` sameVar.init
- `eps` eps
- `lissage` should likelihood be smoothed
- `pureR` should algorithm run in full R or use Rcpp speed improvements
- `...` additional parameters

Value

A list with `Linc`, the incomplete loglikelihood =`Linc.param=paramtau` posterior probability

Usage

```r
initialisePhi(P, val = -Inf)
```

Arguments

- `P` number of classes
- `val` the value used for initialisation default is -Inf

Value

A set of parameter phi
**likelihood**

Generic function for likelihood

**Description**

Generic function for likelihood

**Usage**

likelihood(x, ...)

**Arguments**

- **x**: object from which likelihood can be extracted
- **...**: additional parameters

---

**logdens_simultanee_cpp**

**Description**

Calculate logdensity of a bivariate signal

**Usage**

logdens_simultanee_cpp(xk, mu, sigma, prop)

logdens_simultanee(xk, phi)

**Arguments**

- **xk**: the bivariate signal
- **mu**: mean parameter for each signal
- **sigma**: standard deviation parameter for each signal
- **prop**: mixture parameter
- **phi**: parameters of the mixture, P components

**Value**

the value of the log density
map_seg

plot_seg plot segmented movement data on a map.

Description

plot_seg plot segmented movement data on a map.

Usage

map_seg(
  data,
  output,
  interactive = FALSE,
  html = FALSE,
  scale = 1,
  UTMstring = "+proj=longlat +datum=WGS84 +no_defs",
  width = 400,
  height = 400,
  order = NULL,
  pointsize = 1,
  linesize = 0.5,
  coord.names = c("x", "y"),
  ...
)

Arguments

data the data.frame with the different variable
output outputs of the segmentation or segclust algorithm for one number of segment
interactive should graph be interactive with leaflet?
html should the graph be incorporated in a markdown file through htmltools::tagList()
scale for dividing coordinates to have compatibility with leaflet
UTMstring projection of the coordinates
width width
height height
order should cluster be ordered
pointsize size of points
linesize size of lines
coord.names names of coordinates
... additional arguments

Value

a ggplot object
## Examples

```r
## Not run:
# res.seg is a result of the segmentation-only algorithm:
res.seg = 10
outputs = res.seg$outputs[[paste(nseg, "segments")]]
map <- map_segm(data=res.seg$data, output=outputs)
# res.segclust is a result of the segmentation-clustering algorithm:
nseg = 10; ncluster = 3
outputs = res.segclust$outputs[[paste(ncluster,"class -",nseg, "segments")]]
map <- map_segm(data=res.seg$data, output=outputs)

## End(Not run)
```

---

### matrixRupt

`matrixRupt` transforms a vector of change point into a data frame with start and end of every segment.

#### Description

`matrixRupt` transforms a vector of change point into a data frame with start and end of every segment.

#### Usage

```r
matrixRupt(x, vectorRupt)
```

#### Arguments

- `x`: the vectorRupt
- `vectorRupt`: the vector containing the change point

#### Value

- the matrix of change point

---

### Mstep_simultanee

`Mstep_simultanee` computes the MLE within the EM framework.

#### Description

`Mstep_simultanee` computes the MLE within the EM framework.

#### Usage

```r
Mstep_simultanee(x, rupt, tau, phi, sameSigma = TRUE)
```
**Mstep_simultanee_cpp**

**Arguments**

- **x**: the bivariate signal
- **rupt**: the rupture dataframe
- **tau**: the K*P matrix containing posterior probabilities of membership to clusters
- **phi**: the parameters of the mixture
- **sameSigma**: TRUE if all segments have the same variance

**Value**

- **phi**: the updated value of the parameters

---

**Description**

*Mstep_simultanee computes the MLE within the EM framework*

**Usage**

```r
Mstep_simultanee_cpp(x, rupt, tau, phi, sameSigma = TRUE)
```

**Arguments**

- **x**: the bivariate signal
- **rupt**: the rupture dataframe
- **tau**: the K*P matrix containing posterior probabilities of membership to clusters
- **phi**: the parameters of the mixture
- **sameSigma**: whether segments have the same variance

**Value**

- **phi**: the updated value of the parameters
neighbors tests whether neighbors of point $k,P$ can be used to re-initilize the EM algorithm and to improve the log-likelihood.

**Usage**

```r
neighborsbis(
  kv.hull,
  x,
  L,
  k,
  param,
  P,
  lmin,
  eps,
  sameSigma = TRUE,
  pureR = FALSE
)
```

**Arguments**

- `kv.hull`: convex hull of likelihood
- `x`: the initial dataset
- `L`: the likelihood
- `k`: the points of interest
- `param`: param outputs of segmentation
- `P`: the number of class
- `lmin`: minimal size of the segment to be implemented
- `eps`: eps
- `sameSigma`: should segments have same variance?
- `pureR`: should algorithm use only R functions or benefit from Rcpp faster algorithm

**Value**

smoothing likelihood
Description

`plot_segm` plot segmented time serie.

Usage

```r
plot_segm(
  data,
  output,
  interactive = FALSE,
  diag.var,
  x_col = "expectTime",
  html = FALSE,
  order = FALSE,
  stationarity = FALSE
)
```

Arguments

- **data**: the data.frame with the different variable
- **output**: outputs of the segmentation or segclust algorithm for one number of segment
- **interactive**: should graph be interactive through leaflet ?
- **diag.var**: names of the variables on which statistics are calculated
- **x_col**: column name for time
- **html**: should the graph be incorporated in a markdown file through htmltools::tagList()
- **order**: should cluster be ordered
- **stationarity**: if TRUE, cut each segment in three and plot each part with its own mean to assess stationarity of each segment

Value

a graph

Examples

```r
## Not run:
#res.segclust is the results of the segmentation-clustering algorithm
ncluster = 3
nseg = 10

g <- plot_segm(data = res.segclust$data, output =
  res.segclust$outputs[[paste(ncluster,"class -",nseg, "segments")]],
  diag.var = x$Diagnostic variables',x_col = 'dateTime)
#res.seg is the results of the segmentation-only algorithm
```
plot_states

nseg = 10

g <- plot_states(data = res.segclust$data,
output = res.segclust$outputs[[paste(nseg, "segments")]],
diag.var = x$"Diagnostic variables", x_col = "dateTime")

### End(Not run)

---

**plot_states**

*Plot states statistics*

**Description**

plot_states plot states statistics

**Usage**

plot_states(outputs, diag.var, position_width = 0.3, order = FALSE)

**Arguments**

- **outputs**: outputs of the segmentation or segclust algorithm for one number of segment
- **diag.var**: names of the variables on which statistics are calculated
- **position_width**: width between different model if several models are compared
- **order**: should cluster be ordered

**Value**

a graph

**Examples**

```r
## Not run:
res.segclust is the results of the segmentation-clustering algorithm
ncluster = 3
nseg = 10
g <- plot_states(output = res.segclust$outputs[[
    paste(ncluster,"class -",nseg, "segments")
]],
diag.var = c("dist","angle2")
res.seg is the results of the segmentation-only algorithm
nseg = 10
g <- plot_states(output = res.segclust$outputs[[paste(nseg, "segments")]],
diag.var = c("dist","angle2"))

## End(Not run)
```

prepare_HMM

Prepare HMM output for proper comparison plots

Description

prepare_HMM

Usage

prepare_HMM(data, hmm.model = NULL, diag.var, order.var = diag.var[1])

Arguments

data
data
hmm.model  hmm.model
diag.var  diag.var
order.var  order.var

Examples

## Not run:
# Example taken from moveHMM package.
# 1. simulate data
# define all the arguments of simData
nbAnimals <- 1
nbStates <- 2
nbCovs <- 2
mu <- c(15, 50)
sigma <- c(10, 20)
angleMean <- c(pi, 0)
kappa <- c(0.7, 1.5)
stepPar <- c(mu, sigma)
anglePar <- c(angleMean, kappa)
stepDist <- "gamma"
angleDist <- "vm"
zeroInflation <- FALSE
obsPerAnimal <- c(50, 100)

data <- moveHMM::simData(nbAnimals = nbAnimals, nbStates = nbStates,
                         stepDist = stepDist, angleDist = angleDist,
                         stepPar = stepPar, anglePar = anglePar, nbCovs = nbCovs,
                         zeroInflation = zeroInflation,
                         obsPerAnimal = obsPerAnimal)

### 2. fit the model to the simulated data
# define initial values for the parameters
mu0 <- c(20, 70)
sigma0 <- c(10, 30)
kappa0 <- c(1, 1)
stepPar0 <- c(mu0, sigma0) # no zero-inflation, so no zero-mass included
anglePar0 <- kappa0 # the angle mean is not estimated,
# so only the concentration parameter is needed
formula <- ~ cov1 + cos(cov2)
m <- moveHMM::fitHMM(data = data, nbStates = nbStates, stepPar0 = stepPar0,
anglePar0 = anglePar0, formula = formula,
stepDist = stepDist, angleDist = angleDist, angleMean = angleMean)

### 3. Transform into a segmentation-class object
res.hmm <- prepare_HMM(data = data,
hmm.model = m, diag.var = c("step","angle"))
### 4. you can now apply the same function than for segclust2d outputs
plot(res.hmm)
segmap(res.hmm)

## End(Not run)

---

**prepare_shiftfit**  
Prepare shiftfit output for proper comparison plots

### Description
prepare_shiftfit

### Usage
```
prepare_shiftfit(
  data,  # data
  shiftfit.model = NULL,  # shiftfit.model
  diag.var,  # diag.var
  order.var = diag.var[1]  # order.var
)
```

### Arguments
- **data**  
- **shiftfit.model**  
- **diag.var**  
- **order.var**

### Examples
```
## Not run:
data(simulshift)
# 1. subsample to a reasonable size
subdata <- simulshift[seq(1, 30000, by = 100),]
# 2. use algorithm from marcher package
MWN.fit <- with(subdata,
marcher::estimate_shift(T=indice, X=x, Y=y, n.clust = 3))
# 3. convert output
MWN.segm <- prepare_shiftfit(subdata,MWN.fit,diag.var = c("x","y"))
# 4. use segclust2d functions
plot(MWN.segm)
plot(MWN.segm,stationarity = TRUE)
segmap(MWN.segm)
## End(Not run)

prep_segm

Find segment and states for a Picard model

Description

prep_segm find the different segment and states of a given HMM model

Usage

prep_segm(data, param, seg.type = NULL, nseg = NULL)

Arguments

data the data.frame with the different variable
param the param output of the segmentation
seg.type either 'hybrid' or 'dynprog'
nseg number of segment chosen

Value

a data.frame with states of the different segments

prep_segm_HMM

Internal function for HMM

Description

prep_segm_HMM

Usage

prep_segm_HMM(data, hmm.model)

Arguments

data data
hmm.model hmm.model
prep_segm_shiftfit

Description

prep_segm_shiftfit

Usage

prep_segm_shiftfit(data, shiftfit.model)

Arguments

data
data
shiftfit.model shiftfit.model

relabel_states

Description

relabel_states relabel the states of a segmentation/clustering output. This allows merging different states into the same if for instance several of the model states represent the same behavioural states.

Usage

relabel_states(mode.segclust, newlabel, ncluster, nseg, order = TRUE)

Arguments

mode.segclust segclust output
newlabel a vector with the new names ordered, corresponding to state_ordered
ncluster the number of cluster for which you want relabeling
nseg the number of segment for which you want relabeling
order boolean, whether this changes the ordered states or not. FALSE value obsolete for now

Value

a segmentation object with state names changed for the segmentation specified by ncluster and nseg
**repmat**  
repmat repeats a matrix

**Usage**  
repmat(a, n, m)

**Arguments**  
- **a**: the base matrix  
- **n**: number of repetition in lines  
- **m**: number of repetition in columns

**Value**  
a matrix with n repeats of a in lines et m in columns

---

**ruptAsMat**  
ruptAsMat is an internal function to transform a vector giving the change point to matrix 2 columns matrix in which each line gives the beginning and the end of a segment

**Description**  
ruptAsMat is an internal function to transform a vector giving the change point to matrix 2 columns matrix in which each line gives the beginning and the end of a segment

**Usage**  
ruptAsMat(vectRupt)

**Arguments**  
- **vectRupt**: the vector of change point

**Value**  
the matrix containing the segments
Description

Joint Segmentation/Clustering of movement data. Method available for data.frame, move and ltraj objects. The algorithm finds the optimal segmentation for a given number of clusters and segments using an iterated alternation of a Dynamic Programming algorithm and an Expectation-Maximization algorithm. Among the different segmentation found, the best one can be chosen using the maximum of a BIC penalized likelihood.

Usage

segclust(x, ...)

## S3 method for class 'data.frame'
segclust(x, ...)

## S3 method for class 'Move'
segclust(x, ...)

## S3 method for class 'ltraj'
segclust(x, ...)

Arguments

x       data.frame with observations
...
additional parameters given to segclust_internal.

Value

a segmentation-class object

Examples

#' @examples
df <- test_data$data
#' # data is a data.frame with column 'x' and 'y'
#' Simple segmentation with automatic subsampling
#' if data has more than 1000 rows:
res <- segclust(df,
   Kmax = 15, lmin = 10, ncluster = 2:4,
   seg.var = c("x","y"))
#' # Plot results
plot(res)
segmap(res, coord.names = c("x","y"))
#' # check penalized likelihood of
#' alternative number of segment possible.
There should be a clear break if the segmentation is good
plot_BIC(res)
## Not run:
# Advanced options:
# Run with automatic subsampling if df has more than 500 rows:
res <- segclust(df, Kmax = 30, lmin = 10, ncluster = 2:4,
seg.var = c("x","y"), subsample_over = 500)
# Run with subsampling by 2:
res <- segclust(df, Kmax = 30, lmin = 10, ncluster = 2:4,
seg.var = c("x","y"), subsample_by = 2)
# Disable subsampling:
res <- segclust(df, Kmax = 30, lmin = 10,
ncluster = 2:4, seg.var = c("x","y"), subsample = FALSE)
# Disabling automatic scaling of variables for segmentation (standardazing
# the variables):
res <- segclust(df, Kmax = 30, lmin = 10,
seg.var = c("dist","angle"), scale.variable = FALSE)
## End(Not run)

segclust2d

segclust2d: tools for segmentation of animal GPS movement data

Description

Provides two methods for segmentation and joint segmentation/clustering of bivariate time-series. Originally intended for ecological segmentation (home-range and behavioural modes) but easily applied on other series, the package also provides tools for analysing outputs from R packages moveHMM and marcher.

Details

The segmentation method is a bivariate extension of Lavielle’s method available in adehabitatLT (Lavielle 1999; and 2005). This method rely on dynamic programming for efficient segmentation.

The segmentation/clustering method alternates steps of dynamic programming with an Expectation-Maximization algorithm. This is an extension of Picard et al (2007) method (formerly available in cghseg package) to the bivariate case.

The full description of the method is published in Patin et al. (2020).

References:


segclust_internal

Author(s)

Maintainer: Remi Patin <remi.patin@normale.fr>
Authors:
  • Marie-Pierre Etienne
  • Emilie Lebarbier
  • Simon Benhamou

See Also

Useful links:
  • https://github.com/rpatin/segclust2d
  • Report bugs at https://github.com/rpatin/segclust2d/issues

segclust_internal  Internal segmentation/clustering function

Description

Internal segmentation/clustering function

Usage

segclust_internal(
  x,
  seg.var,
  diag.var,
  order.var,
  Kmax,
  ncluster,
  lmin,
  scale.variable,
  sameSigma = FALSE,
  ...
)

Arguments

  x  data.frame with observations
  seg.var  names of the variables used for segmentation (either one or two names).
  diag.var  names of the variables on which statistics are calculated.
  order.var  names of the variable with which states are ordered.
  Kmax  maximum number of segments.
Description

segmap_list create maps with a list of object of segmentation class

Usage

segmap_list(
  x_list,
  ncluster_list = NULL,
  nseg_list = NULL,
  pointsize = 1,
  linesize = 0.5,
  coord.names = c("x", "y")
)

Arguments

x_list            list of segmentation objects for different individuals or path
ncluster_list     list of number of cluster to be selected for each individual. If empty, the function takes the default one
nseg_list         list of number of segment to be selected for each individual. If empty, the function takes the default one
pointsize         size of points
linesize          size of lines
coord.names       names of coordinates

Value

a ggplot2 graph
Segmentation of movement data - Generic function

Description

Segmentation of movement data. No clustering. Method available for data.frame, move and ltraj object. The algorithm finds for each number of segment the optimal segmentation using a Dynamic Programming approach. The number of segment is then chosen using Lavielle’s (2005) procedure based on locating rupture in the penalized likelihood.

Usage

segmentation(x, ...)

## S3 method for class 'data.frame'
segmentation(x, ...)

## S3 method for class 'Move'
segmentation(x, ...)

## S3 method for class 'ltraj'
segmentation(x, ...)

segmentation_internal(
  x,
  seg.var,
  diag.var,
  order.var,
  lmin,
  Kmax,
  scale.variable,
  sameSigma = FALSE,
  ...
)

Arguments

x       data.frame with observations
...      additional parameters given to chooseseg_lavielle
seg.var names of the variables used for segmentation (either one or two names).
diag.var names of the variables on which statistics are calculated.
order.var names of the variable with which states are ordered.
lmin     minimum length of segments.
Kmax     maximum number of segments.
scale.variable TRUE or FALSE for automatic scaling of variables (reduction and centering)
sameSigma does segments have same variance?
Value

a **segmentation-class** object

Examples

df <- test_data()$data
#' # data is a data frame with column 'x' and 'y'
# Simple segmentation with automatic subsampling
# if data has more than 1000 rows:
res <- segmentation(df, Kmax = 30, lmin = 10, seg.var = c("x","y"))
#' # Plot results
plot(res)
segmap(res)
#' # check likelihood of alternative number of segment possible. There should
#' # be a clear break if the segmentation is good
plot_likelihood(res)

## Not run:
#' # Advanced options:
#' # Run with automatic subsampling if df has more than 500 rows:
res <- segmentation(df, Kmax = 30, lmin = 10, 
seg.var = c("x","y"), subsample_over = 500)

#' # Run with subsampling by 2:
res <- segmentation(df, Kmax = 30, lmin = 10, 
seg.var = c("x","y"), subsample_by = 2)

#' # Disable subsampling:
res <- segmentation(df, Kmax = 30, lmin = 10, 
seg.var = c("x","y"), subsample = FALSE)

#' # Run on other kind of variables :
res <- segmentation(df, Kmax = 30, lmin = 10, seg.var = c("dist","angle"))

#' # Automatic scaling of variables for segmentation
#' (set a mean of 0 and a standard deviation of 1 for both variables)
res <- segmentation(df, Kmax = 30, lmin = 10, seg.var = c("dist","angle"), scale.variable = TRUE)

## End(Not run)

---

**segmentation-class**  **segmentation class description**

Description

segmentation class description

print.segmentation prints object of segmentation class
plot.segmentation plot object of segmentation class - wrapper for plot_seg
likelihood.segmentation deprecated function for plotting likelihood estimates of segmentation object. Now use plot_likelihood.
plot_likelihood plot likelihood estimates of a segmentation object - works only for picard segmentation.
get_likelihood returns likelihood estimates of a segmentation object. Deprecated, now use logLik.segmentation.
logLik.segmentation returns log-likelihood estimates of a segmentation object
plot_BIC plot BIC estimates of a segmentation object - works only for segclust algorithm.
BIC returns BIC-based penalized log-likelihood estimates of a segmentation object when segmentation/clustering has been run.
stateplot plot state distribution of a segmentation object
states return data.frame with states statistics a segmentation object
segment return data.frame with segment information of a segmentation object
augment.segmentation return data.frame with original data and state information of a segmentation object
segmap create maps with object of segmentation class (interpreting latitude/longitude)

Usage

## S3 method for class 'segmentation'
print(x, max.level = 1, ...)

## S3 method for class 'segmentation'
plot(x, nseg, ncluster, interactive = FALSE, xcol = "indice", order, ...)

## S3 method for class 'segmentation'
likelihood(x, ...)

plot_likelihood(x)

get_likelihood(x)

## S3 method for class 'segmentation'
logLik(object, ...)

plot_BIC(x)

## S3 method for class 'segmentation'
BIC(object, ...)

stateplot(x, nseg, ncluster, order)

states(x, nseg, ncluster)
segment(x, nseg, ncluster)

## S3 method for class 'segmentation'
augment(x, nseg, ncluster, colname_state = "state", ...)

segmap(
  x,
  interactive = FALSE,
  nseg,
  ncluster,
  html = FALSE,
  scale = 1,
  width = 400,
  height = 400,
  order,
  pointsize = 1,
  linesize = 0.5,
  ...
)

Arguments

- **x**: a segmentation object generated by `segmentation`
- **max.level**: argument to be passed to `utils::str()`
- **...**: additional arguments
- **nseg**: number of segment chosen
- **ncluster**: number of classes chosen
- **interactive**: whether plot are interactive (dygraphs/leaflet) or not (ggplot2)
- **xcol**: column for x axis. can be POSIXct
- **order**: should cluster be ordered
- **object**: a segmentation-class object, created by segclust.
- **colname_state**: column name for the added state column
- **html**: whether htmltools::tagList should be applied on the returned object object for integrating in html pages
- **scale**: for dividing coordinates to have compatibility with leaflet
- **width**: width
- **height**: height
- **pointsize**: size of points
- **linesize**: size of lines

Examples

```r
## Not run:
plot(res.segclust)
```
plot(res.segclust, nseg = 10, ncluster = 3)
## End(Not run)
## Not run:
plot_likelihood(res.seg)
## End(Not run)
## Not run:
logLik(res.seg)
## End(Not run)
## Not run:
plot_BIC(res.segclust)
## End(Not run)
## Not run:
plot_BIC(res.segclust)
## End(Not run)
## Not run:
stateplot(res.segclust)
stateplot(res.seg)
## End(Not run)
## Not run:
states(res.segclust)
states(res.seg)
## End(Not run)
## Not run:
segment(res.segclust)
segment(res.segclust, ncluster = 3, nseg = 30)
segment(res.seg)
segment(res.seg, nseg = 4)
## End(Not run)
## Not run:
augment(res.segclust)
augment(res.segclust, ncluster = 3, nseg = 30)
augment(res.seg)
augment(res.seg, nseg = 4)
## End(Not run)
## Not run:
segmap(res.segclust, coord.names = c("x", "y"))
segmap(res.segclust, ncluster = 3, nseg = 30)
segmap(res.seg)
```
segmap(res.seg, nseg = 4)
## End(Not run)
```

### simulmode

Simulations of behavioural mode

**Description**

A dataset containing a simulation of 3 different behavioural mode

**Usage**

```
simulmode
```

**Format**

A data frame with 302 rows and 10 variables:

- **indice**: index of position
- **x**: x coordinates
- **y**: y coordinates
- **speed**: smoothed speed
- **spatial_angle**: angle at constant step length
- **dist**: raw speed
- **angle**: angular speed
- **vit_p**: persistence speed
- **vit_r**: rotation speed
- **vit_p_spa**: persistence speed calculated with spatial angles
- **vit_r_spa**: rotation speed calculated with spatial angles
- **dateTime**: arbitrary date in POSIXct format
simulshift  Simulations of home-range shift

Description
A dataset containing a simulation of home-range shift

Usage
simulshift

Format
A data frame with 53940 rows and 10 variables:

indice  index of position
x  x coordinates
y  y coordinates
dateTime  arbitrary date in POSIXct format

spatial_angle  Calculate spatial angle along a path

Description
spatial_angle calculate spatial angle between locations, taking a dataframe as input. Spatial angle is considered as the angle between the focus point, the first location entering a given circle and the last location inside.

Usage
spatial_angle(x, coord.names = c("x", "y"), radius = NULL)

Arguments
x  data.frame with locations
coord.names  names of coordinates column in x
radius  for angle calculation. Default is median of step length.

Value
vector of spatial angle.
Author(s)

Remi Patin, Simon Benhamou.

Examples

## Not run:
data(simulmode)
spatial_angle(simulmode)

## End(Not run)

stat_segm

Calculate statistics on a given segmentation

Description

stat_segm calculates statistics of a given segmentation: mean and variance of the different states.

it also creates standard objects for plot.

Usage

stat_segm(
data, diag.var, order.var = NULL, param = NULL, seg.type = NULL, nseg
)

Arguments

data the data.frame with the different variable
diag.var names of the variables on which statistics are calculated
order.var names of the variable with which states are ordered
param parameters of output segmentation
seg.type either 'hybrid' or 'dynprog'
nseg number of segment chosen

Value

a list which first element is a data.frame with states of the different segments and which second element is a data.frame with mean and variance of the different states
Examples

```r
## Not run:
# res.segclust is a result of a segmentation-clustering algorithm
param <- res.segclust$param[["3 class"]]
nseg = 10
out <- stat_segm(data, diag.var = c("dist", "angle"),
  order.var = "dist", param = param, nseg = nseg, seg.type = "segclust")

## End(Not run)
```

---

**stat_segm_HMM**  
*Get segment statistic for HMM model*

### Description

**stat_segm_HMM**

### Usage

```r
stat_segm_HMM(data, hmm.model = NULL, diag.var, order.var = NULL)
```

### Arguments

- **data**: data
- **hmm.model**: hmm.model
- **diag.var**: diag.var
- **order.var**: order.var

---

**stat_segm_shiftfit**  
*Get segment statistic for shiftfit model*

### Description

**stat_segm_shiftfit**

### Usage

```r
stat_segm_shiftfit(data, shiftfit.model = NULL, diag.var, order.var = NULL)
```

### Arguments

- **data**: data
- **shiftfit.model**: shiftfit.model
- **diag.var**: diag.var
- **order.var**: order.var
subsample_rename

**Description**
merge subsampled data.frame df with fulldata to add segmentation information on the full data.frame

**Usage**
subsample_rename(df, fulldata, colname)

**Arguments**
df subsampled data.frame with additional information on segmentation
fulldata full data.frame
colname column name

test_data

**Description**
Test function generating fake data

**Usage**
test_data()

wrap_dynprog_cpp

**Description**
DynProg Rcpp DynProg computes the change points given a cost matrix matD and a maximum number of segments Kmax

**Usage**
wrap_dynprog_cpp(G, K)
Arguments

G the cost Matrix of size n x n
K the number of segments considered

Value

a list with J.est a vector with Kmax value, the Kth is the minimum contrast for a model with K segments (-J.est is the log-likelihood) and with t.test a matrix, line K are the coordinates of the change points for a model with K segments
Index

* datasets
  - simulmode, 48
  - simulshift, 49

add_covariates, 4
angular_speed, 5
apply_rowSums, 5
apply_subsampling, 6
argcheck_diag.var, 7
argcheck_Kmax, 7
argcheck_lmin, 8
argcheck_nclust, 8
argcheck_order.var, 9
argcheck_ordering, 9
argcheck_scale.variable, 10
argcheck_seg.var, 10
argcheck_segclust, 11
argcheck_segmentation, 11
argcheck_type_coord, 12
arma_repmat, 12
augment, 13
augment.segmentation
  (segmentation-class), 44
BIC.segmentation (segmentation-class), 44
bisig_plot, 13

calc_BIC, 14
calc_dist, 14
calc_speed, 15
calc_stat_states, 16
check_repetition, 16
choose_kmax, 18
chooseseg_lavielle, 17, 42, 43
colsums_sapply, 18
cumsum_cpp, 19

DynProg, 19, 20
DynProg_algo_cpp, 20

EM.algo_simultanee, 20
EM.algo_simultanee_Cpp, 21
EM.init_simultanee, 21
Estep_simultanee, 22

find_mu_sd, 23

get_likelihood (segmentation-class), 44
Gmean_simultanee, 23
Gmixt_algo_cpp, 24
Gmixt_simultanee, 24, 25
Gmixt_simultanee_fullcpp, 25

hybrid_simultanee, 25
initialisePhi, 26

likelihood, 27
likelihood.segmentation
  (segmentation-class), 44
logdens_simultanee
  (logdens_simultanee_cpp), 27
logdens_simultanee_cpp, 27
logLik.segmentation, 45
logLik.segmentation
  (segmentation-class), 44

map_segm, 28
matrixRupt, 29
Mstep_simultanee, 29
Mstep_simultanee_cpp, 30

neighborsbis, 31

plot.segmentation (segmentation-class), 44
plot_BIC (segmentation-class), 44
plot_likelihood, 45
plot_likelihood (segmentation-class), 44
plot_segm, 32, 45
plot_states, 33
INDEX

prep_segm, 36
prep_segm_HMM, 36
prep_segm_shiftfit, 37
prepare_HMM, 34
prepare_shiftfit, 35
print.segmentation
  (segmentation-class), 44
relabel_states, 37
repmat, 38
ruptAsMat, 38
segclust, 8, 10, 12, 39
segclust2d, 40
segclust2d-package (segclust2d), 40
segclust_internal, 39, 41
segmap (segmentation-class), 44
segmap_list, 42
segment (segmentation-class), 44
segmentation, 8, 10, 12, 43, 46
segmentation-class, 44
segmentation_internal (segmentation), 43
simulmode, 48
simulshift, 49
spatial_angle, 49
stat_segm, 50
stat_segm_HMM, 51
stat_segm_shiftfit, 51
stateplot (segmentation-class), 44
states (segmentation-class), 44
subsample_rename, 52

test_data, 52

wrap_dynprog_cpp, 52