Package ‘segclust2d’

February 27, 2019

**Type** Package

**Title** Bivariate Segmentation/Clustering Methods and Tools

**Version** 0.2.0


**License** GPL-3

**LazyData** TRUE

**RoxygenNote** 6.1.1

**Depends** R (>= 3.3.0)

**Imports** RColorBrewer (>= 1.1-2), dplyr (>= 0.5.0), plyr (>= 1.8.4), reshape2 (>= 1.4.1), ggplot2(>= 2.1.0), magrittr, Rcpp, zoo, grDevices, graphics, stats, utils, scales

**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

**LinkingTo** Rcpp, RcppArmadillo

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Remi Patin [aut, cre], Marie-Pierre Etienne [aut], Emilie Lebarbier [aut], Simon Benhamou [aut]
R topics documented:

add_covariates .......................................................... 3
angular_speed .......................................................... 4
apply_rowSums .......................................................... 5
arma_repmat .............................................................. 5
augment ................................................................. 6
bisig_plot ............................................................... 6
calc_BIC ................................................................. 7
calc_dist ................................................................. 7
calc_speed ............................................................... 8
calc_stat_states ......................................................... 9
check_repetition ........................................................ 9
chooseseg_lavielle ..................................................... 10
choose_kmax ............................................................ 11
colsums_sapply ........................................................ 11
cumsum_cpp ............................................................. 12
DynProg ................................................................. 12
DynProg_algo_cpp ...................................................... 13
EM.algo_simultanee .................................................... 13
EM.algo_simultanee_Cpp ............................................. 14
EM.init_simultanee .................................................... 14
Estep_simultanee ....................................................... 15
find_mu_sd .............................................................. 16
Gmean_simultanee ...................................................... 16
Gmixt_algo_cpp ......................................................... 17
Gmixt_simultanee ...................................................... 17
Gmixt_simultanee_fullcpp ............................................ 18
hybrid_simultanee ...................................................... 18
initialisePhi ............................................................ 19
likelihood ............................................................... 19
logdens_simultanee_cpp ............................................... 20
map_segm .............................................................. 20
matrixRupt ............................................................ 21
Mstep_simultanee ....................................................... 22
Mstep_simultanee_cpp ................................................ 22
neighborsbis .......................................................... 23
plot_segm .............................................................. 24
plot_states ............................................................ 24
prepare_HMM ........................................................... 25
prepare_shiffit ......................................................... 27
prep_segm .............................................................. 28
prep_segm_HMM ......................................................... 28
### Description

Add several covariates to movement observations. `add_covariates` add 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

```r
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 = F, timecol = "dateTime", units = "hour",
               radius = NULL, ...)
```
angular_speed

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

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

```
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**

**Description**

Internal function for Expectation-Maximization (EM) algorithm.

**Usage**

apply_rowSums(rupt, x)

**Arguments**

- **rupt**: current estimated breaks in signal
- **x**: bivariate signal

**arma_repmat**

**Description**

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

**Usage**

arma_repmat(A, n, m)

**Arguments**

- **A**: matrix
- **n**: number of repetition in line
- **m**: number of repetition in column
**augment**

*Generic function for augment*

**Description**

see broom::augment for more informations

**Usage**

```r
augment(x, ...)
```

**Arguments**

- `x`: object to be augmented
- `...`: additional arguments

---

**bisig_plot**

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

**Description**

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

**Usage**

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

**Arguments**

- `x`: the signal to be plotted
- `rupt`: optionnal, if given add vertical lines at change points (rupt should a vector)
- `mu`: optionnal the mean of each class of segment,
- `pop`: optionnal 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**

```r
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**

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

**Arguments**

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

**Value**

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 = F, units = "hour")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>data.frame with locations</td>
</tr>
<tr>
<td>coord.names</td>
<td>names of coordinates column in x</td>
</tr>
<tr>
<td>timecol</td>
<td>names of POSIXct time column</td>
</tr>
<tr>
<td>smoothed</td>
<td>whether speed are smoothed or not</td>
</tr>
<tr>
<td>units</td>
<td>units for time calculation. Default &quot;hour&quot;</td>
</tr>
</tbody>
</table>

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  

**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**

```r
## 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  

**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)
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

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

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

## Not run:
resseg <- 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(resseg, S = 0.60)

## End(Not run)

colsums_sapply  colsums_sapply

Description

Internal function for Expectation-Maximization (EM) algorithm.

Usage

colsums_sapply(i, rupt, x, mu, tau)
**Arguments**

- \( i \) : number of signal
- \( \text{rupt} \) : current estimated breaks in signal
- \( x \) : bivariate signal
- \( \mu \) : mean parameter for each signal
- \( \tau \) : tau

---

**cumsum_cpp**

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

**Usage**

\[ \text{cumsum\_cpp}(x) \]

**Arguments**

- \( x \) : Numerical Vector

---

**DynProg**

*DynProg computes the change points given a cost matrix \( \text{matD} \) and a maximum number of segments \( \text{Kmax} \)*

**Description**

DynProg computes the change points given a cost matrix \( \text{matD} \) and a maximum number of segments \( \text{Kmax} \)

**Usage**

\[ \text{DynProg}(\text{matD}, \text{Kmax}) \]

**Arguments**

- \( \text{matD} \) : the cost Matrix of size \( n \times n \)
- \( \text{Kmax} \) : the maximal number of segments

**Value**

A list with \( \text{J.est} \) a vector with \( \text{Kmax} \) value, the \( Kth \) is the minimum contrast for a model with \( K \) segments (\( -\text{J.est} \) is the log-likelihood) and with \( \text{t.test} \) a matrix, line \( K \) are the coordinates of the change points for a model with \( K \) segments
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  EM.algo_simultanee caculates the MLE of phi for given change-point instants

Description

EM.algo_simultanee caculates 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  ahe 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, tauj =(tau_kj) the probability for segment k to belong to classe,lvinc = lvinc,empty = empty,dv = dv
EM.algo_simultaneeCpp

EM.algo_simultanee 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_simultaneeCpp(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 = (tau_kj) 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)
Estep_simultanee

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

\begin{verbatim}
Estep_simultanee  Estep_simultanee computes posterior probabilities and incomplete-
data log-likelihood for mixture models
\end{verbatim}

Description

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

Usage

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

Arguments

\begin{itemize}
  \item \texttt{logdensity} \texttt{is a K*P matrix containing the conditional log-densities for each segment}  
  \item \texttt{phi} \texttt{a list containing the parameters of the mixture}  
  \item \texttt{eps} \texttt{eps}
\end{itemize}

Value

\begin{itemize}
  \item a list with \( \tau \) a \( K*P \) matrix, \( \tau_{kj} \) is the posterior probability for segment \( k \) to belong to classe \( j \)  
  \item \texttt{lvinc} the incomplete log vrais \( P(X=x) \)
\end{itemize}
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**

```r
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**

```r
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**

```cpp
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**

**Gmixt_simultanee calculates the cost matrix for a segmentation/clustering model**

**Description**

Gmixt_simultanee calculates the cost matrix for a segmentation/clustering model

**Usage**

```cpp
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 $G(i,j)$, the mixture density for segment between points $(i+1)$ to $j = \sum_{p=1}^{P} \log(\pi_p f(y_{ij}; \theta_p))$

**Rq:** this density if factorized in order to avoid numerical zeros in the log
**Gmixt_simultanee_fullcpp**

**Description**

C++ function replacing Gmixt_simultanee

**Usage**

```
Gmixt_simultanee_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**

performs a simultaneous seg - clustering for bivariate signals.

**Usage**

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

**Arguments**

- **x**: the two-dimensionnal 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
**initialisePhi**

```r
initialisePhi = function(p, val = -Inf) {
  phi = list()
  for (i in 1:p) {
    phi[[i]] = list()
    phi[[i]]$sameVar.init = sameVar.init
    phi[[i]]$eps = eps
    phi[[i]]$lissage = if (lissage) TRUE else FALSE
    phi[[i]]$pureR = if (pureR) TRUE else FALSE
  }
  list(Linc = Linc, param = paramtau, posterior.prob = posterior.prob)
}
```

**Value**

a list with \( L_{inc} \), the incomplete loglikelihood = \( L_{inc} \cdot \text{param} = \text{paramtau} \) posterior probability

**Description**

`initialisePhi` is the constructor for a set of parameters for a segclust model

**Usage**

`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_simultanea_cpp

Description
Calculate logdensity of a bivariate signal

Usage
logdens_simultanea_cpp(xk, mu, sigma, prop)

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_segm

Description
plot_segm plot segmented traject on a map.

Usage
map_segm(data, output, interactive = F, html = F, 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"))
matrixRupt transforms a vector of change point into a data.frame with start and end of every segment

Usage

matrixRupt(x, vectorRupt)
**Arguments**

- `x` - the vector containing the change point
- `rupt` - the vector containing the change point

**Value**

- the matrix of change point

---

**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` - TRUE if all segment have the same variance

**Value**

- `phi` - the updated value of the parameters

---

**Usage**

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

**Description**

Mstep_simultanee computes the MLE within the EM framework
neighborsbis

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

Description

neighborsbis tests whether neighbors of point k,P can be used to re-initialize the EM algorithm and to improve the log-likelihood.

Usage

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

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

smoothin likelihood
plot_segm

Plot segmentation on time-serie

Description

plot_segm plot segmented time serie.

Usage

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

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

## 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$'Diagnostic variables',x_col = 'dateTime')
#res.seg is the results of the segmentation-only algorithm
nseg = 10
g <- plot_segm(data = res.segclust$data, 
  output = res.segclust$outputs[[paste(nseg, "segments")]],
  diag.var = x$'Diagnostic variables',x_col = 'dateTime')

## End(Not run)
plot_states

Description

plot_states plot states statistics

Usage

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

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

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

## End(Not run)

prepare_HMM

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"))
prepare_shiftfit

### 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**

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

**Arguments**

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

**Examples**

```r
## 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
Description

prep_segm_shiftfit

Usage

prep_segm_shiftfit(data, shiftfit.model)

Arguments

data data
shiftfit.model shiftfit.model

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
**Description**
repmap 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

---

**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
Segmentation/Clustering of movement data - Generic function

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 cluster 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, Kmax, lmin, ncluster, type = "behavior",
  seg.var = NULL, diag.var = seg.var, order.var = seg.var[1],
  coord.names = c("x", "y"), ...)

## S3 method for class 'Move'
segclust(x, Kmax, lmin, ncluster, type = "behavior",
  seg.var = NULL, diag.var = seg.var, order.var = seg.var[1],
  coord.names = c("x", "y"), ...)

## S3 method for class 'ltraj'
segclust(x, Kmax, lmin, ncluster, type = "behavior",
  seg.var = NULL, diag.var = seg.var, order.var = seg.var[1],
  coord.names = c("x", "y"), ...)

Arguments

x            data used for segmentation. Supported: data.frame, Move object, ltraj object

...          additional parameters given to segclust_internal.

Kmax         maximum number of segments.

lmin         minimum length of segments.

ncluster     number of cluster into which segments should be grouped. Can be a vector if one want to test several number of clusters.

type         type of segmentation. Either "home-range" or "behavior". Changes default values of arguments order, scale.variable in the different functions used on the output. Default for segmentation: "home-range"; default for segmentation/clustering: "behavior".

seg.var      for behavioral segmentation: names of the variables used for segmentation (either one or two names).
diag.var

for behavioral segmentation: names of the variables on which statistics are calculated.

order.var

for behavioral segmentation: names of the variable with which states are ordered.

coord.names

for home-range segmentation and data.frame, names of coordinates. Default x and y. Not needed for move and ltraj objects

Value

a segmentation-class object

Examples

```r
# @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)
```

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 relies 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 not published yet.

References:


**segclust_internal**

**Internal segmentation/clustering function**

**Description**

Internal segmentation/clustering function

**Usage**

```r
segclust_internal(x, seg.var = NULL, diag.var = NULL, 
order.var = NULL, scale.variable = NULL, Kmax, ncluster = NULL, 
lmin = NULL, dat = NULL, type = NULL, sameSigma = FALSE, 
subsample_over = 1000, subsample_by = NA, subsample = TRUE, ...)
```

**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).
- `diag.var`: for behavioral segmentation: names of the variables on which statistics are calculated.
- `order.var`: for behavioral segmentation: names of the variable with which states are ordered.
- `scale.variable`: should variables be standardized? (reduced and centered)
- `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

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, Kmax, lmin, type = "home-range",
    seg.var, diag.var = seg.var, order.var = seg.var[1],
    coord.names = c("x", "y"), ...)

## S3 method for class 'Move'
segmentation(x, Kmax, lmin, type = "home-range",
    seg.var = NULL, diag.var = seg.var, order.var = seg.var[1],
    coord.names = c("coords.x1", "coords.x2"), ...)

## S3 method for class 'ltraj'
segmentation(x, Kmax, lmin, type = "home-range",
    seg.var = NULL, diag.var = seg.var, order.var = seg.var[1],
    coord.names = c("x", "y"), ...)

Arguments

x          data used for segmentation. Supported: data.frame, Move object, ltraj object
...        additional parameters given to segmentation_internal
Kmax       maximum number of segments.
lmin       minimum length of segments.
type       type of segmentation. Either "home-range" or "behavior". Changes default values of arguments order, scale.variable in the different functions used on the output. Default for segmentation: "home-range"; default for segmentation/clustering : "behavior".
seg.var    for behavioral segmentation: names of the variables used for segmentation (either one or two names).
diag.var   for behavioral segmentation: names of the variables on which statistics are calculated.
order.var  for behavioral segmentation: names of the variable with which states are ordered.
coord.names  for home-range segmentation and data.frame, names of coordinates. Default x and y. Not needed for move and ltraj objects

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, coord.names = c("x", "y"),
type = 'home-range')
# 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, coord.names = c("x", "y"),
type = 'home-range', subsample_over = 500)

## Run with subsampling by 2:
res <- segmentation(df, Kmax = 30, lmin = 10, coord.names = c("x", "y"),
type = 'home-range', subsample_by = 2)

## Disable subsampling:
res <- segmentation(df, Kmax = 30, lmin = 10, coord.names = c("x", "y"),
type = 'home-range', subsample = FALSE)

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

## 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"),
type = 'behavior', scale.variable = TRUE)

## End(Not run)
Description

segmentation class description
print.segmentation prints object of segmentation class
plot.segmentation plot object of segmentation class - wrapper for plot_segm
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

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

## S3 method for class 'segmentation'
plot(x, nseg = NULL, ncluster = NULL,
     interactive = F, 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 = NULL, ncluster = NULL, order = NULL)

states(x, nseg = NULL, ncluster = NULL)

segment(x, nseg = NULL, ncluster = NULL)

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

segmap(x, interactive = F, nseg = NULL, ncluster = NULL, html = F,
    scale = 1, width = 400, height = 400, order = NULL,
    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

## Not run:
plot(res.segclust)
plot(res.segclust, nseg = 10, ncluster = 3)

## End(Not run)
## Not run:
plot_likelihood(res.seg)
## segmentation_internal

### Internal segmentation function

```r
## 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:
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)
```
Description

Internal segmentation function

Usage

 segmentation_internal(x, seg.var = NULL, diag.var = NULL, 
 order.var = NULL, scale.variable = NULL, Kmax, lmin = NULL, 
 dat = NULL, type = NULL, sameSigma = F, subsample_over = 10000, 
 subsample = TRUE, subsample_by = NA, ...) 

Arguments

 x data.frame with observations
 seg.var for behavioral segmentation: names of the variables used for segmentation (either one or two names).
 diag.var for behavioral segmentation: names of the variables on which statistics are calculated.
 order.var for behavioral segmentation: names of the variable with which states are ordered.
 scale.variable should variables be standardized? (reduced and centered)
 Kmax maximum number of segments.
 lmin minimum length of segments.
 dat bivariate data (one signal per row)
 type type of segmentation. Either "home-range" or "behavior". Changes default values of arguments order, scale.variable in the different functions used on the output. Default for segmentation: "home-range"; default for segmentation/clustering: "behavior".
 sameSigma does segments have same variance?
 subsample_over over which size should subsampling begin (depending on speed and memory limitations)
 subsample if FALSE disable subsample
 subsample_by override subsample_over to adjust manually subsampling
 ...

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** smooth 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

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**

```r
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**

```r
## 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**

```r
stat_segm(data, diag.var, order.var = NULL, param = NULL, 
seg.type = NULL, nseg)
```
**stat_segm_HMM**

**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)
```

---

**Get segment statistic for HMM model**

**Description**

The **stat_segm_HMM** function

**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`  
- `shiftfit.model`  
- `diag.var`  
- `order.var`  

**subsample**  
*Internal function for subsampling*

**Description**

If `nrow(x) > subsample_over`, subsample with the minimum needed to get a data.frame smaller than `subsample_over`.

**Usage**

```r
subsample(x, subsample_over, subsample_by = NA)
```

**Arguments**

- `x`  
- `subsample_over`  
- `subsample_by`
### subsample_rename

**Internal function for subsampling**

**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

**Test function generating fake data**

**Description**
Test function generating fake data

**Usage**
```
test_data()
```

### wrap_dynprog_cpp

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

**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 os size \( n \times n \)
- \( K \) the number of segments considered

Value

a list with \( \text{J.est} \) a vector with \( K_{\text{max}} \) value, the \( K_{\text{th}} \) is the minimum contrast for a model with \( K \) segments (\( \text{J.est} \) is the log-likelihood) and with \( \text{t.test} \) a matrix, line \( K \) are the coordinates of the change points for a model with \( K \) segments
Index

*Topic datasets
   simulmode, 40
   simulshift, 41
add_covariates, 3
angular_speed, 4
apply_rowSums, 5
arma_repmat, 5
augment, 6
augment.segmentation
   (segmentation-class), 36
BIC.segmentation (segmentation-class), 36
bisig_plot, 6
calc_BIC, 7
calc_dist, 7
calc_speed, 8
calc_stat_states, 9
check_repetition, 9
choose_kmax, 11
chooseseg_lavielle, 10, 34, 40
colsums_sapply, 11
cumsum_cpp, 12
DynProg, 12, 13
DynProg.algo_cpp, 13
EM.algo_simultanee, 13
EM.algo_simultanee_Cpp, 14
EM.init_simultanee, 14
Estep_simultanee, 15
find_mu_sd, 16
get_likelihood (segmentation-class), 36
Gmean_simultanee, 16
Gmixt.algo_cpp, 17
Gmixt_simultanee, 17, 18
Gmixt_simultanee_fullcpp, 18
hybrid_simultanee, 18
initialisePhi, 19
likelihood, 19
likelihood.segmentation
   (segmentation-class), 36
logdens_simultanee
   (logdens_simultanee_cpp), 20
logdens_simultanee_cpp, 20
logLik.segmentation, 37
logLik.segmentation
   (segmentation-class), 36
map.segm, 20
matrixRupt, 21
Mstep_simultanee, 22
Mstep_simultanee_cpp, 22
neighborsbis, 23
plot.segmentation (segmentation-class), 36
plot_BIC (segmentation-class), 36
plot_likelihood, 37
plot_likelihood (segmentation-class), 36
plot.segm, 24, 37
plot_states, 25
prep.segm, 28
prep.segm_HMM, 28
prep.segm_shiftfit, 29
prepare_HMM, 25
prepare_shiftfit, 27
print.segmentation
   (segmentation-class), 36
relabel_states, 29
repmat, 30
ruptAsMat, 30
segclust, 31
segclust2d, 32
segclust2d-package (segclust2d), 32
segclust_internal, 31, 33
segmap (segmentation-class), 36
segmap_list, 34
segment (segmentation-class), 36
segmentation, 35, 38
segmentation-class, 36
segmentation_internal, 35, 39
simulmode, 40
simulshift, 41
spatial_angle, 42
stat_segm, 42
stat_segm_HMM, 43
stat_segm_shiftfit, 44
stateplot (segmentation-class), 36
states (segmentation-class), 36
subsample, 44
subsample_rename, 45
test_data, 45
wrap_dynprog_cpp, 45