Package ‘moveHMM’

June 23, 2025

Type Package

Title Animal Movement Modelling using Hidden Markov Models

Version 1.11

Date 2025-06-17

Maintainer Theo Michelot <`theo.michelot@dal.ca`>

Description Provides tools for animal movement modelling using hidden Markov models. These include processing of tracking data, fitting hidden Markov models to movement data, visualization of data and fitted model, decoding of the state process, etc. <doi:10.1111/2041-210X.12578>.

URL `https://github.com/TheoMichelot/moveHMM`,
`https://cran.r-project.org/package=moveHMM`

License GPL-3

LazyData TRUE

Depends R (>= 3.5.0)

Imports Rcpp, boot, MASS, sp, geosphere, ggplot2, ggmap, numDeriv

LinkingTo Rcpp, RcppArmadillo

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Theo Michelot [aut, cre],
Roland Langrock [aut, ctb],
Toby Patterson [aut, ctb],
Brett McClintock [ctb],
Eric Rexstad [ctb]

Repository CRAN

Date/Publication 2025-06-23 15:40:02 UTC

Contents

`AIC.moveHMM` *AIC*

Description

Akaike information criterion of a moveHMM model.

Usage

```
## S3 method for class 'moveHMM'
AIC(object, ..., k = 2)
```

Arguments

`object`	A `moveHMM` object.
`...`	Optional additional `moveHMM` objects, to compare AICs of the different models.
`k`	Penalty per parameter. Default: 2; for classical AIC.

Value

The AIC of the model(s) provided. If several models are provided, the AICs are output in ascending order.

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m
AIC(m)
```

`angleCI` *Confidence intervals for angle parameters*

Description

Simulation-based computation of confidence intervals for the parameters of the angle distribution. Used in `CI`.

Usage

```
angleCI(m, alpha, nbSims = 10^6)
```

Arguments

`m`	A `moveHMM` object
`alpha`	Range of the confidence intervals. Default: 0.95 (i.e. 95% CIs).
`nbSims`	Number of simulations. Default: 10^6.

Value

A list of the following objects:

`lower`	Lower bound of the confidence interval for the parameters of the angle distribution
`upper`	Upper bound of the confidence interval for the parameters of the angle distribution

---

`CI` *Confidence intervals*

Description

Computes the confidence intervals of the step length and turning angle parameters, as well as for the transition probabilities regression parameters.

Usage

```
CI(m, alpha = 0.95, nbSims = 10^6)
```

Arguments

`m`	A `moveHMM` object
`alpha`	Range of the confidence intervals. Default: 0.95 (i.e. 95% CIs).
`nbSims`	Number of simulations in the computation of the CIs for the angle parameters. Default: 10^6.

Value

A list of the following objects:

`stepPar`	Confidence intervals for the parameters of the step lengths distribution
`anglePar`	Confidence intervals for the parameters of the turning angles distribution
`beta`	Confidence intervals for the regression coefficients of the transition probabilities.

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

CI(m)
```

`dexp_rcpp` *Exponential density function*

Description

Probability density function of the exponential distribution (written in C++)

Usage

```
dexp_rcpp(x, rate, foo = 0)
```

Arguments

`x`	Vector of quantiles
`rate`	Rate
`foo`	Unused (for compatibility with template)

Value

Vector of densities

`dgamma_rcpp` *Gamma density function*

Description

Probability density function of the gamma distribution (written in C++)

Usage

```
dgamma_rcpp(x, mu, sigma)
```

Arguments

`x`	Vector of quantiles
`mu`	Mean
`sigma`	Standard deviation

Value

Vector of densities

`dlnorm_rcpp` *Log-normal density function*

Description

Probability density function of the log-normal distribution (written in C++)

Usage

```
dlnorm_rcpp(x, meanlog, sdlog)
```

Arguments

`x`	Vector of quantiles
`meanlog`	Mean of the distribution on the log-scale
`sdlog`	Standard deviation of the distribution on the log-scale

Value

Vector of densities

`dvm` *Density function of von Mises distribution*

Description

Density function of von Mises distribution

Usage

```
dvm(x, mu, kappa, log = FALSE)
```

Arguments

`x`	Angle
`mu`	Mean parameter
`kappa`	Concentration parameter
`log`	Should log-density be returned?

Value

Von Mises density

`dvm_rcpp` *Von Mises density function*

Description

Probability density function of the Von Mises distribution, defined as a function of the modified Bessel function of order 0 (written in C++)

Usage

```
dvm_rcpp(x, mu, kappa)
```

Arguments

`x`	Vector of quantiles
`mu`	Mean
`kappa`	Concentration

Value

Vector of densities

`dweibull_rcpp` *Weibull density function*

Description

Probability density function of the Weibull distribution (written in C++)

Usage

```
dweibull_rcpp(x, shape, scale)
```

Arguments

`x`	Vector of quantiles
`shape`	Shape
`scale`	Scale

Value

Vector of densities

`dwrpcauchy` *Density function of wrapped Cauchy distribution*

Description

Density function of wrapped Cauchy distribution

Usage

```
dwrpcauchy(x, mu, rho, log = FALSE)
```

Arguments

`x`	Angle
`mu`	Mean parameter
`rho`	Concentration parameter
`log`	Should log-density be returned?

Value

Wrapped Cauchy density

`dwrpcauchy_rcpp` *Wrapped Cauchy density function*

Description

Probability density function of the wrapped Cauchy distribution (written in C++)

Usage

```
dwrpcauchy_rcpp(x, mu, rho)
```

Arguments

`x`	Vector of quantiles
`mu`	Mean
`rho`	Concentration

Value

Vector of densities

`elk_data` *Elk data set from Morales et al. (2004, Ecology)*

Description

It is a data frame with the following columns:

- `ID` Track identifier
- `Easting` Easting coordinate of locations
- `Northing` Northing coordinate of locations
- `dist_water` Distance of elk to water (in metres)

Usage

```
elk_data
```

`example` *Example dataset*

Description

This data is generated by the function `exGen`, and used in the examples and tests of other functions to keep them as short as possible.

Usage

```
example
```

Details

It is a list of the following objects:

- `data` A `moveData` object
- `m` A `moveHMM` object
- `simPar` The parameters used to simulate `data`
- `par0` The initial parameters in the optimization to fit `m`

`exGen` *Example data simulation*

Description

Generate the file `data/example.RData`, used in other functions' examples and unit tests.

Usage

```
exGen()
```

`fitHMM` *Fit an HMM to the data*

Description

Fit an hidden Markov model to the data provided, using numerical optimization of the log-likelihood function.

Usage

```
fitHMM(
  data,
  nbStates,
  stepPar0,
  anglePar0 = NULL,
  beta0 = NULL,
  delta0 = NULL,
  formula = ~1,
  stepDist = c("gamma", "weibull", "lnorm", "exp"),
  angleDist = c("vm", "wrpcauchy", "none"),
  angleMean = NULL,
  stationary = FALSE,
  knownStates = NULL,
  verbose = 0,
  nlmPar = NULL,
  fit = TRUE
)
```

Arguments

`data`	An object `moveData`.
`nbStates`	Number of states of the HMM.
`stepPar0`	Vector of initial state-dependent step length distribution parameters. The parameters should be in the order expected by the pdf of `stepDist`, and the zero-mass parameter should be the last. Note that zero-mass parameters are mandatory if there are steps of length zero in the data. For example, for a 2-state model using the gamma distribution and including zero-inflation, the vector of initial parameters would be something like: `c(mu1,mu2,sigma1,sigma2,zeromass1,zeromass2)`.
`anglePar0`	Vector of initial state-dependent turning angle distribution parameters. The parameters should be in the order expected by the pdf of `angleDist`. For example, for a 2-state model using the Von Mises (vm) distribution, the vector of initial parameters would be something like: `c(mu1,mu2,kappa1,kappa2)`.
`beta0`	Initial matrix of regression coefficients for the transition probabilities (more information in "Details"). Default: `NULL`. If not specified, `beta0` is initialized such that the diagonal elements of the transition probability matrix are dominant.
`delta0`	Initial value for the initial distribution of the HMM. Default: `rep(1/nbStates,nbStates)`.

`formula`	Regression formula for the covariates. Default: `~1` (no covariate effect).
`stepDist`	Name of the distribution of the step lengths (as a character string). Supported distributions are: gamma, weibull, lnorm, exp. Default: gamma.
`angleDist`	Name of the distribution of the turning angles (as a character string). Supported distributions are: vm, wrpcauchy. Set to `"none"` if the angle distribution should not be estimated. Default: vm.
`angleMean`	Vector of means of turning angles if not estimated (one for each state). Default: `NULL` (the angle mean is estimated).
`stationary`	`FALSE` if there are covariates. If `TRUE`, the initial distribution is considered equal to the stationary distribution. Default: `FALSE`.
`knownStates`	Vector of values of the state process which are known prior to fitting the model (if any). Default: NULL (states are not known). This should be a vector with length the number of rows of 'data'; each element should either be an integer (the value of the known states) or NA if the state is not known.
`verbose`	Determines the print level of the optimizer. The default value of 0 means that no printing occurs, a value of 1 means that the first and last iterations of the optimization are detailed, and a value of 2 means that each iteration of the optimization is detailed.
`nlmPar`	List of parameters to pass to the optimization function `nlm` (which should be either `'gradtol'`, `'stepmax'`, `'steptol'`, or `'iterlim'` – see `nlm`'s documentation for more detail)
`fit`	`TRUE` if an HMM should be fitted to the data, `FALSE` otherwise. If fit=`FALSE`, a model is returned with the MLE replaced by the initial parameters given in input. This option can be used to assess the initial parameters. Default: `TRUE`.

Details

- The matrix `beta` of regression coefficients for the transition probabilities has one row for the intercept, plus one row for each covariate, and one column for each non-diagonal element of the transition probability matrix. For example, in a 3-state HMM with 2 covariates, the matrix `beta` has three rows (intercept + two covariates) and six columns (six non-diagonal elements in the 3x3 transition probability matrix - filled in row-wise). In a covariate-free model (default), `beta` has one row, for the intercept.
- The choice of initial parameters is crucial to fit a model. The algorithm might not find the global optimum of the likelihood function if the initial parameters are poorly chosen.

Value

A `moveHMM` object, i.e. a list of:

`mle`	The maximum likelihood estimates of the parameters of the model (if the numerical algorithm has indeed identified the global maximum of the likelihood function), which is a list of: `stepPar` (step distribution parameters), `anglePar` (angle distribution parameters), `beta` (transition probabilities regression coefficients - more information in "Details"), and `delta` (initial distribution).
`data`	The movement data

`mod`	The object returned by the numerical optimizer `nlm`
`conditions`	A few conditions used to fit the model (`stepDist`, `angleDist`, `zeroInflation`, `estAngleMean`, `stationary`, and `formula`)
`rawCovs`	Raw covariate values, as found in the data (if any). Used in `plot.moveHMM`.
`knownStates`	Vector of states known a priori, as provided in input (if any, `NULL` otherwise). Used in `viterbi`,`logAlpha`, and `logBeta`
`nlmTime`	Computing time for optimisation, obtained with system.time

References

Patterson T.A., Basson M., Bravington M.V., Gunn J.S. 2009. Classifying movement behaviour in relation to environmental conditions using hidden Markov models. Journal of Animal Ecology, 78 (6), 1113-1123.

Langrock R., King R., Matthiopoulos J., Thomas L., Fortin D., Morales J.M. 2012. Flexible and practical modeling of animal telemetry data: hidden Markov models and extensions. Ecology, 93 (11), 2336-2342.

Examples

```
set.seed(872)
### 1. simulate data
# define all the arguments of simData
nbAnimals <- 2
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 <- 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 <- fitHMM(data=data,nbStates=nbStates,stepPar0=stepPar0,anglePar0=anglePar0,formula=formula,
```

```
              stepDist=stepDist,angleDist=angleDist,angleMean=angleMean)

print(m)
```

`getPalette` *Discrete colour palette for states*

Description

Discrete colour palette for states

Usage

```
getPalette(nbStates)
```

Arguments

`nbStates`	Number of states

Value

Vector of colours, of length nbStates.

`getPlotData` *Data to produce plots of fitted model*

Description

Data to produce plots of fitted model

Usage

```
getPlotData(m, type, format = "wide", alpha = 0.95)
```

Arguments

`m`	Fitted HMM object, as output by fitHMM.
`type`	Type of plot, one of: "dist", "tpm", "stat"
`format`	Format of data, either "wide" (for base graphics) or "long" (for ggplot)
`alpha`	Level of confidence intervals. Default: 0.95, i.e., 95% confidence intervals

Details

- If type = "dist", the function evaluates each state-dependent distribution over the range of observed variable (step length or turning angle), and weighs them by the proportion of time spent in each state (obtained from Viterbi state sequence).
- If type = "tpm", the function returns transition probabilities estimated over a range of covariate values. Other covariates are fixed to their mean values.

Value

Data frame (or list of data frames) containing data in a format that can easily be plotted. If type = "dist", the output is a list with two elements, "step" and "angle". If type = "tpm" or "stat", the output is a list with one element for each covariate. See details for more extensive description of output.

`haggis_data` *Wild haggis data set from Michelot et al. (2016, Methods Eco Evol)*

Description

Data frame of the first three tracks from Michelot et al. (2016), with columns:

- `ID` Track identifier
- `x` Easting coordinate of locations
- `y` Northing coordinate of locations
- `slope` Terrain slope (in degrees)
- `temp` Air temperature (in degrees Celsius)

Usage

```
haggis_data
```

`is.moveData` *Is moveData*

Description

Check that an object is of class `moveData`. Used in `fitHMM`.

Usage

```
is.moveData(x)
```

Arguments

`x`	An R object

Value

`TRUE` if `x` is of class `moveData`, `FALSE` otherwise.

`is.moveHMM` *Is moveHMM*

Description

Check that an object is of class `moveHMM`. Used in `CI`, `plotPR`, `plotStates`, `pseudoRes`, `stateProbs`, and `viterbi`.

Usage

```
is.moveHMM(x)
```

Arguments

`x`	An R object

Value

`TRUE` if `x` is of class `moveHMM`, `FALSE` otherwise.

`logAlpha` *Forward log-probabilities*

Description

Used in `stateProbs` and `pseudoRes`.

Usage

```
logAlpha(m)
```

Arguments

`m`	A `moveHMM` object.

Value

The matrix of forward log-probabilities.

Examples

```
## Not run:
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

la <- logAlpha(m)

## End(Not run)
```

`logBeta` *Backward log-probabilities*

Description

Used in `stateProbs`.

Usage

```
logBeta(m)
```

Arguments

`m`	A `moveHMM` object.

Value

The matrix of backward log-probabilities.

Examples

```
## Not run:
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

lb <- logBeta(m)

## End(Not run)
```

`moveData` *Constructor of* `moveData` *objects*

Description

This constructor is not intended to be used, except inside the function `prepData`. Refer to the documentation for that function.

Usage

```
moveData(data)
```

Arguments

`data`	A data frame with columns: `ID` (track ID(s)), `step` (step length), `angle` (turning angle, if any), `x` (Easting or longitude), `y` (Norting or latitude), and covariates (if any).

Value

An object of class `moveData`.

`moveHMM` *Constructor of* `moveHMM` *objects*

Description

This constructor is not intended to be used, except inside the function `fitHMM`. Refer to the documentation for that function.

Usage

```
moveHMM(m)
```

Arguments

`m`	A list of attributes of the fitted model: `mle` (maximum likelihood estimates of model parameters), `data` (movement data), `mod` (output of numerical optimizer `nlm`), `conditions` (few conditions used to fit the model: `stepDist`, `angleDist`, `zeroInflation`, `estAngleMean`, `stationary`, and `formula`), `rawCovs` (optional – only if there are covariates in the data).

Value

An object `moveHMM`.

`n2w` *Scaling function: natural to working parameters.*

Description

Scales each parameter from its natural interval to the set of real numbers, to allow for unconstrained optimization. Used during the optimization of the log-likelihood.

Usage

```
n2w(par, bounds, beta, delta = NULL, nbStates, estAngleMean)
```

Arguments

`par`	Vector of state-dependent distributions parameters.
`bounds`	Matrix with 2 columns and as many rows as there are elements in `par`. Each row contains the lower and upper bound for the correponding parameter.
`beta`	Matrix of regression coefficients for the transition probabilities.
`delta`	Initial distribution. Default: `NULL` ; if the initial distribution is not estimated.
`nbStates`	The number of states of the HMM.
`estAngleMean`	`TRUE` if the angle mean is estimated, `FALSE` otherwise.

Value

A vector of unconstrained parameters.

Examples

```
## Not run:
nbStates <- 3
par <- c(0.001,0.999,0.5,0.001,1500.3,7.1)
bounds <- matrix(c(0,1, # bounds for first parameter
                   0,1, # bounds for second parameter
                   0,1, # ...
                   0,Inf,
                   0,Inf,
                   0,Inf),
                 byrow=TRUE,ncol=2)
beta <- matrix(rnorm(18),ncol=6,nrow=3)
delta <- c(0.6,0.3,0.1)

# vector of working parameters
wpar <- n2w(par=par,bounds=bounds,beta=beta,delta=delta,nbStates=nbStates,
           estAngleMean=FALSE)

## End(Not run)
```

`nLogLike` *Negative log-likelihood function*

Description

Negative log-likelihood function

Usage

```
nLogLike(
  wpar,
  nbStates,
  bounds,
  parSize,
  data,
  covs,
  stepDist = c("gamma", "weibull", "lnorm", "exp"),
  angleDist = c("vm", "wrpcauchy", "none"),
  angleMean = NULL,
  zeroInflation = FALSE,
  stationary = FALSE,
  knownStates = NULL
)
```

Arguments

`wpar`	Vector of working parameters.
`nbStates`	Number of states of the HMM.
`bounds`	Matrix with 2 columns and as many rows as there are elements in `wpar`. Each row contains the lower and upper bound for the correponding parameter.
`parSize`	Vector of two values: number of parameters of the step length distribution, number of parameters of the turning angle distribution.
`data`	An object `moveData`.
`covs`	Model matrix for transition probabilities
`stepDist`	Name of the distribution of the step lengths (as a character string). Supported distributions are: gamma, weibull, lnorm, exp. Default: gamma.
`angleDist`	Name of the distribution of the turning angles (as a character string). Supported distributions are: vm, wrpcauchy. Set to `"none"` if the angle distribution should not be estimated. Default: vm.
`angleMean`	Vector of means of turning angles if not estimated (one for each state). Default: `NULL` (the angle mean is estimated).
`zeroInflation`	`TRUE` if the step length distribution is inflated in zero. Default: `FALSE`. If `TRUE`, initial values for the zero-mass parameters should be included in `stepPar0`.
`stationary`	`FALSE` if there are covariates. If `TRUE`, the initial distribution is considered equal to the stationary distribution. Default: `FALSE`.
`knownStates`	Vector of values of the state process which are known prior to fitting the model (if any). Default: NULL (states are not known). This should be a vector with length the number of rows of 'data'; each element should either be an integer (the value of the known states) or NA if the state is not known.

Value

The negative log-likelihood of the parameters given the data.

Examples

```
## Not run:
# data is a moveData object (as returned by prepData), automatically loaded with the package
simPar <- example$simPar
par0 <- example$par0

estAngleMean <- is.null(simPar$angleMean)
bounds <- moveHMM:::parDef(simPar$stepDist,simPar$angleDist,simPar$nbStates,
                           estAngleMean,simPar$zeroInflation)$bounds
parSize <- moveHMM:::parDef(simPar$stepDist,simPar$angleDist,simPar$nbStates,
                            estAngleMean,simPar$zeroInflation)$parSize

par <- c(par0$stepPar0,par0$anglePar0)
wpar <- moveHMM:::n2w(par,bounds,par0$beta0,par0$delta0,simPar$nbStates,FALSE)

covs <- model.matrix(example$m$conditions$formula, example$data)
```

```
l <- nLogLike(wpar=wpar,nbStates=simPar$nbStates,bounds=bounds,parSize=parSize,
              data=data,covs=covs, stepDist=simPar$stepDist,
              angleDist=simPar$angleDist,
              zeroInflation=simPar$zeroInflation)

## End(Not run)
```

`nLogLike_rcpp` *Negative log-likelihood*

Description

Computation of the negative log-likelihood (forward algorithm - written in C++)

Usage

```
nLogLike_rcpp(
  nbStates,
  beta,
  covs,
  data,
  stepDist,
  angleDist,
  stepPar,
  anglePar,
  delta,
  aInd,
  zeroInflation,
  stationary,
  knownStates
)
```

Arguments

`nbStates`	Number of states
`beta`	Matrix of regression coefficients for the transition probabilities
`covs`	Covariates
`data`	A `moveData` object of the observations
`stepDist`	The name of the step length distribution
`angleDist`	The name of the turning angle distribution
`stepPar`	State-dependent parameters of the step length distribution
`anglePar`	State-dependent parameters of the turning angle distribution
`delta`	Stationary distribution

`aInd`	Vector of indices of the rows at which the data switches to another animal
`zeroInflation`	`true` if zero-inflation is included in the step length distribution, `false` otherwise.
`stationary`	`false` if there are covariates. If `true`, the initial distribution is considered equal to the stationary distribution.
`knownStates`	Vector of values of the state process which are known prior to fitting the model (if any). Default: NULL (states are not known). This should be a vector with length the number of rows of 'data'; each element should either be an integer (the value of the known states) or NA if the state is not known.

Value

Negative log-likelihood

`parDef` *Parameters definition*

Description

Parameters definition

Usage

```
parDef(stepDist, angleDist, nbStates, estAngleMean, zeroInflation)
```

Arguments

`stepDist`	Name of the distribution of the step lengths.
`angleDist`	Name of the distribution of the turning angles. Set to `"none"` if the angle distribution should not be estimated.
`nbStates`	Number of states of the HMM.
`estAngleMean`	`TRUE` if the mean of the turning angles distribution is estimated, `FALSE` otherwise.
`zeroInflation`	`TRUE` if the step length distribution is inflated in zero.

Value

A list of:

`parSize`	Vector of two values: number of parameters of the step length distribution, number of parameters of the turning angle distribution
`bounds`	Matrix with 2 columns and `sum(parSize)` rows - each row contains the lower and upper bound for the correponding parameter)
`parNames`	Names of parameters of step distribution (the names of the parameters of the angle distribution are always the same).

`plot.moveData` *Plot* `moveData`

Description

Plot `moveData`

Usage

```
## S3 method for class 'moveData'
plot(x, animals = NULL, compact = FALSE, ask = TRUE, breaks = "Sturges", ...)
```

Arguments

`x`	An object `moveData`
`animals`	Vector of indices or IDs of animals for which information will be plotted. Default: `NULL` ; all animals are plotted.
`compact`	`TRUE` for a compact plot (all individuals at once), `FALSE` otherwise (default – one individual at a time).
`ask`	If `TRUE`, the execution pauses between each plot.
`breaks`	Histogram parameter. See `hist` documentation.
`...`	Currently unused. For compatibility with generic method.

Examples

```
# data is a moveData object (as returned by prepData), automatically loaded with the package
data <- example$data

plot(data,compact=TRUE,breaks=20,ask=FALSE)
```

`plot.moveHMM` *Plot* `moveHMM`

Description

Plot the fitted step and angle densities over histograms of the data, transition probabilities as functions of the covariates, and maps of the animals' tracks colored by the decoded states.

Usage

```
## S3 method for class 'moveHMM'
plot(
  x,
  animals = NULL,
  ask = TRUE,
  breaks = "Sturges",
  col = NULL,
  plotTracks = TRUE,
  plotCI = FALSE,
  alpha = 0.95,
  ...
)
```

Arguments

`x`	Object `moveHMM`
`animals`	Vector of indices or IDs of animals for which information will be plotted. Default: `NULL`; all animals are plotted.
`ask`	If `TRUE`, the execution pauses between each plot.
`breaks`	Histogram parameter. See `hist` documentation. See `hist` documentation. Default: `NULL` ; the function sets default values.
`col`	Vector or colors for the states (one color per state).
`plotTracks`	If `TRUE`, the Viterbi-decoded tracks are plotted (default).
`plotCI`	If `TRUE`, confidence intervals are plotted on the transition probabilities (default: FALSE).
`alpha`	Significance level of the confidence intervals if plotCI=TRUE. Default: 0.95 (i.e. 95% CIs).
`...`	Currently unused. For compatibility with generic method.

Details

The state-dependent densities are weighted by the frequency of each state in the most probable state sequence (decoded with the function `viterbi`). For example, if the most probable state sequence indicates that one third of observations correspond to the first state, and two thirds to the second state, the plots of the densities in the first state are weighted by a factor 1/3, and in the second state by a factor 2/3.

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

plot(m,ask=TRUE,animals=1,breaks=20)
```

`plotPR` *Plot pseudo-residuals*

Description

Plots time series, qq-plots (against the standard normal distribution), and sample ACF functions of the pseudo-residuals

Usage

```
plotPR(m)
```

Arguments

`m`	A `moveHMM` object

Details

- If some turning angles in the data are equal to pi, the corresponding pseudo-residuals will not be included. Indeed, given that the turning angles are defined on (-pi,pi], an angle of pi results in a pseudo-residual of +Inf (check Section 6.2 of reference for more information on the computation of pseudo-residuals).
- If some steps are of length zero (i.e. if there is zero-inflation), the corresponding pseudo-residuals are shown as segments, because pseudo-residuals for discrete data are defined as segments (see Zucchini and MacDonald, 2009, Section 6.2).

References

Zucchini, W. and MacDonald, I.L. 2009. Hidden Markov Models for Time Series: An Introduction Using R. Chapman & Hall (London).

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

plotPR(m)
```

`plotSat` *Plot observations on satellite image*

Description

Plot tracking data on a satellite map. This function only works with longitude and latitude values (not with UTM coordinates), and uses the package `ggmap` to fetch a satellite image from Google. An Internet connection is required to use this function.

Usage

```
plotSat(
  data,
  zoom = NULL,
  location = NULL,
  segments = TRUE,
  compact = TRUE,
  col = NULL,
  alpha = 1,
  size = 1,
  states = NULL,
  animals = NULL,
  ask = TRUE,
  return = FALSE
)
```

Arguments

`data`	Data frame of the data, with necessary fields 'x' (longitude values) and 'y' (latitude values).
`zoom`	The zoom level, as defined for `get_map`. Integer value between 3 (continent) and 21 (building).
`location`	Location of the center of the map to be plotted.
`segments`	`TRUE` if segments should be plotted between the observations (default), `FALSE` otherwise.
`compact`	`FALSE` if tracks should be plotted separately, `TRUE` otherwise (default).
`col`	Palette of colours to use for the dots and segments. If not specified, uses default palette.
`alpha`	Transparency argument for `geom_point`.
`size`	Size argument for `geom_point`.
`states`	A sequence of integers, corresponding to the decoded states for these data (such that the observations are colored by states).
`animals`	Vector of indices or IDs of animals/tracks to be plotted. Default: `NULL`; all animals are plotted.

`ask`	If `TRUE`, the execution pauses between each plot.
`return`	If `TRUE`, the function returns a ggplot object (which can be edited and plotted manually). If `FALSE`, the function automatically plots the map (default).

Details

If the plot displays the message "Sorry, we have no imagery here", try a lower level of zoom.

References

D. Kahle and H. Wickham. ggmap: Spatial Visualization with ggplot2. The R Journal, 5(1), 144-161. URL: http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf

---

`plotStates` *Plot states*

Description

Plot the states and states probabilities.

Usage

```
plotStates(m, animals = NULL, ask = TRUE)
```

Arguments

`m`	A `moveHMM` object
`animals`	Vector of indices or IDs of animals for which states will be plotted.
`ask`	If `TRUE`, the execution pauses between each plot.

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

# plot states for first and second animals
plotStates(m,animals=c(1,2))
```

`plotStationary` *Plot stationary state probabilities*

Description

Plot stationary state probabilities

Usage

```
plotStationary(m, col = NULL, plotCI = FALSE, alpha = 0.95)
```

Arguments

`m`	An object `moveHMM`
`col`	Vector or colors for the states (one color per state).
`plotCI`	Logical. Should 95% confidence intervals be plotted? (Default: FALSE)
`alpha`	Significance level of the confidence intervals if plotCI=TRUE. Default: 0.95 (i.e. 95% CIs).

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

plotStationary(m)
```

`predictStationary` *Predict stationary state probabilities*

Description

Predict stationary state probabilities

Usage

```
predictStationary(
  m,
  newData,
  beta = m$mle$beta,
  returnCI = FALSE,
  alpha = 0.95
)
```

Arguments

`m`	Fitted moveHMM object, as returned by `fitHMM`
`newData`	Data frame with columns for the covariates
`beta`	Optional matrix of regression coefficients for the transition probability model. By default, uses estimates in `m`.
`returnCI`	Logical indicating whether confidence intervals should be returned. Default: FALSE.
`alpha`	Confidence level if returnCI = TRUE. Default: 0.95, i.e., 95% confidence intervals.

Value

List with elements 'mle', 'lci', and 'uci' (the last two only if returnCI = TRUE). Each element is a matrix of stationary state probabilities with one row for each row of newData and one column for each state.

`predictTPM` *Predict transition probabilities for new covariate values*

Description

Predict transition probabilities for new covariate values

Usage

```
predictTPM(m, newData, beta = m$mle$beta, returnCI = FALSE, alpha = 0.95)
```

Arguments

`m`	Fitted moveHMM object, as returned by `fitHMM`
`newData`	Data frame with columns for the covariates
`beta`	Optional matrix of regression coefficients for the transition probability model. By default, uses estimates in `m`.
`returnCI`	Logical indicating whether confidence intervals should be returned. Default: FALSE.
`alpha`	Confidence level if returnCI = TRUE. Default: 0.95, i.e., 95% confidence intervals.

Value

List with elements 'mle', 'lci', and 'uci' (the last two only if returnCI = TRUE). Each element is an array, where each layer is a transition probability matrix corresponding to a row of newData.

`prepData` *Preprocessing of the tracking data*

Description

Preprocessing of the tracking data

Usage

```
prepData(
  trackData,
  type = c("LL", "UTM"),
  coordNames = c("x", "y"),
  LLangle = NULL
)
```

Arguments

`trackData`	A dataframe of the tracking data, including at least coordinates (either longitude/latitude values or cartesian coordinates), and optionnaly a field `ID` (identifiers for the observed individuals). Additionnal fields are considered as covariates. Note that, if the names of the coordinates are not "x" and "y", the `coordNames` argument should specified. Tracking data should be structured so that the rows for each track (or each animal) are grouped together, and ordered by date, in the data frame.
`type`	`'LL'` if longitude/latitude provided (default), `'UTM'` if easting/northing.
`coordNames`	Names of the columns of coordinates in the data frame. Default: `c("x","y")`.
`LLangle`	Logical. If TRUE, the turning angle is calculated with `geosphere::bearing` (default), else calculated with `atan2`.

Value

An object `moveData`, i.e. a dataframe of:

`ID`	The ID(s) of the observed animal(s)
`step`	The step lengths - in kilometers if longitude/latitude provided, and in the metrics of the data otherwise
`angle`	The turning angles (if any) - in radians
`x`	Either Easting or longitude (or e.g. depth for 1D data)
`y`	Either Northing or latitude (all zero if 1D data)
`...`	Covariates (if any)

Examples

```
coord1 <- c(1,2,3,4,5,6,7,8,9,10)
coord2 <- c(1,1,1,2,2,2,1,1,1,2)
trackData <- data.frame(coord1=coord1,coord2=coord2)
d <- prepData(trackData,type='UTM',coordNames=c("coord1","coord2"))
```

`print.moveHMM` *Print* `moveHMM`

Description

Print `moveHMM`

Usage

```
## S3 method for class 'moveHMM'
print(x, ...)
```

Arguments

`x`	A `moveHMM` object.
`...`	Currently unused. For compatibility with generic method.

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

print(m)
```

`pseudoRes` *Pseudo-residuals*

Description

The pseudo-residuals of a moveHMM model, as described in Zucchini and McDonad (2009).

Usage

```
pseudoRes(m)
```

Arguments

`m`	A `moveHMM` object.

Details

If some turning angles in the data are equal to pi, the corresponding pseudo-residuals will not be included. Indeed, given that the turning angles are defined on (-pi,pi], an angle of pi results in a pseudo-residual of +Inf (check Section 6.2 of reference for more information on the computation of pseudo-residuals).

Value

A list of:

`stepRes`	The pseudo-residuals for the step lengths
`angleRes`	The pseudo-residuals for the turning angles

References

Zucchini, W. and MacDonald, I.L. 2009. Hidden Markov Models for Time Series: An Introduction Using R. Chapman & Hall (London).

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m
res <- pseudoRes(m)
qqnorm(res$stepRes)
qqnorm(res$angleRes)
```

`rvm` *Sample from von Mises distribution*

Description

Sample from von Mises distribution

Usage

```
rvm(n, mu, kappa)
```

Arguments

`n`	Number of samples
`mu`	Mean parameter
`kappa`	Concentration parameter

Details

Uses basic rejection sampling, based on dvm(), which might be inefficient for large kappa. Could be improved following Best & Fisher (1979), Efficient simulation of the von Mises distribution, JRSSC, 28(2), 152-157.

Value

Vector of n samples from vm(mu, kappa)

`rwrpcauchy` *Sample from wrapped Cauchy distribution*

Description

Sample from wrapped Cauchy distribution

Usage

```
rwrpcauchy(n, mu, rho)
```

Arguments

`n`	Number of samples
`mu`	Mean parameter
`rho`	Concentration parameter

Details

Uses basic rejection sampling, based on dwrpcauchy(), which might be inefficient for large rho.

Value

Vector of n samples from wrpcauchy(mu, rho)

`simData` *Simulation tool*

Description

Simulates movement data from an HMM.

Usage

```
simData(
  nbAnimals = 1,
  nbStates = 2,
  stepDist = c("gamma", "weibull", "lnorm", "exp"),
  angleDist = c("vm", "wrpcauchy", "none"),
  stepPar = NULL,
  anglePar = NULL,
  beta = NULL,
  covs = NULL,
  nbCovs = 0,
  zeroInflation = FALSE,
  obsPerAnimal = c(500, 1500),
  model = NULL,
  states = FALSE
)
```

Arguments

`nbAnimals`	Number of observed individuals to simulate.
`nbStates`	Number of behavioural states to simulate.
`stepDist`	Name of the distribution of the step lengths (as a character string). Supported distributions are: gamma, weibull, lnorm, exp. Default: gamma.
`angleDist`	Name of the distribution of the turning angles (as a character string). Supported distributions are: vm, wrpcauchy. Set to `"none"` if the angle distribution should not be estimated. Default: vm.
`stepPar`	Parameters of the step length distribution.
`anglePar`	Parameters of the turning angle distribution.
`beta`	Matrix of regression parameters for the transition probabilities (more information in "Details").
`covs`	Covariate values to include in the model, as a dataframe. Default: `NULL`. Covariates can also be simulated according to a standard normal distribution, by setting `covs` to `NULL`, and specifying `nbCovs>0`.
`nbCovs`	Number of covariates to simulate (0 by default). Does not need to be specified of `covs` is specified.
`zeroInflation`	`TRUE` if the step length distribution is inflated in zero. Default: `FALSE`. If `TRUE`, values for the zero-mass parameters should be included in `stepPar`.
`obsPerAnimal`	Either the number of the number of observations per animal (if single value), or the bounds of the number of observations per animal (if vector of two values). In the latter case, the numbers of obervations generated for each animal are uniformously picked from this interval. Default: `c(500,1500)`.
`model`	A moveHMM object. This option can be used to simulate from a fitted model. Default: NULL. Note that, if this argument is specified, most other arguments will be ignored – except for nbAnimals, obsPerAnimal, covs (if covariate values different from those in the data should be specified), and states.
`states`	`TRUE` if the simulated states should be returned, `FALSE` otherwise (default).

Details

- The matrix `beta` of regression coefficients for the transition probabilities has one row for the intercept, plus one row for each covariate, and one column for each non-diagonal element of the transition probability matrix. For example, in a 3-state HMM with 2 covariates, the matrix `beta` has three rows (intercept + two covariates) and six columns (six non-diagonal elements in the 3x3 transition probability matrix - filled in row-wise). In a covariate-free model (default), `beta` has one row, for the intercept.
- If the length of covariate values passed (either through 'covs', or 'model') is not the same as the number of observations suggested by 'nbAnimals' and 'obsPerAnimal', then the series of covariates is either shortened (removing last values - if too long) or extended (starting over from the first values - if too short).

Value

An object moveData, i.e. a dataframe of:

`ID`	The ID(s) of the observed animal(s)
`step`	The step lengths
`angle`	The turning angles (if any)
`x`	Either easting or longitude
`y`	Either northing or latitude
`...`	Covariates (if any)

Examples

```
# 1. Pass a fitted model to simulate from
# (m is a moveHMM object - as returned by fitHMM - automatically loaded with the package)
# We keep the default nbAnimals=1.
m <- example$m
obsPerAnimal=c(50,100)
data <- simData(model=m,obsPerAnimal=obsPerAnimal)

# 2. Pass the parameters of the model to simulate from
stepPar <- c(1,10,1,5,0.2,0.3) # mean1, mean2, sd1, sd2, z1, z2
anglePar <- c(pi,0,0.5,2) # mean1, mean2, k1, k2
stepDist <- "gamma"
angleDist <- "vm"
data <- simData(nbAnimals=5,nbStates=2,stepDist=stepDist,angleDist=angleDist,stepPar=stepPar,
                anglePar=anglePar,nbCovs=2,zeroInflation=TRUE,obsPerAnimal=obsPerAnimal)

stepPar <- c(1,10,1,5) # mean1, mean2, sd1, sd2
anglePar <- c(pi,0,0.5,0.7) # mean1, mean2, k1, k2
stepDist <- "weibull"
angleDist <- "wrpcauchy"
data <- simData(nbAnimals=5,nbStates=2,stepDist=stepDist,angleDist=angleDist,stepPar=stepPar,
                anglePar=anglePar,obsPerAnimal=obsPerAnimal)

# step length only and zero-inflation
stepPar <- c(1,10,1,5,0.2,0.3) # mean1, mean2, sd1, sd2, z1, z2
```

```
stepDist <- "gamma"
data <- simData(nbAnimals=5,nbStates=2,stepDist=stepDist,angleDist="none",stepPar=stepPar,
               nbCovs=2,zeroInflation=TRUE,obsPerAnimal=obsPerAnimal)

# include covariates
# (note that it is useless to specify "nbCovs", which is determined
# by the number of columns of "cov")
cov <- data.frame(temp=rnorm(500,20,5))
stepPar <- c(1,10,1,5) # mean1, mean2, sd1, sd2
anglePar <- c(pi,0,0.5,2) # mean1, mean2, k1, k2
stepDist <- "gamma"
angleDist <- "vm"
data <- simData(nbAnimals=5,nbStates=2,stepDist=stepDist,angleDist=angleDist,stepPar=stepPar,
                anglePar=anglePar,covs=cov)
```

`splitAtGaps` *Split track at gaps*

Description

Defines a new ID variable, which changes where there is a long gap in the data. This is sometimes useful for preprocessing data prior to using `prepData`.

Usage

```
splitAtGaps(data, maxGap = 60, shortestTrack = 0, units = "mins")
```

Arguments

`data`	Data frame with (at least) columns for "ID" and "time"
`maxGap`	Longest allowed gap, in minutes by default (but see argument "units"). Track will be split at longer gaps.
`shortestTrack`	Shortest track to keep after splitting, in minutes. Shorter tracks will be removed from the output data set.
`units`	Character string, e.g., "mins" (default), "secs", "hours", "days". Time units used for the maxGap and shortestTrack arguments.

Value

Data frame with identical structure as input, where ID column has been replaced by new ID for split tracks. Old ID still accessible as ID_old column

---

`stateProbs` *State probabilities*

Description

For a given model, computes the probability of the process being in the different states at each time point.

Usage

```
stateProbs(m)
```

Arguments

`m`	A `moveHMM` object.

Value

The matrix of state probabilities, with element [i,j] the probability of being in state j in observation i.

References

Zucchini, W. and MacDonald, I.L. 2009. Hidden Markov Models for Time Series: An Introduction Using R. Chapman & Hall (London).

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

sp <- stateProbs(m)
```

`stationary` *Stationary state probabilities*

Description

Calculates the stationary probabilities of each state, for given covariate values.

Usage

```
stationary(m, covs, beta = m$mle$beta)
```

Arguments

`m`	Fitted model (as output by `fitHMM`).
`covs`	Either a data frame or a design matrix of covariates.
`beta`	Optional matrix of regression coefficients for the transition probability model. By default, uses estimates in `m`.

Value

Matrix of stationary state probabilities. Each row corresponds to a row of covs, and each column corresponds to a state.

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

# data frame of covariates
stationary(m, covs = data.frame(cov1 = 0, cov2 = 0))

# design matrix (each column corresponds to row of m$mle$beta)
stationary(m, covs = matrix(c(1,0,cos(0)),1,3))
```

`summary.moveData` *Summary* `moveData`

Description

Summary `moveData`

Usage

```
## S3 method for class 'moveData'
summary(object, ...)
```

Arguments

`object`	A `moveData` object.
`...`	Currently unused. For compatibility with generic method.

Examples

```
# Use example data set as output by prepData
summary(example$data)
```

`trMatrix_rcpp` *Transition probability matrix*

Description

Computation of the transition probability matrix, as a function of the covariates and the regression parameters. Written in C++. Used in `fitHMM`, `logAlpha`, `logBeta`, `plot.moveHMM`, `pseudoRes`, and `viterbi`.

Usage

```
trMatrix_rcpp(nbStates, beta, covs)
```

Arguments

`nbStates`	Number of states
`beta`	Matrix of regression parameters
`covs`	Matrix of covariate values

Value

Three dimensional array `trMat`, such that `trMat[,,t]` is the transition matrix at time t.

`turnAngle` *Turning angle*

Description

Used in `prepData`.

Usage

```
turnAngle(x, y, z, LLangle)
```

Arguments

`x`	First point
`y`	Second point
`z`	Third point
`LLangle`	Logical. If TRUE, the turning angle is calculated with `geosphere::bearing`, else calculated with `atan2`.

Value

The angle between vectors (x,y) and (y,z)

Examples

```
## Not run:
x <- c(0,0)
y <- c(4,6)
z <- c(10,7)
turnAngle(x,y,z,LLangle=FALSE)

## End(Not run)
```

`viterbi` *Viterbi algorithm*

Description

For a given model, reconstructs the most probable states sequence, using the Viterbi algorithm.

Usage

```
viterbi(m, newdata = NULL)
```

Arguments

`m`	An object `moveHMM`
`newdata`	An object `moveData` (optional)

Value

The sequence of most probable states.

References

Zucchini, W. and MacDonald, I.L. 2009. Hidden Markov Models for Time Series: An Introduction Using R. Chapman & Hall (London).

Examples

```
# m is a moveHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

# reconstruction of states sequence
states <- viterbi(m)
```

`w2n` *Scaling function: working to natural parameters*

Description

Scales each parameter from the set of real numbers, back to its natural interval. Used during the optimization of the log-likelihood.

Usage

```
w2n(wpar, bounds, parSize, nbStates, nbCovs, estAngleMean, stationary)
```

Arguments

`wpar`	Vector of state-dependent distributions unconstrained parameters.
`bounds`	Matrix with 2 columns and as many rows as there are elements in `wpar`. Each row contains the lower and upper bound for the correponding parameter.
`parSize`	Vector of two values: number of parameters of the step length distribution, number of parameters of the turning angle distribution.
`nbStates`	The number of states of the HMM.
`nbCovs`	The number of covariates.
`estAngleMean`	`TRUE` if the angle mean is estimated, `FALSE` otherwise.
`stationary`	`FALSE` if there are covariates. If TRUE, the initial distribution is considered equal to the stationary distribution. Default: `FALSE`.

Value

A list of:

`stepPar`	Matrix of natural parameters of the step length distribution
`anglePar`	Matrix of natural parameters of the turning angle distribution
`beta`	Matrix of regression coefficients of the transition probabilities
`delta`	Initial distribution

Examples

```
## Not run:
nbStates <- 3
nbCovs <- 2
par <- c(0.001,0.999,0.5,0.001,1500.3,7.1)
parSize <- c(1,1)
bounds <- matrix(c(0,1,0,1,0,1,
                   0,Inf,0,Inf,0,Inf),
                 byrow=TRUE,ncol=2)
beta <- matrix(rnorm(18),ncol=6,nrow=3)
delta <- c(0.6,0.3,0.1)
```

```
wpar <- n2w(par,bounds,beta,delta,nbStates,FALSE)
print(w2n(wpar,bounds,parSize,nbStates,nbCovs,estAngleMean=FALSE,stationary=FALSE))

## End(Not run)
```

Index