

# Package ‘momentuHMM’

September 3, 2021

**Type** Package

**Title** Maximum Likelihood Analysis of Animal Movement Behavior Using Multivariate Hidden Markov Models

**Version** 1.5.4

**Date** 2021-09-02

**Depends** R (>= 2.10)

**Author** Brett McClintock, Theo Michelot

**Maintainer** Brett McClintock <brett.mcclintock@noaa.gov>

**Description** Extended tools for analyzing telemetry data using generalized hidden Markov models. Features of momentuHMM (pronounced “momentum”) include data pre-processing and visualization, fitting HMMs to location and auxiliary biotelemetry or environmental data, biased and correlated random walk movement models, hierarchical HMMs, multiple imputation for incorporating location measurement error and missing data, user-specified design matrices and constraints for covariate modelling of parameters, random effects, decoding of the state process, visualization of fitted models, model checking and selection, and simulation. See McClintock and Michelot (2018) <[doi:10.1111/2041-210X.12995](https://doi.org/10.1111/2041-210X.12995)>.

**License** GPL-3

**LazyData** TRUE

**Imports** Rcpp, doParallel, foreach, numDeriv, CircStats, crawl (>= 2.2.1), mvtnorm, sp, MASS, Brodidingnag, doRNG, rlang

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** testthat, setRNG, splines, splines2 (>= 0.2.8), R.rsp, raster, conicfit, ggplot2, ggmap, lubridate, dplyr, magrittr, scatterplot3d, BB, expm, matrixcalc, moveHMM, extraDistr, data.tree (>= 1.0.0), geosphere, mitools, doFuture, future, car, survival, proclim, nleqslv, qdapRegex

**RoxygenNote** 7.1.1

**Encoding** UTF-8

**NeedsCompilation** yes

**VignetteBuilder** R.rsp

**URL** <https://github.com/bmcclintock/momentuHMM>,  
<https://github.com/bmcclintock/momentuHMM/discussions>

**BugReports** <https://github.com/bmcclintock/momentuHMM/issues>

**Repository** CRAN

**Date/Publication** 2021-09-03 04:30:02 UTC

## R topics documented:

AIC.momentuHMM	4
AICweights	5
allProbs	7
checkPar0	7
CIbeta	11
circAngles	12
CIreal	13
crawlMerge	15
crawlWrap	16
crwData	20
crwHierData	20
crwHierSim	21
crwSim	21
dbern_rcpp	22
dbeta_rcpp	22
dcat_rcpp	23
dexp_rcpp	23
dgamma_rcpp	24
distAngle	24
dlnorm_rcpp	25
dlogis_rcpp	26
dmvnorm_rcpp	26
dnbinom_rcpp	27
dnorm_rcpp	27
dpois_rcpp	28
dt_rcpp	28
dvm_rcpp	29
dweibull_rcpp	29
dwrpcauchy_rcpp	30
exampleData	30
expandPar	31
fitHMM	33
formatHierHMM	47
getCovNames	49
getDM_rcpp	49
getPar	50
getPar0	51
getParDM	54

getTrProbs . . . . .	58
HMMfits . . . . .	61
is.crwData . . . . .	61
is.crwHierData . . . . .	62
is.crwHierSim . . . . .	62
is.crwSim . . . . .	63
is.HMMfits . . . . .	63
is.miHMM . . . . .	64
is.miSum . . . . .	64
is.momentuHierHMM . . . . .	65
is.momentuHierHMMDData . . . . .	65
is.momentuHMM . . . . .	66
is.momentuHMMDData . . . . .	66
logAlpha . . . . .	67
logBeta . . . . .	67
MIfitHMM . . . . .	68
miHMM . . . . .	76
MIpool . . . . .	77
miSum . . . . .	79
mixtureProbs . . . . .	79
momentuHierHMM . . . . .	80
momentuHierHMMDData . . . . .	81
momentuHMM . . . . .	82
momentuHMMDData . . . . .	82
n2w . . . . .	83
nLogLike . . . . .	84
nLogLike_rcpp . . . . .	87
parDef . . . . .	88
plot.crwData . . . . .	90
plot.miHMM . . . . .	91
plot.miSum . . . . .	93
plot.momentuHMM . . . . .	95
plot.momentuHMMDData . . . . .	97
plotPR . . . . .	98
plotSat . . . . .	99
plotSpatialCov . . . . .	101
plotStates . . . . .	102
plotStationary . . . . .	103
prepData . . . . .	104
print.miHMM . . . . .	108
print.miSum . . . . .	109
print.momentuHMM . . . . .	110
pseudoRes . . . . .	110
randomEffects . . . . .	111
setModelName . . . . .	114
setStateNames . . . . .	114
simData . . . . .	115
simObsData . . . . .	127

stateProbs . . . . .	130
stationary . . . . .	130
summary.momentuHMMDData . . . . .	131
timeInStates . . . . .	132
trMatrix_rcpp . . . . .	133
turnAngle . . . . .	134
viterbi . . . . .	135
w2n . . . . .	135
XBloop_rcpp . . . . .	138

<b>Index</b>	<b>139</b>
--------------	------------

---

AIC.momentuHMM	<i>AIC</i>
----------------	------------

---

## Description

Akaike information criterion of momentuHMM model(s).

## Usage

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

## Arguments

object	A momentuHMM object.
...	Optional additional momentuHMM objects, to compare AICs of the different models. These can be passed as a list using the !!! operator (see <a href="#">rlang</a> and example in <a href="#">AICweights</a> ).
k	Penalty per parameter. Default: 2 ; for classical AIC.
n	Optional sample size. If specified, the small sample correction AIC is used (i.e., $AIC_c = AIC + k p(p+1) / (n-p-1)$ where p is the number of parameters).

## Value

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

## Examples

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

## Not run:
# HMM specifications
nbStates <- 2
```

```

stepDist <- "gamma"
angleDist <- "vm"
mu0 <- c(20,70)
sigma0 <- c(10,30)
kappa0 <- c(1,1)
stepPar0 <- c(mu0,sigma0)
anglePar0 <- c(-pi/2,pi/2,kappa0)
formula <- ~cov1+cov2

# example$m is a momentuHMM object (as returned by fitHMM), automatically loaded with the package
mod1 <- fitHMM(example$m$data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
               Par0=list(step=stepPar0,angle=anglePar0),
               formula=~1,estAngleMean=list(angle=TRUE))

Par0 <- getPar0(mod1,formula=formula)
mod2 <- fitHMM(example$m$data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
               Par0=Par0$Par,beta0=Par0$beta,
               formula=formula,estAngleMean=list(angle=TRUE))

AIC(mod1,mod2)

Par0nA <- getPar0(mod1,estAngleMean=list(angle=FALSE))
mod3 <- fitHMM(example$m$data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
               Par0=Par0nA$Par,beta0=Par0nA$beta,
               formula=~1)

AIC(mod1,mod2,mod3)

# add'l models provided as a list using the !!! operator
AIC(mod1, !!!list(mod2,mod3))

## End(Not run)

```

---

AICweights

*Calculate Akaike information criterion model weights*


---

## Description

Calculate Akaike information criterion model weights

## Usage

```
AICweights(..., k = 2, n = NULL)
```

## Arguments

... [momentuHMM](#), [HMMfits](#), or [miHMM](#) objects, to compare AIC weights of the different models. The first object must be a [momentuHMM](#), [HMMfits](#), or [miHMM](#) object, but additional model objects can be passed as a list using the `!!!` operator (see [rlang](#)).

k	Penalty per parameter. Default: 2 ; for classical AIC.
n	Optional sample size. If specified, the small sample correction AIC is used (i.e., $AICc = AIC + kp(p+1)/(n-p-1)$ where p is the number of parameters).

### Details

- Model objects must all be either of class `momentuHMM` or multiple imputation model objects (of class `HMMfits` and/or `miHMM`).
- AIC is only valid for comparing models fitted to the same data. The data for each model fit must therefore be identical. For multiple imputation model objects, respective model fits must have identical data.

### Value

The AIC weights of the models. If multiple imputation objects are provided, then the mean model weights (and standard deviations) are provided.

### Examples

```
## Not run:
# HMM specifications
nbStates <- 2
stepDist <- "gamma"
angleDist <- "vm"
mu0 <- c(20,70)
sigma0 <- c(10,30)
kappa0 <- c(1,1)
stepPar0 <- c(mu0,sigma0)
anglePar0 <- c(-pi/2,pi/2,kappa0)
formula <- ~cov1+cov2

# example$m is a momentuHMM object (as returned by fitHMM), automatically loaded with the package
mod1 <- fitHMM(example$m$data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
               Par0=list(step=stepPar0,angle=anglePar0),
               formula=~1,estAngleMean=list(angle=TRUE))

Par0 <- getPar0(mod1,formula=formula)
mod2 <- fitHMM(example$m$data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
               Par0=Par0$Par,beta0=Par0$beta,
               formula=formula,estAngleMean=list(angle=TRUE))

AICweights(mod1,mod2)

Par0nA <- getPar0(mod1,estAngleMean=list(angle=FALSE))
mod3 <- fitHMM(example$m$data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
               Par0=Par0nA$Par,beta0=Par0nA$beta,
               formula=~1)

AICweights(mod1,mod2,mod3)
```

```
# add'l models provided as a list using the !!! operator
AICweights(mod1, !!!list(mod2,mod3))

## End(Not run)
```

---

allProbs	<i>Matrix of all probabilities</i>
----------	------------------------------------

---

### Description

Used in functions [viterbi](#), [logAlpha](#), [logBeta](#).

### Usage

```
allProbs(m)
```

### Arguments

`m` Object [momentuHMM](#) or [miSum](#).

### Value

Matrix of all probabilities.

### Examples

```
## Not run:
P <- momentuHMM:::allProbs(m=example$m)

## End(Not run)
```

---

checkPar0	<i>Check parameter length and order for a <a href="#">fithMM</a> (or <a href="#">MIfithMM</a>) model</i>
-----------	--

---

### Description

Prints parameters with labels based on DM, formula, and/or formulaDelta. See [fithMM](#) for further argument details.

**Usage**

```
checkPar0(data, ...)  
  
## Default S3 method:  
checkPar0(  
  data,  
  nbStates,  
  dist,  
  Par0 = NULL,  
  beta0 = NULL,  
  delta0 = NULL,  
  estAngleMean = NULL,  
  circularAngleMean = NULL,  
  formula = ~1,  
  formulaDelta = NULL,  
  stationary = FALSE,  
  mixtures = 1,  
  formulaPi = NULL,  
  DM = NULL,  
  userBounds = NULL,  
  workBounds = NULL,  
  betaCons = NULL,  
  betaRef = NULL,  
  deltaCons = NULL,  
  stateNames = NULL,  
  fixPar = NULL,  
  prior = NULL,  
  ...  
)  
  
## S3 method for class 'hierarchical'  
checkPar0(  
  data,  
  hierStates,  
  hierDist,  
  Par0 = NULL,  
  hierBeta = NULL,  
  hierDelta = NULL,  
  estAngleMean = NULL,  
  circularAngleMean = NULL,  
  hierFormula = NULL,  
  hierFormulaDelta = NULL,  
  mixtures = 1,  
  formulaPi = NULL,  
  DM = NULL,  
  userBounds = NULL,  
  workBounds = NULL,  
  betaCons = NULL,
```



```

    deltaCons = NULL,
    fixPar = NULL,
    prior = NULL,
    ...
)

```

## Arguments

data	<a href="#">momentuHMMDData</a> object, <a href="#">momentuHierHMMDData</a> object, or a data frame containing the data stream and covariate values
...	further arguments passed to or from other methods
nbStates	Number of states of the HMM.
dist	A named list indicating the probability distributions of the data streams.
Par0	Optional named list containing vectors of state-dependent probability distribution parameters for each data stream specified in <code>dist</code> . If <code>Par0</code> is not provided, then ordered parameter indices are returned.
beta0	Optional matrix of regression coefficients for the transition probabilities. If <code>beta0</code> is not provided, then ordered parameter indices are returned.
delta0	Optional values or regression coefficients for the initial distribution of the HMM. If <code>delta0</code> is not provided, then ordered parameter indices are returned.
estAngleMean	An optional named list indicating whether or not to estimate the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy').
circularAngleMean	An optional named list indicating whether to use circular-linear or circular-circular regression on the mean of circular distributions ('vm' and 'wrpcauchy') for turning angles.
formula	Regression formula for the transition probability covariates.
formulaDelta	Regression formula for the initial distribution.
stationary	FALSE if there are time-varying covariates in <code>formula</code> or any covariates in <code>formulaDelta</code> . If TRUE, the initial distribution is considered equal to the stationary distribution. Default: FALSE.
mixtures	Number of mixtures for the state transition probabilities.
formulaPi	Regression formula for the mixture distribution probabilities. Note that only the covariate values from the first row for each individual ID in <code>data</code> are used (i.e. time-varying covariates cannot be used for the mixture probabilities).
DM	An optional named list indicating the design matrices to be used for the probability distribution parameters of each data stream.
userBounds	An optional named list of 2-column matrices specifying bounds on the natural (i.e. real) scale of the probability distribution parameters for each data stream.
workBounds	An optional named list of 2-column matrices specifying bounds on the working scale of the probability distribution, transition probability, and initial distribution parameters.
betaCons	Matrix of the same dimension as <code>beta0</code> composed of integers identifying any equality constraints among the t.p.m. parameters.

betaRef	Numeric vector of length nbStates indicating the reference elements for the t.p.m. multinomial logit link.
deltaCons	Matrix of the same dimension as delta0 composed of integers identifying any equality constraints among the initial distribution working scale parameters. Ignored unless a formula is provided in formulaDelta.
stateNames	Optional character vector of length nbStates indicating state names.
fixPar	An optional list of vectors indicating parameters which are assumed known prior to fitting the model.
prior	A function that returns the log-density of the working scale parameter prior distribution(s).
hierStates	A hierarchical model structure <a href="#">Node</a> for the states ('state'). See <a href="#">fitHMM</a> .
hierDist	A hierarchical data structure <a href="#">Node</a> for the data streams ('dist'). See <a href="#">fitHMM</a> .
hierBeta	A hierarchical data structure <a href="#">Node</a> for the initial matrix of regression coefficients for the transition probabilities at each level of the hierarchy ('beta'). See <a href="#">fitHMM</a> .
hierDelta	A hierarchical data structure <a href="#">Node</a> for the initial values for the initial distribution at each level of the hierarchy ('delta'). See <a href="#">fitHMM</a> .
hierFormula	A hierarchical formula structure for the transition probability covariates for each level of the hierarchy ('formula'). See <a href="#">fitHMM</a> .
hierFormulaDelta	A hierarchical formula structure for the initial distribution covariates for each level of the hierarchy ('formulaDelta'). See <a href="#">fitHMM</a> . Default: NULL (no covariate effects and fixPar\$delta is specified on the working scale).

### See Also

[fitHMM](#), [MIfitHMM](#)

### Examples

```
m <- example$m
checkPar0(data=m$data, nbStates=2, dist=m$conditions$dist,
           estAngleMean = m$conditions$estAngleMean,
           formula = m$conditions$formula)

par <- getPar(m)
checkPar0(data=m$data, nbStates=2, dist=m$conditions$dist,
           estAngleMean = m$conditions$estAngleMean,
           formula = m$conditions$formula,
           Par0=par$Par, beta0=par$beta, delta0=par$delta)

dummyDat <- data.frame(step=0, angle=0, cov1=0, cov2=0)
checkPar0(data=dummyDat, nbStates=2, dist=m$conditions$dist,
           estAngleMean = m$conditions$estAngleMean,
           formula = m$conditions$formula)

## Not run:
simDat <- simData(nbStates=2, dist=m$conditions$dist, Par = par$Par,
                 spatialCovs = list(forest=forest),
```

```

        centers = matrix(0,1,2),
        nbCovs = 2)
checkPar0(data = simDat, nbStates=2, dist=m$conditions$dist,
          formula = ~forest,
          DM = list(step=list(mean=~cov1, sd=~cov2),
                    angle=list(mean=~center1.angle,concentration=~1)),
          estAngleMean=list(angle=TRUE),
          circularAngleMean=list(angle=TRUE))

par <- list(step=rnorm(8),angle=rnorm(4))
beta0 <- matrix(rnorm(4),2,2)
delta0 <- c(0.5,0.5)
checkPar0(data = simDat, nbStates=2, dist=m$conditions$dist,
          Par0 = par, beta0 = beta0, delta0 = delta0,
          formula = ~forest,
          DM = list(step=list(mean=~cov1, sd=~cov2),
                    angle=list(mean=~center1.angle,concentration=~1)),
          estAngleMean=list(angle=TRUE),
          circularAngleMean=list(angle=TRUE))

## End(Not run)

```

---

CIbeta

*Confidence intervals for working (i.e., beta) parameters*


---

### Description

Computes the standard errors and confidence intervals on the beta (i.e., working) scale of the data stream probability distribution parameters, as well as for the transition probabilities regression parameters. Working scale depends on the real (i.e., natural) scale of the parameters. For non-circular distributions or for circular distributions with `estAngleMean=FALSE`:

### Usage

```
CIbeta(m, alpha = 0.95)
```

### Arguments

<code>m</code>	A <code>momentuHMM</code> object
<code>alpha</code>	Significance level of the confidence intervals. Default: 0.95 (i.e. 95% CIs).

### Details

1) if both lower and upper bounds are finite then logit is the working scale; 2) if lower bound is finite and upper bound is infinite then log is the working scale.

For circular distributions with `estAngleMean=TRUE` and no constraints imposed by a design matrix (DM) or bounds (`userBounds`), then the working parameters are complex functions of both the angle mean and concentrations/sd natural parameters (in this case, it's probably best just to focus on the

real parameter estimates!). However, if constraints are imposed by DM or userBounds on circular distribution parameters with `estAngleMean=TRUE` and `circularAngleMean=FALSE`:

1) if the natural bounds are  $(-\pi, \pi]$  then tangent is the working scale, otherwise if both lower and upper bounds are finite then logit is the working scale; 2) if lower bound is finite and upper bound is infinite then log is the working scale.

When circular-circular regression is specified using `circularAngleMean`, the working scale for the mean turning angle is not as easily interpretable, but the link function is  $\text{atan2}(\sin(X)*B, 1+\cos(X)*B)$ , where X are the angle covariates and B the angle coefficients. Under this formulation, the reference turning angle is 0 (i.e., movement in the same direction as the previous time step). In other words, the mean turning angle is zero when the coefficient(s)  $B=0$ .

## Value

A list of the following objects:

...	List(s) of estimates ('est'), standard errors ('se'), and confidence intervals ('lower', 'upper') for the working parameters of the data streams
beta	List of estimates ('est'), standard errors ('se'), and confidence intervals ('lower', 'upper') for the working parameters of the transition probabilities

## Examples

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

CIbeta(m)
```

---

circAngles	<i>Convert standard direction angles (in radians relative to the x-axis) to turning angle covariates suitable for circular-circular regression on the angle mean</i>
------------	--

---

## Description

This function can be used to convert angular covariates (e.g., ocean currents, wind direction) measured in radians relative to the x-axis to turning angle covariates suitable for circular-circular regression in `fitHMM` or `MIfitHMM`.

## Usage

```
circAngles(refAngle, data, coordNames = c("x", "y"))
```

**Arguments**

refAngle	Numeric vector of standard direction angles (in radians) relative to the x-axis, where 0 = east, $\pi/2$ = north, $\pi$ = west, $-\pi/2$ = south
data	data frame containing fields for the x- and y-coordinates (identified by coordNames) and 'ID' (if more than one individual)
coordNames	Names of the columns of coordinates in data. Default: <code>c("x", "y")</code> .

**Value**

A vector of turning angles between the movement direction at time step t-1 and refAngle at time t

**Examples**

```
# extract data from momentuHMM example
data<-example$m$data

# generate fake angle covariates
u <- rnorm(nrow(data)) # horizontal component
v <- rnorm(nrow(data)) # vertical component
refAngle <- atan2(v,u)

# add turning angle covariate to data
data$cov3 <- circAngles(refAngle=refAngle,data=data)
```

---

CIreal

*Confidence intervals for the natural (i.e., real) parameters*


---

**Description**

Computes the standard errors and confidence intervals on the real (i.e., natural) scale of the data stream probability distribution parameters, as well as for the transition probabilities parameters. If covariates are included in the probability distributions or TPM formula, the mean values of non-factor covariates are used for calculating the natural parameters. For any covariate(s) of class 'factor', then the value(s) from the first observation in the data are used.

**Usage**

```
CIreal(m, alpha = 0.95, covs = NULL, parms = NULL)

## Default S3 method:
CIreal(m, alpha = 0.95, covs = NULL, parms = NULL)

## S3 method for class 'hierarchical'
CIreal(m, alpha = 0.95, covs = NULL, parms = NULL)
```

**Arguments**

<code>m</code>	A <code>momentuHMM</code> , <code>momentuHierHMM</code> , <code>miHMM</code> , or <code>miSum</code> object
<code>alpha</code>	Significance level of the confidence intervals. Default: 0.95 (i.e. 95% CIs).
<code>covs</code>	Data frame consisting of a single row indicating the covariate values to be used in the calculations. By default, no covariates are specified.
<code>parms</code>	Optional character vector indicating which groups of real parameters to calculate confidence intervals for (e.g., <code>'step'</code> , <code>'angle'</code> , <code>'gamma'</code> , <code>'delta'</code> , etc.). Default: <code>NULL</code> , in which case confidence intervals are calculated for all groups of parameters in the model.

**Details**

For any covariates that are not specified using `covs`, the means of the covariate(s) are used (unless the covariate is a factor, in which case the first factor in the data is used).

**Value**

A list of the following objects:

<code>...</code>	List(s) of estimates ( <code>'est'</code> ), standard errors ( <code>'se'</code> ), and confidence intervals ( <code>'lower'</code> , <code>'upper'</code> ) for the natural parameters of the data streams
<code>gamma</code>	List of estimates ( <code>'est'</code> ), standard errors ( <code>'se'</code> ), and confidence intervals ( <code>'lower'</code> , <code>'upper'</code> ) for the transition probabilities
<code>delta</code>	List of estimates ( <code>'est'</code> ), standard errors ( <code>'se'</code> ), and confidence intervals ( <code>'lower'</code> , <code>'upper'</code> ) for the initial state probabilities
<code>hierGamma</code>	A hierarchical data structure <a href="#">Node</a> including a list of estimates ( <code>'est'</code> ), standard errors ( <code>'se'</code> ), and confidence intervals ( <code>'lower'</code> , <code>'upper'</code> ) for the transition probabilities for each level of the hierarchy (only applies if <code>m</code> is a hierarchical model object)
<code>hierDelta</code>	A hierarchical data structure <a href="#">Node</a> including a list of estimates ( <code>'est'</code> ), standard errors ( <code>'se'</code> ), and confidence intervals ( <code>'lower'</code> , <code>'upper'</code> ) for the initial state probabilities for each level of the hierarchy (only applies if <code>m</code> is a hierarchical model object)

**Examples**

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

ci1<-CIreal(m)

# specify 'covs'
ci2<-CIreal(m,covs=data.frame(cov1=mean(m$data$cov1),cov2=mean(m$data$cov2)))

all.equal(ci1,ci2)
```

---

crawlMerge	<i>Merge crwData or crwHierData object with additional data streams and/or covariates</i>
------------	---

---

### Description

This function can be used to merge `crwData` or `crwHierData` objects (as returned by `crawlWrap`) with additional data streams and/or covariates that are unrelated to location.

### Usage

```
crawlMerge(crwData, data, Time.name)
```

### Arguments

<code>crwData</code>	A <code>crwData</code> or <code>crwHierData</code> object
<code>data</code>	A data frame containing required columns <code>ID</code> , <code>Time.name</code> , and, if <code>crwData</code> is hierarchical, <code>level</code> , plus any additional data streams and/or covariates to merge with <code>crwData</code> .
<code>Time.name</code>	Character string indicating name of the time column to be used for merging

### Details

Specifically, the function merges the `crwData$crwPredict` data frame with `data` based on the `ID`, `Time.name`, and, if `crwData` is hierarchical, `level` columns. Thus `data` must contain `ID`, `Time.name`, and, if `crwData` is hierarchical, `level` columns.

Only rows of `data` with `ID`, `Time.name`, and, if `crwData` is hierarchical, `level` values that exactly match `crwData$crwPredict` are merged. Typically, the `Time.name` column in `data` should match predicted times of locations in `crwData$crwPredict` (i.e. those corresponding to `crwData$crwPredict$locType=="p"`)

### Value

A `crwData` object

### Examples

```
## Not run:
# extract simulated obsData from example data
obsData <- miExample$obsData

# error ellipse model
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)

# Fit crwMLE models to obsData and predict locations
# at default intervals for both individuals
crwOut <- crawlWrap(obsData=obsData,
  theta=c(4,0), fixPar=c(1,1,NA,NA),
  err.model=err.model, attempts=100)
```

```
# create data frame with fake data stream
data <- data.frame(ID=rep(factor(c(1,2)),times=c(753,652)),
                  time=c(1:753,1:652),
                  fake=rpois(753+652,5))

# merge fake data stream with crwOut
crwOut <- crawlMerge(crwOut,data,"time")

## End(Not run)
```

---

crawlWrap

*Fit and predict tracks for using crawl*

---

### Description

Wrapper function for fitting `crawl::crwMLE` models and predicting locations with `crawl::crwPredict` for multiple individuals.

### Usage

```
crawlWrap(
  obsData,
  timeStep = 1,
  ncores = 1,
  retryFits = 0,
  retrySD = 1,
  retryParallel = FALSE,
  mov.model = ~1,
  err.model = NULL,
  activity = NULL,
  drift = NULL,
  coord = c("x", "y"),
  proj = NULL,
  Time.name = "time",
  time.scale = "hours",
  theta,
  fixPar,
  method = "L-BFGS-B",
  control = NULL,
  constr = NULL,
  prior = NULL,
  need.hess = TRUE,
  initialSANN = list(maxit = 200),
  attempts = 1,
  predTime = NULL,
  fillCols = FALSE,
```



```

    coordLevel = NULL,
    ...
)

```

## Arguments

obsData	data.frame object containing fields for animal ID ('ID'), time of observation (identified by Time.name, must be numeric or POSIXct), and observed locations (x- and y- coordinates identified by coord), such as that returned by <a href="#">simData</a> when temporally-irregular observed locations or measurement error are included. Alternatively, a <a href="#">SpatialPointsDataFrame</a> or sf object will also be accepted, in which case the coord values will be taken from the spatial data set and ignored in the arguments. Note that <a href="#">crwMLE</a> requires that longitude/latitude coordinates be projected to UTM (i.e., easting/northing). For further details see <a href="#">crwMLE</a> .
timeStep	Length of the time step at which to predict regular locations from the fitted model. Unless predTime is specified, the sequence of times is seq(a_i, b_i, timeStep) where a_i and b_i are the times of the first and last observations for individual i. timeStep can be numeric (regardless of whether obsData[[Time.name]] is numeric or POSIXct) or a character string (if obsData[[Time.name]] is of class POSIXct) containing one of "sec", "min", "hour", "day", "DSTday", "week", "month", "quarter" or "year". This can optionally be preceded by a positive integer and a space, or followed by "s" (e.g., "2 hours"; see <a href="#">seq.POSIXt</a> ). timeStep is not used for individuals for which predTime is specified.
ncores	Number of cores to use for parallel processing. Default: 1 (no parallel processing).
retryFits	Number of times to attempt to achieve convergence and valid (i.e., not NaN) variance estimates after the initial model fit.
retrySD	An optional list of scalars or vectors for each individual indicating the standard deviation to use for normal perturbations of theta when retryFits>0 (or attempts>1). Instead of a list object, retrySD can also be a scalar or a vector, in which case the same values are used for each individual. If a scalar is provided, then the same value is used for each parameter. If a vector is provided, it must be of length length(theta) for the corresponding individual(s). Default: 1, i.e., a standard deviation of 1 is used for all parameters of all individuals. Ignored unless retryFits>0 (or attempts>1).
retryParallel	Logical indicating whether or not to perform retryFits attempts for each individual in parallel. Default: FALSE. Ignored unless retryFits>0 and ncores>1. Note that when attempts are done in parallel (i.e. retryParallel=TRUE), the current value for the log-likelihood of each individual and warnings about convergence are not printed to the console.
mov.model	List of mov.model objects (see <a href="#">crwMLE</a> ) containing an element for each individual. If only one movement model is provided, then the same movement model is used for each individual.
err.model	List of err.model objects (see <a href="#">crwMLE</a> ) containing an element for each individual. If only one error model is provided, then the same error model is used for

	each individual (in which case the names of the <code>err.model</code> components corresponding to easting/longitudinal and northing/latitudinal location error must match <code>coord</code> ).
<code>activity</code>	List of activity objects (see <a href="#">crwMLE</a> ) containing an element for each individual. If only one activity covariate is provided, then the same activity covariate is used for each individual.
<code>drift</code>	List of drift objects (see <a href="#">crwMLE</a> ) containing an element for each individual. If only one drift component is provided, then the same drift component is used for each individual.
<code>coord</code>	A 2-vector of character values giving the names of the "x" and "y" coordinates in data. See <a href="#">crwMLE</a> .
<code>proj</code>	A list of valid epsg integer codes or proj4string for <code>obsData</code> that does not inherit either 'sf' or 'sp'. A valid 'crs' list is also accepted. Otherwise, ignored. If only one <code>proj</code> is provided, then the same projection is used for each individual.
<code>Time.name</code>	Character indicating name of the location time column. See <a href="#">crwMLE</a> .
<code>time.scale</code>	character. Scale for conversion of POSIX time to numeric for modeling. Defaults to "hours".
<code>theta</code>	List of theta objects (see <a href="#">crwMLE</a> ) containing an element for each individual. If only one theta is provided, then the same starting values are used for each individual. If theta is not specified, then <a href="#">crwMLE</a> default values are used (i.e. each parameter is started at zero).
<code>fixPar</code>	List of <code>fixPar</code> objects (see <a href="#">crwMLE</a> ) containing an element for each individual. If only one <code>fixPar</code> is provided, then the same parameters are held fixed to the given value for each individual. If <code>fixPar</code> is not specified, then no parameters are fixed.
<code>method</code>	Optimization method that is passed to <a href="#">optim</a> .
<code>control</code>	Control list which is passed to <a href="#">optim</a> .
<code>constr</code>	List of <code>constr</code> objects (see <a href="#">crwMLE</a> ) containing an element for each individual. If only one <code>constr</code> is provided, then the same box constraints for the parameters are used for each individual.
<code>prior</code>	List of prior objects (see <a href="#">crwMLE</a> ) containing an element for each individual. If only one prior is provided, then the same prior is used for each individual.
<code>need.hess</code>	A logical value which decides whether or not to evaluate the Hessian for parameter standard errors
<code>initialSANN</code>	Control list for <a href="#">optim</a> when simulated annealing is used for obtaining start values. See details
<code>attempts</code>	The number of times likelihood optimization will be attempted in cases where the fit does not converge or is otherwise non-valid. Note this is not the same as <code>retryFits</code> because <code>attempts</code> only applies when the current fit clearly does not appear to have converged; <code>retryFits</code> will proceed with additional model fitting attempts regardless of the model output.
<code>predTime</code>	List of <code>predTime</code> objects (see <a href="#">crwPredict</a> ) containing an element for each individual. <code>predTime</code> can be specified as an alternative to the automatic sequences generated according to <code>timeStep</code> . If only one <code>predTime</code> object is provided, then the same prediction times are used for each individual.

fillCols	Logical indicating whether or not to use the <code>crawl::fillCols</code> function for filling in missing values in <code>obsData</code> for which there is a single unique value. Default: FALSE. If the output from <code>crawlWrap</code> is intended for analyses using <code>fithMM</code> or <code>MifithMM</code> , setting <code>fillCols=TRUE</code> should typically be avoided.
coordLevel	Character string indicating the level of the hierarchy for the location data. Ignored unless <code>obsData</code> includes a 'level' field.
...	Additional arguments that are ignored.

### Details

- Consult `crwMLE` and `crwPredict` for further details about model fitting and prediction.
- Note that the names of the list elements corresponding to each individual in `mov.model`, `err.model`, `activity`, `drift`, `theta`, `fixPar`, `constr`, `prior`, and `predTime` must match the individual IDs in `obsData`. If only one element is provided for any of these arguments, then the same element will be applied to all individuals.

### Value

A `crwData` or `crwHierData` object, i.e. a list of:

<code>crwFits</code>	A list of <code>crwFit</code> objects returned by <code>crawl::crwMLE</code> . See <code>crwMLE</code>
<code>crwPredict</code>	A <code>crwPredict</code> data frame with <code>obsData</code> merged with the predicted locations. See <code>crwPredict</code> .

The `crwData` object is used in `MifithMM` analyses that account for temporal irregularity or location measurement error.

### See Also

`MifithMM`, `simData`

### Examples

```
## Not run:
# extract simulated obsData from example data
obsData <- miExample$obsData

# error ellipse model
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)

# Fit crwMLE models to obsData and predict locations
# at default intervals for both individuals
crwOut1 <- crawlWrap(obsData=obsData,
  theta=c(4,0), fixPar=c(1,1,NA,NA),
  err.model=err.model, attempts=100)

# Fit the same crwMLE models and predict locations
# at same intervals but specify for each individual using lists
crwOut2 <- crawlWrap(obsData=obsData,
  theta=list(c(4,0),c(4,0)), fixPar=list(c(1,1,NA,NA),c(1,1,NA,NA)),
```

```
err.model=list(err.model,err.model),
predTime=list('1'=seq(1,633),'2'=seq(1,686)))

## End(Not run)
```

---

crwData                      *Constructor of crwData objects*

---

**Description**

Constructor of crwData objects

**Usage**

```
crwData(m)
```

**Arguments**

m                      A list of attributes of crawl output: crwFits (a list of crwFit objects) and crwPredict (a crwPredict object)

**Value**

An object crwData.

**See Also**

[crawlWrap](#), [MifitHMM](#)

---

crwHierData                      *Constructor of crwHierData objects*

---

**Description**

Constructor of crwHierData objects

**Usage**

```
crwHierData(m)
```

**Arguments**

m                      A list of attributes of crawl output: crwFits (a list of crwFit objects) and crwPredict (a crwPredict object)

**Value**

An object crwHierData.

**See Also**

[crawlWrap](#), [MIfitHMM](#)

---

crwHierSim	<i>Constructor of crwHierSim objects</i>
------------	--

---

**Description**

Constructor of crwHierSim objects

**Usage**

```
crwHierSim(m)
```

**Arguments**

**m** A list of attributes required for multiple imputation data generated from a [crwHierData](#) object using [MIfitHMM](#): miData (a list of [momentuHMMData](#) objects), and crwSimulator (a list of [crwSimulator](#) objects).  
crwHierSim objects are returned by [MIfitHMM](#) when argument miData is a [crwHierData](#) object and argument fit=FALSE.

**Value**

An object crwHierSim.

---

crwSim	<i>Constructor of crwSim objects</i>
--------	--------------------------------------

---

**Description**

Constructor of crwSim objects

**Usage**

```
crwSim(m)
```

**Arguments**

`m` A list of attributes required for multiple imputation data generated from a `crwData` object using `MifitHMM`: `miData` (a list of `momentuHMMData` objects), and `crwSimulator` (a list of `crwSimulator` objects).  
`crwSim` objects are returned by `MifitHMM` when argument `miData` is a `crwData` object and argument `fit=FALSE`.

**Value**

An object `crwSim`.

---

`dbern_rcpp`

*Bernoulli density function*

---

**Description**

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

**Usage**

`dbern_rcpp(x, prob, foo)`

**Arguments**

`x` Vector of quantiles  
`prob` success probability  
`foo` Unused (for compatibility with template)

**Value**

Vector of densities

---

`dbeta_rcpp`

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

---

**Description**

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

**Usage**

`dbeta_rcpp(x, shape1, shape2)`

**Arguments**

x	Vector of quantiles
shape1	Shape1
shape2	Shape2

**Value**

Vector of densities

---

dcat_rcpp	<i>Categorical density function</i>
-----------	-------------------------------------

---

**Description**

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

**Usage**

```
dcat_rcpp(x, prob, foo)
```

**Arguments**

x	Vector of quantiles
prob	success probability
foo	Unused (for compatibility with template)

**Value**

Vector of densities

---

dexp_rcpp	<i>Exponential density function</i>
-----------	-------------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

Vector of densities

---

dgamma_rcpp	<i>Gamma density function</i>
-------------	-------------------------------

---

**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

---

distAngle	<i>Calculate distance between points y and z and turning angle between points x, y, and z</i>
-----------	---

---

**Description**

Calculate distance between points y and z and turning angle between points x, y, and z

**Usage**

```
distAngle(x, y, z, type = "UTM", angleCov = TRUE)
```



**Arguments**

x	location 1
y	location 2
z	location 3
type	'UTM' if easting/northing provided (the default), 'LL ' if longitude/latitude
angleCov	logical indicating to not return NA when x=y or y=z. Default: TRUE (i.e. NA is not returned if x=y or y=z).

**Details**

Used in [prepData](#) and [simData](#) to get distance and turning angle covariates between locations (x1,x2), (y1,y2) and activity center (z1,z2).

If type='LL ' then distance is calculated as great circle distance using [spDistsN1](#), and turning angle is calculated based on initial bearings using [bearing](#).

**Value**

2-vector with first element the distance between y and z and second element the turning angle between (x,y) and (y,z).

---

dlnorm_rcpp	<i>Log-normal density function</i>
-------------	------------------------------------

---

**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

---

dlogis_rcpp	<i>logistic density function</i>
-------------	----------------------------------

---

**Description**

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

**Usage**

```
dlogis_rcpp(x, location, scale)
```

**Arguments**

x	Vector of quantiles
location	mean of the distribution
scale	Dispersion parameter

**Value**

Vector of densities

---

dmvnorm_rcpp	<i>C++ implementation of multivariate Normal probability density function for multiple inputs</i>
--------------	---

---

**Description**

C++ implementation of multivariate Normal probability density function for multiple inputs

**Usage**

```
dmvnorm_rcpp(x, mean, varcovM)
```

**Arguments**

x	data matrix of dimension $p \times n$ , $p$ being the dimension of the data and $n$ the number of data points.
mean	mean vectors matrix of dimension $p \times n$
varcovM	list of length $n$ of variance-covariance matrices, each of dimensions $p \times p$ .

**Value**

matrix of densities of dimension  $K \times n$ .

---

dnbinom_rcpp	<i>negative binomial density function</i>
--------------	---

---

**Description**

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

**Usage**

```
dnbinom_rcpp(x, mu, size)
```

**Arguments**

x	Vector of quantiles
mu	Mean of the distribution
size	Dispersion parameter

**Value**

Vector of densities

---

dnorm_rcpp	<i>Normal density function</i>
------------	--------------------------------

---

**Description**

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

**Usage**

```
dnorm_rcpp(x, mean, sd)
```

**Arguments**

x	Vector of quantiles
mean	Mean of the distribution
sd	Standard deviation of the distribution

**Value**

Vector of densities

---

`dpois_rcpp`*Poisson density function*

---

**Description**

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

**Usage**

```
dpois_rcpp(x, rate, foo)
```

**Arguments**

<code>x</code>	Vector of quantiles
<code>rate</code>	Rate
<code>foo</code>	Unused (for compatibility with template)

**Value**

Vector of densities

---

`dt_rcpp`*student t density function*

---

**Description**

Probability density function of non-central student t (written in C++)

**Usage**

```
dt_rcpp(x, df, ncp)
```

**Arguments**

<code>x</code>	Vector of quantiles
<code>df</code>	degrees of freedom
<code>ncp</code>	non-centrality parameter

**Value**

Vector of densities

---

`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**

<code>x</code>	Vector of quantiles
<code>mu</code>	Mean
<code>kappa</code>	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**

<code>x</code>	Vector of quantiles
<code>shape</code>	Shape
<code>scale</code>	Scale

**Value**

Vector of densities

---

dwrpcauchy_rcpp	<i>Wrapped Cauchy density function</i>
-----------------	--

---

**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

---

exampleData	<i>Example dataset</i>
-------------	------------------------

---

**Description**

These data are used in the examples and tests of functions to keep them as short as possible.

**Usage**

```
example
```

```
miExample
```

```
forest
```

**Details**

example is a list of the following objects for demonstrating `fitHMM`:

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

miExample is a list of the following objects for demonstrating `crawlWrap`, `MIfitHMM`, and `MIpool`:

- obsData Simulated observation data with measurement error and temporal irregularity (generated by `simData`)
- bPar initial parameter estimates for MIfitHMM examples

forest is a simulated spatial covariate `raster` object of the `RasterLayer` class

---

expandPar	<i>Expand vector of free working parameters to vector of all working parameters including any fixed parameters (used in fitHMM.R and nLogLike.R)</i>
-----------	--

---

### Description

Expand vector of free working parameters to vector of all working parameters including any fixed parameters (used in fitHMM.R and nLogLike.R)

### Usage

```
expandPar(
  optPar,
  optInd,
  fixPar,
  wparIndex,
  betaCons,
  deltaCons,
  nbStates,
  nbCovsDelta,
  stationary,
  nbCovs,
  nbRecovs = 0,
  mixtures = 1,
  nbCovsPi = 0
)
```

### Arguments

optPar	vector of free working parameters
optInd	indices of constrained parameters
fixPar	Vector of working parameters which are assumed known prior to fitting the model (NA indicates parameters is to be estimated)
wparIndex	Vector of indices for the elements of fixPar that are not NA
betaCons	Matrix of the same dimension as beta0 composed of integers identifying any equality constraints among the t.p.m. parameters.
deltaCons	Matrix of the same dimension as delta0 composed of integers identifying any equality constraints among the initial distribution working scale parameters.

nbStates	Number of states of the HMM
nbCovsDelta	Number of initial distribution covariates
stationary	FALSE if there are time-varying covariates in formula or any covariates in formulaDelta. If TRUE, the initial distribution is considered equal to the stationary distribution. Default: FALSE.
nbCovs	Number of t.p.m. covariates
nbRecovs	Number of recharge covariates
mixtures	Number of mixtures for the state transition probabilities
nbCovsPi	Number of mixture probability covariates

### Value

A vector of all working parameters including any fixed parameters

### Examples

```
## Not run:
nbStates <- 2
stepDist <- "gamma" # step distribution
angleDist <- "vm" # turning angle distribution

# extract data from momentuHMM example
data <- example$m$data

### 1. 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)
stepPar <- c(mu0,sigma0) # no zero-inflation, so no zero-mass included
anglePar <- kappa0 # not estimating angle mean, so not included
formula <- ~cov1+cos(cov2)

# constrain cov1 effect to state 1 -> 2 and cov2 effect to state 2 -> 1
fixPar <- list(beta=c(NA,NA,0,NA,0,NA))

m <- fithMM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
            Par0=list(step=stepPar,angle=anglePar),formula=formula,fixPar=fixPar)

# convert free parameter vector (m$mod$wpar) to full set of working parameters (m$mod$estimate)
est <- momentuHMM:::expandPar(m$mod$wpar,m$conditions$optInd,unlist(m$conditions$fixPar),
                             m$conditions$wparIndex,m$conditions$betaCons,m$conditions$deltaCons,
                             nbStates,
                             ncol(m$covsDelta)-1,m$conditions$stationary,nrow(m$mle$beta)-1)

all(est==m$mod$estimate)

## End(Not run)
```



---

`fitHMM`*Fit a multivariate HMM to the data*

---

**Description**

Fit a (multivariate) hidden Markov model to the data provided, using numerical optimization of the log-likelihood function.

**Usage**

```
fitHMM(data, ...)  
  
## S3 method for class 'momentuHMMData'  
fitHMM(  
  data,  
  nbStates,  
  dist,  
  Par0,  
  beta0 = NULL,  
  delta0 = NULL,  
  estAngleMean = NULL,  
  circularAngleMean = NULL,  
  formula = ~1,  
  formulaDelta = NULL,  
  stationary = FALSE,  
  mixtures = 1,  
  formulaPi = NULL,  
  nlmPar = list(),  
  fit = TRUE,  
  DM = NULL,  
  userBounds = NULL,  
  workBounds = NULL,  
  betaCons = NULL,  
  betaRef = NULL,  
  deltaCons = NULL,  
  mvnCoords = NULL,  
  stateNames = NULL,  
  knownStates = NULL,  
  fixPar = NULL,  
  retryFits = 0,  
  retrySD = NULL,  
  optMethod = "nlm",  
  control = list(),  
  prior = NULL,  
  modelName = NULL,  
  ...  
)
```

```

## S3 method for class 'momentuHierHMMData'
fitHMM(
  data,
  hierStates,
  hierDist,
  Par0,
  hierBeta = NULL,
  hierDelta = NULL,
  estAngleMean = NULL,
  circularAngleMean = NULL,
  hierFormula = NULL,
  hierFormulaDelta = NULL,
  mixtures = 1,
  formulaPi = NULL,
  nlmPar = list(),
  fit = TRUE,
  DM = NULL,
  userBounds = NULL,
  workBounds = NULL,
  betaCons = NULL,
  deltaCons = NULL,
  mvnCoords = NULL,
  knownStates = NULL,
  fixPar = NULL,
  retryFits = 0,
  retrySD = NULL,
  optMethod = "nlm",
  control = list(),
  prior = NULL,
  modelName = NULL,
  ...
)

```

### Arguments

<code>data</code>	A <code>momentuHMMData</code> (as returned by <code>prepData</code> or <code>simData</code> ) or a <code>momentuHierHMMData</code> (as returned by <code>prepData</code> or <code>simHierData</code> ) object.
<code>...</code>	further arguments passed to or from other methods
<code>nbStates</code>	Number of states of the HMM.
<code>dist</code>	A named list indicating the probability distributions of the data streams. Currently supported distributions are 'bern', 'beta', 'cat', 'exp', 'gamma', 'lnorm', 'logis', 'negbinom', 'norm', 'mvnorm2' (bivariate normal distribution), 'mvnorm3' (trivariate normal distribution), 'pois', 'rw_norm' (normal random walk), 'rw_mvnorm2' (bivariate normal random walk), 'rw_mvnorm3' (trivariate normal random walk), 'vm', 'vmConsensus', 'weibull', and 'wrpcauchy'. For example, <code>dist=list(step='gamma', angle='vm')</code> indicates 3 data streams ('step', 'angle', and 'dives') and their respective probability distributions ('gamma', 'vm', and 'pois'). The names of the data streams

	(e.g., 'step', 'angle', 'dives') must match component names in data.
Par0	A named list containing vectors of initial state-dependent probability distribution parameters for each data stream specified in <code>dist</code> . The parameters should be in the order expected by the pdfs of <code>dist</code> , and any zero-mass and/or one-mass parameters should be the last (if both are present, then zero-mass parameters must precede one-mass parameters). Note that zero-mass parameters are mandatory if there are zeros in data streams with a 'gamma', 'weibull', 'exp', 'lnorm', or 'beta' distribution, and one-mass parameters are mandatory if there are ones in data streams with a 'beta' distribution. For example, for a 2-state model using the Von Mises (vm) distribution for a data stream named 'angle' and the zero-inflated gamma distribution for a data stream named 'step', the vector of initial parameters would be something like: <code>Par0=list(step=c(mean_1, mean_2, sd_1, sd_2, zeromass_1, z</code>
	If DM is not specified for a given data stream, then <code>Par0</code> is on the natural (i.e., real) scale of the parameters. However, if DM is specified for a given data stream, then <code>Par0</code> must be on the working (i.e., beta) scale of the parameters, and the length of <code>Par0</code> must match the number of columns in the design matrix. See details below.
beta0	Initial matrix of regression coefficients for the transition probabilities (more information in 'Details'). Default: NULL. If not specified, <code>beta0</code> 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: <code>rep(1/nbStates, nbStates)</code> . If <code>formulaDelta</code> includes a formula, then <code>delta0</code> must be specified as a $k \times (nbStates-1)$ matrix, where $k$ is the number of covariates and the columns correspond to states 2:nbStates. See details below.
estAngleMean	An optional named list indicating whether or not to estimate the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy'). For example, <code>estAngleMean=list(angle=TRUE)</code> indicates the angle mean is to be estimated for 'angle'. Default is NULL, which assumes any angle means are fixed to zero and are not to be estimated. Any <code>estAngleMean</code> elements corresponding to data streams that do not have angular distributions are ignored. <code>estAngleMean</code> is also ignored for any 'vmConsensus' data streams (because the angle mean must be estimated in consensus models).
circularAngleMean	An optional named list indicating whether to use circular-linear (FALSE) or circular-circular (TRUE) regression on the mean of circular distributions ('vm' and 'wrpcauchy') for turning angles. For example, <code>circularAngleMean=list(angle=TRUE)</code> indicates the angle mean is to be estimated for 'angle' using circular-circular regression. Whenever circular-circular regression is used for an angular data stream, a corresponding design matrix (DM) must be specified for the data stream, and the previous movement direction (i.e., a turning angle of zero) is automatically used as the reference angle (i.e., the intercept). Any circular-circular regression covariates in data should therefore be relative to the previous direction of movement (instead of standard directions relative to the x-axis; see <a href="#">prepData</a> and <a href="#">circAngles</a> ). See Duchesne et al. (2015) for specifics on the circular-circular regression model using previous movement direction as the reference angle. Default is NULL, which assumes circular-linear regression is used for any angular distributions for which the mean angle is to be estimated.

circularAngleMean elements corresponding to angular data streams are ignored unless the corresponding element of estAngleMean is TRUE. Any circularAngleMean elements corresponding to data streams that do not have angular distributions are ignored. circularAngleMean is also ignored for any 'vmConsensus' data streams (because the consensus model is a circular-circular regression model). Alternatively, circularAngleMean can be specified as a numeric scalar, where the value specifies the coefficient for the reference angle (i.e., directional persistence) term in the circular-circular regression model. For example, setting circularAngleMean to 0 specifies a circular-circular regression model with no directional persistence term (thus specifying a biased random walk instead of a biased correlated random walk). Setting circularAngleMean to 1 is equivalent to setting it to TRUE, i.e., a circular-circular regression model with a coefficient of 1 for the directional persistence reference angle.

formula	Regression formula for the transition probability covariates. Default: ~1 (no covariate effect). In addition to allowing standard functions in R formulas (e.g., <code>cos(cov)</code> , <code>cov1*cov2</code> , <code>I(cov^2)</code> ), special functions include <code>cosinor(cov, period)</code> for modeling cyclical patterns, spline functions ( <code>bs</code> , <code>ns</code> , <code>bSpline</code> , <code>cSpline</code> , <code>iSpline</code> , and <code>mSpline</code> ), and state- or parameter-specific formulas (see details). Any formula terms that are not state- or parameter-specific are included on all of the transition probabilities.
formulaDelta	Regression formula for the initial distribution. Default for <code>fitHMM.momentuHMMData</code> : NULL (no covariate effects; both <code>delta0</code> and <code>fixPar\$delta</code> are specified on the real scale). Default for <code>fitHMM.momentuHierHMMData</code> : ~1 (both <code>delta0</code> and <code>fixPar\$delta</code> are specified on the working scale). Standard functions in R formulas are allowed (e.g., <code>cos(cov)</code> , <code>cov1*cov2</code> , <code>I(cov^2)</code> ). When any formula is provided, then both <code>delta0</code> and <code>fixPar\$delta</code> are specified on the working scale.
stationary	FALSE if there are time-varying covariates in <code>formula</code> or any covariates in <code>formulaDelta</code> . If TRUE, the initial distribution is considered equal to the stationary distribution. Default: FALSE.
mixtures	Number of mixtures for the state transition probabilities (i.e. discrete random effects *sensu* DeRuiter et al. 2017). Default: <code>mixtures=1</code> .
formulaPi	Regression formula for the mixture distribution probabilities. Default: NULL (no covariate effects; both <code>beta0\$pi</code> and <code>fixPar\$pi</code> are specified on the real scale). Standard functions in R formulas are allowed (e.g., <code>cos(cov)</code> , <code>cov1*cov2</code> , <code>I(cov^2)</code> ). When any formula is provided, then both <code>beta0\$pi</code> and <code>fixPar\$pi</code> are specified on the working scale. Note that only the covariate values from the first row for each individual ID in data are used (i.e. time-varying covariates cannot be used for the mixture probabilities).
nlmPar	List of parameters to pass to the optimization function <code>nlm</code> (which should be either <code>print.level</code> , <code>gradtol</code> , <code>stepmax</code> , <code>steptol</code> , <code>iterlim</code> , or <code>hessian</code> – see <code>nlm</code> 's documentation for more detail). For <code>print.level</code> , 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. Ignored unless <code>optMethod="nlm"</code> .
fit	TRUE if an HMM should be fitted to the data, FALSE otherwise. If <code>fit=FALSE</code> , 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, parameter bounds, etc. Default: TRUE.

DM	<p>An optional named list indicating the design matrices to be used for the probability distribution parameters of each data stream. Each element of DM can either be a named list of linear regression formulas or a “pseudo” design matrix. For example, for a 2-state model using the gamma distribution for a data stream named 'step', <code>DM=list(step=list(mean=~cov1, sd=~1))</code> specifies the mean parameters as a function of the covariate 'cov1' for each state. This model could equivalently be specified as a 4x6 “pseudo” design matrix using character strings for the covariate: <code>DM=list(step=matrix(c(1,0,0,0, 'cov1', 0,0,0,0,1,0,0,0, 'cov1', 0,0,0,0,1,0,0,0, 'cov1', 0,0,0,0,1,0,0,0, 'cov1', 0,0,0,0,1,0,0,0), nrow=4, ncol=6))</code> where the 4 rows correspond to the state-dependent parameters (mean_1, mean_2, sd_1, sd_2) and the 6 columns correspond to the regression coefficients.</p> <p>Design matrices specified using formulas allow standard functions in R formulas (e.g., <code>cos(cov)</code>, <code>cov1*cov2</code>, <code>I(cov^2)</code>). Special formula functions include <code>cosinor(cov, period)</code> for modeling cyclical patterns, spline functions (<code>bs</code>, <code>ns</code>, <code>bSpline</code>, <code>cSpline</code>, <code>iSpline</code>, and <code>mSpline</code>), <code>angleFormula(cov, strength, by)</code> for the angle mean of circular-circular regression models, and state-specific formulas (see details). Any formula terms that are not state-specific are included on the parameters for all nbStates states.</p>
userBounds	<p>An optional named list of 2-column matrices specifying bounds on the natural (i.e., real) scale of the probability distribution parameters for each data stream. For each matrix, the first column pertains to the lower bound and the second column the upper bound. For example, for a 2-state model using the wrapped Cauchy ('wrpcauchy') distribution for a data stream named 'angle' with <code>estAngleMean\$angle=TRUE</code>, <code>userBounds=list(angle=matrix(c(-pi, -pi, -1, -1, pi, pi, 1, 1), nrow=2, ncol=2))</code> specifies (-1,1) bounds for the concentration parameters instead of the default [0,1) bounds.</p>
workBounds	<p>An optional named list of 2-column matrices specifying bounds on the working scale of the probability distribution, transition probability, and initial distribution parameters. For each matrix, the first column pertains to the lower bound and the second column the upper bound. For data streams, each element of workBounds should be a k x 2 matrix with the same name of the corresponding element of Par0, where k is the number of parameters. For transition probability parameters, the corresponding element of workBounds must be a k x 2 matrix named “beta”, where <code>k=length(beta0)</code>. For initial distribution parameters, the corresponding element of workBounds must be a k x 2 matrix named “delta”, where <code>k=length(delta0)</code>. workBounds is ignored for any given data stream unless DM is also specified.</p>
betaCons	<p>Matrix of the same dimension as beta0 composed of integers identifying any equality constraints among the t.p.m. parameters. See details.</p>
betaRef	<p>Numeric vector of length nbStates indicating the reference elements for the t.p.m. multinomial logit link. Default: NULL, in which case the diagonal elements of the t.p.m. are the reference. See details.</p>
deltaCons	<p>Matrix of the same dimension as delta0 composed of integers identifying any equality constraints among the initial distribution working scale parameters. Ignored unless a formula is provided in formulaDelta. See details.</p>

mvnCoords	Character string indicating the name of location data that are to be modeled using a multivariate normal distribution. For example, if <code>mu="mvnorm2"</code> was included in <code>dist</code> and <code>(mu.x, mu.y)</code> are location data, then <code>mvnCoords="mu"</code> needs to be specified in order for these data to be properly treated as locations in functions such as <code>plot.momentuHMM</code> , <code>plot.miSum</code> , <code>plot.miHMM</code> , <code>plotSpatialCov</code> , and <code>Mipool</code> .
stateNames	Optional character vector of length <code>nbStates</code> indicating state names.
knownStates	Vector of values of the state process which are known prior to fitting the model (if any). Default: <code>NULL</code> (states are not known). This should be a vector with length the number of rows of <code>'data'</code> ; each element should either be an integer (the value of the known states) or <code>NA</code> if the state is not known.
fixPar	An optional list of vectors indicating parameters which are assumed known prior to fitting the model. Default: <code>NULL</code> (no parameters are fixed). For data streams, each element of <code>fixPar</code> should be a vector of the same name and length as the corresponding element of <code>Par0</code> . For transition probability parameters, the corresponding element of <code>fixPar</code> must be named "beta" and have the same dimensions as <code>beta0</code> . For initial distribution parameters, the corresponding element of <code>fixPar</code> must be named "delta" and have the same dimensions as <code>delta0</code> . Each parameter should either be numeric (the fixed value of the parameter) or <code>NA</code> if the parameter is to be estimated. Corresponding <code>fixPar</code> parameters must be on the same scale as <code>Par0</code> (e.g. if <code>DM</code> is specified for a given data stream, any fixed parameters for this data stream must be on the working scale), <code>beta0</code> , and <code>delta0</code> .
retryFits	Non-negative integer indicating the number of times to attempt to iteratively fit the model using random perturbations of the current parameter estimates as the initial values for likelihood optimization. <code>Normal(0, retrySD^2)</code> perturbations are used on the working scale parameters. Default: 0. When <code>retryFits &gt; 0</code> , the model with the largest log likelihood value is returned. Ignored if <code>fit=FALSE</code> .
retrySD	An optional list of scalars or vectors indicating the standard deviation to use for normal perturbations of each working scale parameter when <code>retryFits &gt; 0</code> . For data streams, each element of <code>retrySD</code> should be a vector of the same name and length as the corresponding element of <code>Par0</code> (if a scalar is provided, then this value will be used for all working parameters of the data stream). For transition probability parameters, the corresponding element of <code>retrySD</code> must be named "beta" and have the same dimensions as <code>beta0</code> . For initial distribution parameters, the corresponding element of <code>retrySD</code> must be named "delta" and have the same dimensions as <code>delta0</code> (if <code>delta0</code> is on the working scale) or be of length <code>nbStates-1</code> (if <code>delta0</code> is on the natural scale). Alternatively <code>retrySD</code> can be a scalar, in which case this value is used for all parameters. Default: <code>NULL</code> (in which case <code>retrySD=1</code> for data stream parameters and <code>retrySD=10</code> for initial distribution and state transition probabilities). Ignored unless <code>retryFits &gt; 0</code> .
optMethod	The optimization method to be used. Can be "nlm" (the default; see <code>nlm</code> ), "Nelder-Mead" (see <code>optim</code> ), or "SANN" (see <code>optim</code> ).
control	A list of control parameters to be passed to <code>optim</code> (ignored unless <code>optMethod="Nelder-Mead"</code> or <code>optMethod="SANN"</code> ).
prior	A function that returns the log-density of the working scale parameter prior distribution(s). See 'Details'.

modelName	An optional character string providing a name for the fitted model. If provided, modelName will be returned in <code>print.momentuHMM</code> , <code>AIC.momentuHMM</code> , <code>AICweights</code> , and other functions.
hierStates	A hierarchical model structure <code>Node</code> for the states ('state'). See details.
hierDist	A hierarchical data structure <code>Node</code> for the data streams ('dist'). See details.
hierBeta	A hierarchical data structure <code>Node</code> for the matrix of initial values for the regression coefficients of the transition probabilities at each level of the hierarchy ('beta'). See details.
hierDelta	A hierarchical data structure <code>Node</code> for the matrix of initial values for the regression coefficients of the initial distribution at each level of the hierarchy ('delta'). See details.
hierFormula	A hierarchical formula structure for the transition probability covariates for each level of the hierarchy ('formula'). Default: NULL (only hierarchical-level effects, with no covariate effects). Any formula terms that are not state- or parameter-specific are included on all of the transition probabilities within a given level of the hierarchy. See details.
hierFormulaDelta	A hierarchical formula structure for the initial distribution covariates for each level of the hierarchy ('formulaDelta'). Default: NULL (no covariate effects and <code>fixPar\$delta</code> is specified on the working scale).

## Details

- By default the matrix `beta0` 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 formula 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), `beta0` has one row, for the intercept. While the diagonal elements are by default the reference elements, other elements can serve as the reference using the `betaRef` argument. For example, in a 3-state model, setting `betaRef=c(3,2,3)` changes the reference elements to state transition 1 -> 3 for state 1 (instead of 1 -> 1), state transition 2 -> 2 for state 2 (same as default), and state transition 3 -> 3 for state 3 (same as default).
- When covariates are not included in `formulaDelta` (i.e. `formulaDelta=NULL`), then `delta0` (and `fixPar$delta`) are specified as a vector of length `nbStates` that sums to 1. When any formula is specified for `formulaDelta` (e.g. `formulaDelta=~1`, `formulaDelta=~cov1`), then `delta0` (and `fixPar$delta`) must be specified as a `k x (nbStates-1)` matrix of working parameters, where `k` is the number of regression coefficients and the columns correspond to states 2:nbStates. For example, in a 3-state HMM with `formulaDelta=~cov1+cov2`, the matrix `delta0` has three rows (intercept + two covariates) and 2 columns (corresponding to states 2 and 3). The initial distribution working parameters are transformed to the real scale as `exp(covsDelta*Delta)/rowSums(exp(covsDelta*Delta))`, where `covsDelta` is the `N x k` design matrix, `Delta=cbind(rep(0,k),delta0)` is a `k x nbStates` matrix of working parameters, and `N=length(unique(data$ID))`.
- The choice of initial parameters (particularly `Par0` and `beta0`) 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.

- If DM is specified for a particular data stream, then the initial values are specified on the working (i.e., beta) scale of the parameters. The working scale of each parameter is determined by the link function used. If a parameter P is bound by (0,Inf) then the working scale is the log(P) scale. If the parameter bounds are (-pi,pi) then the working scale is tan(P/2) unless circular-circular regression is used. Otherwise if the parameter bounds are finite then logit(P) is the working scale. However, when both zero- and one-inflation are included, then a multinomial logit link is used because the sum of the zeromass and onemass probability parameters cannot exceed 1. The function `getParDM` is intended to help with obtaining initial values on the working scale when specifying a design matrix and other parameter constraints (see example below). When circular-circular regression is specified using `circularAngleMean`, the working scale for the mean turning angle is not as easily interpretable, but the link function is  $\text{atan2}(\sin(X)*B, 1+\cos(X)*B)$ , where X are the angle covariates and B the angle coefficients (see Duchesne et al. 2015). Under this formulation, the reference turning angle is 0 (i.e., movement in the same direction as the previous time step). In other words, the mean turning angle is zero when the coefficient(s) B=0.
- Circular-circular regression in `momentuHMM` is designed for turning angles (not bearings) as computed by `simData` and `prepData`. Any circular-circular regression angle covariates for time step t should therefore be relative to the previous direction of movement for time step t-1. In other words, circular-circular regression covariates for time step t should be the turning angle between the direction of movement for time step t-1 and the standard direction of the covariate relative to the x-axis for time step t. If provided standard directions in radians relative to the x-axis (where 0 = east, pi/2 = north, pi = west, and -pi/2 = south), `circAngles` or `prepData` can perform this calculation for you.

When the circular-circular regression model is used, the special function `angleFormula(cov, strength, by)` can be used in DM for the mean of angular distributions (i.e. 'vm', 'vmConsensus', and 'wrpcauchy'), where `cov` is an angle covariate (e.g. wind direction), `strength` is an optional positive real covariate (e.g. wind speed), and `by` is an optional factor variable for individual- or group-level effects (e.g. ID, sex). The `strength` argument allows angle covariates to be weighted based on their relative strength or importance at time step t as in Rivest et al. (2016). In this case, the link function for the mean angle is  $\text{atan2}((Z * \sin(X)) \%*\% B, 1+(Z * \cos(X)) \%*\% B)$ , where X are the angle covariates, Z the strength covariates, and B the angle coefficients (see Rivest et al. 2016).

- State-specific formulas can be specified in DM using special formula functions. These special functions can take the names `paste0("state", 1:nbStates)` (where the integer indicates the state-specific formula). For example, `DM=list(step=list(mean=~cov1+state1(cov2), sd=~cov2+state2(cov1)))` includes `cov1` on the mean parameter for all states, `cov2` on the mean parameter for state 1, `cov2` on the sd parameter for all states, and `cov1` on the sd parameter for state 2.
- State- and parameter-specific formulas can be specified for transition probabilities in formula using special formula functions. These special functions can take the names `paste0("state", 1:nbStates)` (where the integer indicates the current state from which transitions occur), `paste0("toState", 1:nbStates)` (where the integer indicates the state to which transitions occur), or `paste0("betaCol", nbStates*(nbStates-1))` (where the integer indicates the column of the beta matrix). For example with `nbStates=3`, `formula=~cov1+betaCol1(cov2)+state3(cov3)+toState1(cov4)` includes `cov1` on all transition probability parameters, `cov2` on the beta column corresponding to the transition from state 1->2, `cov3` on transition probabilities from state 3 (i.e., beta columns corresponding to state transitions 3->1 and 3->2), and `cov4` on transition probabilities to state 1 (i.e., beta columns corresponding to state transitions 2->1 and 3->1).



- `betaCons` can be used to impose equality constraints among the t.p.m. parameters. It must be a matrix of the same dimension as `beta0` and be composed of integers, where each beta parameter is sequentially indexed in a column-wise fashion (see `checkPar0`). Parameter indices in `betaCons` must therefore be integers between 1 and `nbStates*(nbStates-1)`.

Use of `betaCons` is perhaps best demonstrated by example. If no constraints are imposed (the default), then `betaCons=matrix(1:length(beta0),nrow(beta0),ncol(beta0))` such that each beta parameter is (column-wise) sequentially identified by a unique integer. Suppose we wish to fit a model with `nbStates=3` states and a covariate ('`cov1`') on the t.p.m. With no constraints on the t.p.m., we would have `betaCons=matrix(1:(2*(nbStates*(nbStates-1))),nrow=2,ncol=nbStates->2","1->3","2->1","2->3","3->1","3->2"))`. If we then wanted to constrain the t.p.m. such that the covariate effect is identical for transitions from state 1 to states 2 and 3 (and vice versa), we have `betaCons=matrix(c(1,2,3,2,5,6,7,8,9,6,11,12),nrow=2,ncol=nbStates->2","1->3","2->1","2->3","3->1","3->2"))`; this results in 10 estimated beta parameters (instead of 12), the "`cov1`" effects indexed by a "2" ("`1->2`" and "`1->3`") constrained to be equal, and the "`cov1`" effects indexed by a "6" ("`2->1`" and "`3->1`") constrained to be equal.

Now suppose we instead wish to constrain these sets of state transition probabilities to be equal, i.e.,  $\Pr(1 \rightarrow 2) = \Pr(1 \rightarrow 3)$  and  $\Pr(2 \rightarrow 1) = \Pr(3 \rightarrow 1)$ ; then we have `betaCons=matrix(c(1,2,1,2,5,6,7,8,5,6->2","1->3","2->1","2->3","3->1","3->2"))`

- Cyclical relationships (e.g., hourly, monthly) may be modeled in DM or formula using the `cosinor(x,period)` special formula function for covariate `x` and sine curve period of time length `period`. For example, if the data are hourly, a 24-hour cycle can be modeled using `~cosinor(cov1,24)`, where the covariate `cov1` is a repeating sequential series of integers indicating the hour of day (`0,1,...,23,0,1,...,23,0,1,...`) (note that `fitHMM` will not do this for you, the appropriate covariate must be included in data; see example below). The `cosinor(x,period)` function converts `x` to 2 covariates `cosinorCos(x)=cos(2*pi*x/period)` and `cosinorSin(x)=sin(2*pi*x/period)` for inclusion in the model (i.e., 2 additional parameters per state). The amplitude of the sine wave is thus `sqrt(B_cos^2 + B_sin^2)`, where `B_cos` and `B_sin` are the working parameters corresponding to `cosinorCos(x)` and `cosinorSin(x)`, respectively (e.g., see Cornelissen 2014).
- Similar to that used in `crawlWrap`, the `prior` argument is a user-specified function that returns the log-density of the working scale parameter prior distribution(s). In addition to including prior information about parameters, one area where priors can be particularly useful is for handling numerical issues that can arise when parameters are near a boundary. When parameters are near boundaries, they can wander into the "nether regions" of the parameter space during optimization. For example, setting `prior=function(par) {sum(dnorm(par,0,sd,log=TRUE))}` with a reasonably large `sd` (e.g. 100 or 1000) can help prevent working parameters from straying too far along the real line. Here `par` is the vector of working scale parameters (as returned by `fitHMM`, e.g., see `example$m$mod$estimate`) in the following order: data stream working parameters (in order `names(dist)`), beta working parameters, and delta working parameters. Instead of specifying the same prior on all parameters, different priors could be specified on different parameters (and not all parameters must have user-specified priors). For example, `prior=function(par){dnorm(par[3],0,100,log=TRUE)}` would only specify a prior for the third working parameter. Note that the `prior` function must return a scalar on the log scale. See 'harbourSealExample.R' in the "vignettes" source directory for an example using the `prior` argument.
- `fitHMM.momentuHierHMMDData` is very similar to `fitHMM.momentuHMMDData` except that in-

stead of simply specifying the number of states (`nbStates`), distributions (`dist`), and a single t.p.m. formula (`formula`), the `hierStates` argument specifies the hierarchical nature of the states, the `hierDist` argument specifies the hierarchical nature of the data streams, and the `hierFormula` argument specifies a t.p.m. formula for each level of the hierarchy. All are specified as `Node` objects from the `data.tree` package.

### Value

A `momentuHMM` or `momentuHierHMM` object, i.e. a list of:

<code>mle</code>	A named list of the maximum likelihood estimates of the parameters of the model (if the numerical algorithm has indeed identified the global maximum of the likelihood function). Elements are included for the parameters of each data stream, as well as <code>beta</code> (transition probabilities regression coefficients - more information in 'Details'), <code>gamma</code> (transition probabilities on real scale, based on mean covariate values if formula includes covariates), and <code>delta</code> (initial distribution).
<code>CIreal</code>	Standard errors and 95% confidence intervals on the real (i.e., natural) scale of parameters
<code>CIbeta</code>	Standard errors and 95% confidence intervals on the beta (i.e., working) scale of parameters
<code>data</code>	The <code>momentuHMMDData</code> or <code>momentuHierHMMDData</code> object
<code>mod</code>	List object returned by the numerical optimizer <code>nlm</code> or <code>optim</code> . Items in <code>mod</code> include the best set of free working parameters found ( <code>wpar</code> ), the best full set of working parameters including any fixed parameters ( <code>estimate</code> ), the value of the likelihood at <code>estimate</code> ( <code>minimum</code> ), the estimated variance-covariance matrix at <code>estimate</code> ( <code>Sigma</code> ), and the elapsed time in seconds for the optimization ( <code>elapsedTime</code> ).
<code>conditions</code>	Conditions used to fit the model, e.g., <code>bounds</code> (parameter bounds), <code>distributions</code> , <code>zeroInflation</code> , <code>estAngleMean</code> , <code>stationary</code> , <code>formula</code> , <code>DM</code> , <code>fullDM</code> (full design matrix), etc.
<code>rawCovs</code>	Raw covariate values for transition probabilities, as found in the data (if any). Used in <code>plot.momentuHMM</code> .
<code>stateNames</code>	The names of the states.
<code>knownStates</code>	Vector of values of the state process which are known.
<code>covsDelta</code>	Design matrix for initial distribution.

### References

- Cornelissen, G. 2014. Cosinor-based rhythmometry. *Theoretical Biology and Medical Modelling* 11:16.
- Duchesne, T., Fortin, D., Rivest L-P. 2015. Equivalence between step selection functions and biased correlated random walks for statistical inference on animal movement. *PLoS ONE* 10 (4): e0122947.
- 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.

Leos-Barajas, V., Gangloff, E.J., Adam, T., Langrock, R., van Beest, F.M., Nabe-Nielsen, J. and Morales, J.M. 2017. Multi-scale modeling of animal movement and general behavior data using hidden Markov models with hierarchical structures. *Journal of Agricultural, Biological and Environmental Statistics*, 22 (3), 232-248.

Maruotti, A., and T. Ryden. 2009. A semiparametric approach to hidden Markov models under longitudinal observations. *Statistics and Computing* 19: 381-393.

McClintock B.T., King R., Thomas L., Matthiopoulos J., McConnell B.J., Morales J.M. 2012. A general discrete-time modeling framework for animal movement using multistate random walks. *Ecological Monographs*, 82 (3), 335-349.

McClintock B.T., Russell D.J., Matthiopoulos J., King R. 2013. Combining individual animal movement and ancillary biotelemetry data to investigate population-level activity budgets. *Ecology*, 94 (4), 838-849.

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.

Rivest, LP, Duchesne, T, Nicosia, A, Fortin, D, 2016. A general angular regression model for the analysis of data on animal movement in ecology. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 65(3):445-463.

### See Also

[getParDM](#), [prepData](#), [simData](#)

[simHierData](#)

### Examples

```
nbStates <- 2
stepDist <- "gamma" # step distribution
angleDist <- "vm" # turning angle distribution

# extract data from momentuHMM example
data <- example$m$data

### 1. 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)
stepPar <- c(mu0,sigma0) # no zero-inflation, so no zero-mass included
anglePar <- kappa0 # not estimating angle mean, so not included
formula <- ~cov1+cos(cov2)

m <- fitHMM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
            Par0=list(step=stepPar,angle=anglePar),formula=formula)

print(m)

## Not run:
### 2. fit the exact same model to the simulated data using DM formulas
```

```

# Get initial values for the parameters on working scale
Par0 <- getParDM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
  Par=list(step=stepPar,angle=anglePar),
  DM=list(step=list(mean=~1,sd=~1),angle=list(concentration=~1)))

mDMf <- fithMM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
  Par0=Par0,formula=formula,
  DM=list(step=list(mean=~1,sd=~1),angle=list(concentration=~1)))

print(mDMf)

### 3. fit the exact same model to the simulated data using DM matrices
# define DM
DMm <- list(step=diag(4),angle=diag(2))

# user-specified dimnames not required but are recommended
dimnames(DMm$step) <- list(c("mean_1","mean_2","sd_1","sd_2"),
  c("mean_1:(Intercept)","mean_2:(Intercept)",
  "sd_1:(Intercept)","sd_2:(Intercept)"))
dimnames(DMm$angle) <- list(c("concentration_1","concentration_2"),
  c("concentration_1:(Intercept)","concentration_2:(Intercept)"))

mDMm <- fithMM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
  Par0=Par0,formula=formula,
  DM=DMm)

print(mDMm)

### 4. fit step mean parameter covariate model to the simulated data using DM
stepDMf <- list(mean=~cov1,sd=~1)
Par0 <- getParDM(data,nbStates,list(step=stepDist,angle=angleDist),
  Par=list(step=stepPar,angle=anglePar),
  DM=list(step=stepDMf,angle=DMm$angle))
mDMfcov <- fithMM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
  Par0=Par0,
  formula=formula,
  DM=list(step=stepDMf,angle=DMm$angle))

print(mDMfcov)

### 5. fit the exact same step mean parameter covariate model using DM matrix
stepDMm <- matrix(c(1,0,0,0,"cov1",0,0,0,0,1,0,0,0,"cov1",0,0,
  0,0,1,0,0,0,0,1),4,6,dimnames=list(c("mean_1","mean_2","sd_1","sd_2"),
  c("mean_1:(Intercept)","mean_1:cov1","mean_2:(Intercept)","mean_2:cov1",
  "sd_1:(Intercept)","sd_2:(Intercept)")))
Par0 <- getParDM(data,nbStates,list(step=stepDist,angle=angleDist),
  Par=list(step=stepPar,angle=anglePar),
  DM=list(step=stepDMm,angle=DMm$angle))
mDMmcov <- fithMM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
  Par0=Par0,
  formula=formula,
  DM=list(step=stepDMm,angle=DMm$angle))

```

```

print(mDMmcov)

### 6. fit circular-circular angle mean covariate model to the simulated data using DM

# Generate fake circular covariate using circAngles
data$cov3 <- circAngles(refAngle=2*atan(rnorm(nrow(data))),data)

# Fit circular-circular regression model for angle mean
# Note no intercepts are estimated for angle means because these are by default
# the previous movement direction (i.e., a turning angle of zero)
mDMcircf <- fitHMM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
  Par0=list(step=stepPar,angle=c(0,0,Par0$angle)),
  formula=formula,
  estAngleMean=list(angle=TRUE),
  circularAngleMean=list(angle=TRUE),
  DM=list(angle=list(mean=~cov3,concentration=~1)))

print(mDMcircf)

### 7. fit the exact same circular-circular angle mean model using DM matrices

# Note no intercept terms are included in DM for angle means because the intercept is
# by default the previous movement direction (i.e., a turning angle of zero)
mDMcircm <- fitHMM(data=data,nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
  Par0=list(step=stepPar,angle=c(0,0,Par0$angle)),
  formula=formula,
  estAngleMean=list(angle=TRUE),
  circularAngleMean=list(angle=TRUE),
  DM=list(angle=matrix(c("cov3",0,0,0,0,"cov3",0,0,0,0,1,0,0,0,0,1),4,4)))

print(mDMcircm)

### 8. Cosinor and state-dependent formulas
nbStates<-2
dist<-list(step="gamma")
Par<-list(step=c(100,1000,50,100))

# include 24-hour cycle on all transition probabilities
# include 12-hour cycle on transitions from state 2
formula=~cosinor(hour24,24)+state2(cosinor(hour12,12))

# specify appropriate covariates
covs<-data.frame(hour24=0:23,hour12=0:11)

beta<-matrix(c(-1.5,1,1,NA,NA,-1.5,-1,-1,1,1),5,2)
# row names for beta not required but can be helpful
rownames(beta)<-c("(Intercept)",
  "cosinorCos(hour24, 24)",
  "cosinorSin(hour24, 24)",
  "cosinorCos(hour12, 12)",
  "cosinorSin(hour12, 12)")
data.cos<-simData(nbStates=nbStates,dist=dist,Par=Par,
  beta=beta,formula=formula,covs=covs)

```

```

m.cosinor<-fitHMM(data.cos,nbStates=nbStates,dist=dist,Par0=Par,formula=formula)
m.cosinor

### 9. Piecewise constant B-spline on step length mean and angle concentration
nObs <- 1000 # length of simulated track
cov <- data.frame(time=1:nObs) # time covariate for splines
dist <- list(step="gamma",angle="vm")
stepDM <- list(mean=~splines2::bSpline(time,df=2,degree=0),sd=~1)
angleDM <- list(mean=~1,concentration=~splines2::bSpline(time,df=2,degree=0))
DM <- list(step=stepDM,angle=angleDM)
Par <- list(step=c(log(1000),1,-1,log(100)),angle=c(0,log(10),2,-5))

data.spline<-simData(obsPerAnimal=nObs,nbStates=1,dist=dist,Par=Par,DM=DM,covs=cov)

Par0 <- list(step=Par$step,angle=Par$angle[-1])
m.spline<-fitHMM(data.spline,nbStates=1,dist=dist,Par0=Par0,
  DM=list(step=stepDM,
    angle=angleDM["concentration"]))

### 10. Initial state (delta) based on covariate
nObs <- 100
dist <- list(step="gamma",angle="vm")
Par <- list(step=c(100,1000,50,100),angle=c(0,0,0.01,0.75))

# create sex covariate
cov <- data.frame(sex=factor(rep(c("F","M"),each=nObs))) # sex covariate
formulaDelta <- ~ sex + 0

# Female begins in state 1, male begins in state 2
delta <- matrix(c(-100,100),2,1,dimnames=list(c("sexF","sexM"),"state 2"))

data.delta<-simData(nbAnimals=2,obsPerAnimal=nObs,nbStates=2,dist=dist,Par=Par,
  delta=delta,formulaDelta=formulaDelta,covs=cov)

Par0 <- list(step=Par$step, angle=Par$angle[3:4])
m.delta <- fitHMM(data.delta, nbStates=2, dist=dist, Par0 = Par0,
  formulaDelta=formulaDelta)

### 11. Two mixtures based on covariate
nObs <- 100
nbAnimals <- 20
dist <- list(step="gamma",angle="vm")
Par <- list(step=c(100,1000,50,100),angle=c(0,0,0.1,2))

# create sex covariate
cov <- data.frame(sex=factor(rep(c("F","M"),each=nObs*nbAnimals/2)))
formulaPi <- ~ sex + 0

# Females more likely in mixture 1, males more likely in mixture 2
beta <- list(beta=matrix(c(-1.5,-0.5,-1.5,-3),2,2),
  pi=matrix(c(-2,2),2,1,dimnames=list(c("sexF","sexM"),"mix2")))

```

```

data.mix<-simData(nbAnimals=nbAnimals,obsPerAnimal=nObs,nbStates=2,dist=dist,Par=Par,
                 beta=beta,formulaPi=formulaPi,mixtures=2,covs=cov)

Par0 <- list(step=Par$step, angle=Par$angle[3:4])
m.mix <- fithMM(data.mix, nbStates=2, dist=dist, Par0 = Par0,
               beta=beta,formulaPi=formulaPi,mixtures=2)

## End(Not run)

```

---

formatHierHMM

---

*Convert hierarchical HMM structure to a conventional HMM*


---

### Description

Convert hierarchical HMM structure to a conventional HMM

### Usage

```

formatHierHMM(
  data,
  hierStates,
  hierDist,
  hierBeta = NULL,
  hierDelta = NULL,
  hierFormula = NULL,
  hierFormulaDelta = NULL,
  mixtures = 1,
  workBounds = NULL,
  betaCons = NULL,
  deltaCons = NULL,
  fixPar = NULL,
  checkData = TRUE
)

```

### Arguments

data	<a href="#">momentuHierHMMDData</a> object or a data frame containing the data streams and covariates.
hierStates	A hierarchical data structure <a href="#">Node</a> for the states ('state'). See <a href="#">fithMM</a> .
hierDist	A hierarchical data structure <a href="#">Node</a> for the data streams ('dist'). See <a href="#">fithMM</a> .
hierBeta	A hierarchical data structure <a href="#">Node</a> for the matrix of initial values for the regression coefficients of the transition probabilities at each level of the hierarchy ('beta'). See <a href="#">fithMM</a> .
hierDelta	A hierarchical data structure <a href="#">Node</a> for the matrix of initial values for the regression coefficients of the initial distribution at each level of the hierarchy ('delta'). See <a href="#">fithMM</a> .

hierFormula	A hierarchical formula structure for the transition probability covariates for each level of the hierarchy ('formula'). See <a href="#">fitHMM</a> . Default: NULL (only hierarchical-level effects, with no covariate effects).
hierFormulaDelta	A hierarchical formula structure for the initial distribution covariates for each level of the hierarchy ('formulaDelta'). See <a href="#">fitHMM</a> . Default: NULL (no covariate effects and <code>fixPar\$delta</code> is specified on the working scale).
mixtures	Number of mixtures for the state transition probabilities (i.e. discrete random effects <i>sensu</i> DeRuiter et al. 2017). See <a href="#">fitHMM</a> . Default: mixtures=1.
workBounds	A list with elements named 'beta' and/or 'delta', where each element is a hierarchical data structure <a href="#">Node</a> indicating t.p.m. and initial distribution working parameter bounds ('workBounds') for parameters in <code>hierBeta</code> and <code>hierDelta</code> , respectively.
betaCons	A hierarchical data structure <a href="#">Node</a> indicating t.p.m. constraints ('betaCons') among parameters in <code>hierBeta</code> at each level of the hierarchy.
deltaCons	A hierarchical data structure <a href="#">Node</a> indicating initial distribution constraints ('deltaCons') among parameters in <code>hierDelta</code> at each level of the hierarchy.
fixPar	A list with elements named 'beta' and/or 'delta', where each element is a hierarchical data structure <a href="#">Node</a> indicating t.p.m. and initial distribution parameters in <code>hierBeta</code> and <code>hierDelta</code> , respectively, which are assumed known.
checkData	logical indicating whether or not to check the suitability of data for the specified hierarchy. Ignored unless data is provided. Default: TRUE.

### Value

A list of arguments needed for specifying a hierarchical HMM as a conventional HMM in [fitHMM](#) or [MIfitHMM](#), including:

nbStates	See <a href="#">fitHMM</a> .
dist	See <a href="#">fitHMM</a> .
formula	See <a href="#">fitHMM</a> .
formulaDelta	See <a href="#">fitHMM</a> .
beta0	See <a href="#">fitHMM</a> .
delta0	See <a href="#">fitHMM</a> .
betaRef	See <a href="#">fitHMM</a> .
betaCons	See <a href="#">fitHMM</a> .
deltaCons	See <a href="#">fitHMM</a> .
fixPar	See <a href="#">fitHMM</a> .
workBounds	See <a href="#">fitHMM</a> .
stateNames	See <a href="#">fitHMM</a> .



---

getCovNames	<i>Get names of any covariates used in probability distribution parameters</i>
-------------	--

---

**Description**

Get names of any covariates used in probability distribution parameters

**Usage**

```
getCovNames(m, p, distname)
```

**Arguments**

m	momentuHMM object
p	list returned by <code>parDef</code>
distname	Name of the data stream

**Value**

A list of:

DMterms	Names of all covariates included in the design matrix for the data stream
DMpartems	A list of the names of all covariates for each of the probability distribution parameters

---

getDM_rcpp	<i>Get design matrix</i>
------------	--------------------------

---

**Description**

Loop for creating full design matrix (X) from pseudo-design matrix (DM). Written in C++. Used in `getDM`.

**Usage**

```
getDM_rcpp(X, covs, DM, nr, nc, cov, nbObs)
```

**Arguments**

X	full design matrix
covs	matrix of covariates
DM	pseudo design matrix
nr	number of rows in design matrix
nc	number of column in design matrix
cov	covariate names
nbObs	number of observations

**Value**

full design matrix (X)

---

getPar	<i>Get starting values from momentuHMM, miHMM, or miSum object returned by fitHMM, MIfitHMM, or MIpool</i>
--------	--

---

**Description**

Get starting values from momentuHMM, miHMM, or miSum object returned by fitHMM, MIfitHMM, or MIpool

**Usage**

```
getPar(m)
```

**Arguments**

m                    A [momentuHMM](#), [miHMM](#), or [miSum](#) object.

**Value**

A list of parameter values (Par, beta, delta) that can be used as starting values in [fitHMM](#) or [MIfitHMM](#)

**See Also**

[getPar0](#), [getParDM](#)

**Examples**

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

---

getPar0	<i>Get starting values for new model from existing momentuHMM or momentuHierHMM model fit</i>
---------	---

---

### Description

For nested models, this function will extract starting parameter values (i.e., Par0 in `fithMM` or `MifithMM`) from an existing `momentuHMM` or `momentuHierHMM` model fit based on the provided arguments for the new model. Any parameters that are not in common between model and the new model (as specified by the arguments) are set to 0. This function is intended to help users incrementally build and fit more complicated models from simpler nested models (and vice versa).

### Usage

```
getPar0(model, ...)

## Default S3 method:
getPar0(
  model,
  nbStates = length(model$stateNames),
  estAngleMean = model$conditions$estAngleMean,
  circularAngleMean = model$conditions$circularAngleMean,
  formula = model$conditions$formula,
  formulaDelta = model$conditions$formulaDelta,
  stationary = model$conditions$stationary,
  mixtures = model$conditions$mixtures,
  formulaPi = model$conditions$formulaPi,
  DM = model$conditions$DM,
  betaRef = model$conditions$betaRef,
  stateNames = model$stateNames,
  ...
)

## S3 method for class 'hierarchical'
getPar0(
  model,
  hierStates = model$conditions$hierStates,
  estAngleMean = model$conditions$estAngleMean,
  circularAngleMean = model$conditions$circularAngleMean,
  hierFormula = model$conditions$hierFormula,
  hierFormulaDelta = model$conditions$hierFormulaDelta,
  mixtures = model$conditions$mixtures,
  formulaPi = model$conditions$formulaPi,
  DM = model$conditions$DM,
  ...
)
```

**Arguments**

model	A <code>momentuHMM</code> , <code>momentuHierHMM</code> , <code>miHMM</code> , or <code>miSum</code> object (as returned by <code>fitHMM</code> , <code>MIfitHMM</code> , or <code>MIpool</code> )
...	further arguments passed to or from other methods
nbStates	Number of states in the new model. Default: <code>nbStates=length(model\$stateNames)</code>
estAngleMean	Named list indicating whether or not the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy') are to be estimated in the new model. Default: <code>estAngleMean=model\$conditions\$estAngleMean</code>
circularAngleMean	Named list indicating whether circular-linear or circular-circular regression on the mean of circular distributions ('vm' and 'wrpcauchy') for turning angles are to be used in the new model. See <code>fitHMM</code> . Default: <code>circularAngleMean=model\$conditions\$circularAngleMean</code>
formula	Regression formula for the transition probability covariates of the new model (see <code>fitHMM</code> ). Default: <code>formula=model\$conditions\$formula</code> .
formulaDelta	Regression formula for the initial distribution covariates of the new model (see <code>fitHMM</code> ). Default: <code>formulaDelta=model\$conditions\$formulaDelta</code> .
stationary	FALSE if there are time-varying covariates in <code>formula</code> or any covariates in <code>formulaDelta</code> . If TRUE, the initial distribution is considered equal to the stationary distribution. Default: FALSE.
mixtures	Number of mixtures for the state transition probabilities (see <code>fitHMM</code> ). Default: <code>formula=model\$conditions\$mixtures</code> .
formulaPi	Regression formula for the mixture distribution probabilities (see <code>fitHMM</code> ). Default: <code>formula=model\$conditions\$formulaPi</code> .
DM	Named list indicating the design matrices to be used for the probability distribution parameters of each data stream in the new model (see <code>fitHMM</code> ). Only parameters with design matrix column names that match those in <code>model\$conditions\$fullDM</code> are extracted, so care must be taken in naming columns if any elements of <code>DM</code> are specified as matrices instead of formulas. Default: <code>DM=model\$conditions\$DM</code> .
betaRef	Numeric vector of length <code>nbStates</code> indicating the reference elements for the t.p.m. multinomial logit link. Default: <code>formula=model\$conditions\$betaRef</code> .
stateNames	Character vector of length <code>nbStates</code> indicating the names and order of the states in the new model. Default: <code>stateNames=model\$stateNames[1:nbStates]</code> .
hierStates	A hierarchical model structure <code>Node</code> for the states (see <code>fitHMM</code> ). Default: <code>hierStates=model\$conditions\$hierStates</code> .
hierFormula	A hierarchical formula structure for the transition probability covariates for each level of the hierarchy (see <code>fitHMM</code> ). Default: <code>hierFormula=model\$conditions\$hierFormula</code> .
hierFormulaDelta	A hierarchical formula structure for the initial distribution covariates for each level of the hierarchy ('formulaDelta'). Default: NULL (no covariate effects and <code>fixPar\$delta</code> is specified on the working scale).

**Details**

All other `fitHMM` (or `MIfitHMM`) model specifications (e.g., `dist`, `hierDist`, `userBounds`, `workBounds`, etc.) and data are assumed to be the same for `model` and the new model (as specified by `nbStates`,

hierStates, estAngleMean, circularAngleMean, formula, hierFormula, formulaDelta, hierFormulaDelta, DM, etc.).

Note that for hierarchical models, getPar0 will return hierarchical data.tree objects (hierBeta and hierDelta) with the default values for fixPar, betaCons, and deltaCons; if hierarchical t.p.m. or initial distribution parameters are subject to constraints, then these must be set manually on the list object returned by getPar0.

### Value

A named list containing starting values suitable for Par0 and beta0 arguments in `fitHMM` or `MIfitHMM`:

Par	A list of vectors of state-dependent probability distribution parameters for each data stream specified in <code>model\$conditions\$dist</code>
beta	Matrix of regression coefficients for the transition probabilities
delta	Initial distribution of the HMM. Only returned if <code>stateNames</code> has the same membership as the state names for <code>model</code>

### See Also

[getPar](#), [getParDM](#), [fitHMM](#), [MIfitHMM](#)

### Examples

```
# model is a momentuHMM object, automatically loaded with the package
model <- example$m
data <- model$data
dist <- model$conditions$dist
nbStates <- length(model$stateNames)
estAngleMean <- model$conditions$estAngleMean

newformula <- ~cov1+cov2
Par0 <- getPar0(model, formula=newformula)

## Not run:
newModel <- fitHMM(model$data, dist=dist, nbStates=nbStates,
                  Par0=Par0$Par, beta0=Par0$beta,
                  formula=newformula,
                  estAngleMean=estAngleMean)

## End(Not run)

newDM1 <- list(step=list(mean=~cov1, sd=~cov1))
Par0 <- getPar0(model, DM=newDM1)

## Not run:
newModel1 <- fitHMM(model$data, dist=dist, nbStates=nbStates,
                  Par0=Par0$Par, beta0=Par0$beta,
                  formula=model$conditions$formula,
                  estAngleMean=estAngleMean,
                  DM=newDM1)
```

```
## End(Not run)

# same model but specify DM for step using matrices
newDM2 <- list(step=matrix(c(1,0,0,0,
                             "cov1",0,0,0,
                             0,1,0,0,
                             0,"cov1",0,0,
                             0,0,1,0,
                             0,0,"cov1",0,
                             0,0,0,1,
                             0,0,0,"cov1"),nrow=nbStates*2))

# to be extracted, new design matrix column names must match
# column names of model$conditions$fullDM
colnames(newDM2$step)<-paste0(rep(c("mean_", "sd_"),each=2*nbStates),
                             rep(1:nbStates,each=2),
                             rep(c(":(Intercept)", ":(cov1)"),2*nbStates))
Par0 <- getPar0(model,DM=newDM2)

## Not run:
newModel2 <- fitHMM(model$data,dist=dist,nbStates=nbStates,
                    Par0=Par0$Par,beta0=Par0$beta,
                    formula=model$conditions$formula,
                    estAngleMean=estAngleMean,
                    DM=newDM2)

## End(Not run)
```

---

getParDM

*Get starting values on working scale based on design matrix and other parameter constraints*

---

## Description

Convert starting values on the natural scale of data stream probability distributions to a feasible set of working scale parameters based on a design matrix and other parameter constraints.

## Usage

```
getParDM(data, ...)

## Default S3 method:
getParDM(
  data = data.frame(),
  nbStates,
  dist,
  Par,
  zeroInflation = NULL,
```

```

    oneInflation = NULL,
    estAngleMean = NULL,
    circularAngleMean = NULL,
    DM = NULL,
    userBounds = NULL,
    workBounds = NULL,
    ...
)

## S3 method for class 'hierarchical'
getParDM(
  data = data.frame(),
  hierStates,
  hierDist,
  Par,
  zeroInflation = NULL,
  oneInflation = NULL,
  estAngleMean = NULL,
  circularAngleMean = NULL,
  DM = NULL,
  userBounds = NULL,
  workBounds = NULL,
  ...
)

```

### Arguments

data	Optional <a href="#">momentuHMMData</a> object, <a href="#">momentuHierHMMData</a> object, or a data frame containing the covariate values. data must be specified if covariates are included in DM.  If a data frame is provided, then either nbStates and dist must be specified (for a regular HMM) or hierStates and hierDist must be specified (for a hierarchical HMM).
...	further arguments passed to or from other methods
nbStates	Number of states of the HMM.
dist	A named list indicating the probability distributions of the data streams. Currently supported distributions are 'bern', 'beta', 'exp', 'gamma', 'lnorm', 'norm', 'mvnorm2' (bivariate normal distribution), 'mvnorm3' (trivariate normal distribution), 'pois', 'rw_norm' (normal random walk), 'rw_mvnorm2' (bivariate normal random walk), 'rw_mvnorm3' (trivariate normal random walk), 'vm', 'vm-Consensus', 'weibull', and 'wrpcauchy'. For example, <code>dist=list(step='gamma', angle='vm', dives=)</code> indicates 3 data streams ('step', 'angle', and 'dives') and their respective probability distributions ('gamma', 'vm', and 'pois').
Par	A named list containing vectors of state-dependent probability distribution parameters for each data stream specified in dist. The parameters should be on the natural scale, in the order expected by the pdfs of dist, and any zero-mass parameters should be the last.

- zeroInflation** A named list of logicals indicating whether the probability distributions of the data streams should be zero-inflated. If `zeroInflation` is TRUE for a given data stream, then values for the zero-mass parameters should be included in the corresponding element of `Par`. Ignored if data is a `momentuHMMDData` or `momentuHierHMMDData` object.
- oneInflation** Named list of logicals indicating whether the probability distributions of the data streams are one-inflated. If `oneInflation` is TRUE for a given data stream, then values for the one-mass parameters should be included in the corresponding element of `Par`. Ignored if data is a `momentuHMMDData` or `momentuHierHMMDData` object.
- estAngleMean** An optional named list indicating whether or not to estimate the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy'). Any `estAngleMean` elements corresponding to data streams that do not have angular distributions are ignored.
- circularAngleMean** An optional named list indicating whether to use circular-linear or circular-circular regression on the mean of circular distributions ('vm' and 'wrpcauchy') for turning angles. See `fitHMM`. `circularAngleMean` elements corresponding to angular data streams are ignored unless the corresponding element of `estAngleMean` is TRUE. Any `circularAngleMean` elements corresponding to data streams that do not have angular distributions are ignored.
- DM** A named list indicating the design matrices to be used for the probability distribution parameters of each data stream. Each element of `DM` can either be a named list of linear regression formulas or a matrix. For example, for a 2-state model using the gamma distribution for a data stream named 'step', `DM=list(step=list(mean=~cov1,sd=~1))` specifies the mean parameters as a function of the covariate 'cov1' for each state. This model could equivalently be specified as a 4x6 matrix using character strings for the covariate: `DM=list(step=matrix(c(1,0,0,0,'cov1',0,0,0,0,1,0,0,0,'cov1',0,0,0,0,1,0,0,0,0,1),4,6))` where the 4 rows correspond to the state-dependent parameters (mean\_1,mean\_2,sd\_1,sd\_2) and the 6 columns correspond to the regression coefficients.
- userBounds** An optional named list of 2-column matrices specifying bounds on the natural (i.e, real) scale of the probability distribution parameters for each data stream. For example, for a 2-state model using the wrapped Cauchy ('wrpcauchy') distribution for a data stream named 'angle' with `estAngleMean$angle=TRUE`, `userBounds=list(angle=matrix(c(-pi,-pi,-1,-1,pi,pi,1,1),4,2))` specifies (-1,1) bounds for the concentration parameters instead of the default [0,1) bounds.
- workBounds** An optional named list of 2-column matrices specifying bounds on the working scale of the probability distribution, transition probability, and initial distribution parameters. For each matrix, the first column pertains to the lower bound and the second column the upper bound. For data streams, each element of `workBounds` should be a  $k \times 2$  matrix with the same name of the corresponding element of `Par0`, where  $k$  is the number of parameters. For transition probability parameters, the corresponding element of `workBounds` must be a  $k \times 2$  matrix named "beta", where  $k=\text{length}(\text{beta}0)$ . For initial distribution parameters, the



corresponding element of `workBounds` must be a  $k \times 2$  matrix named “delta”, where  $k = \text{length}(\text{delta}0)$ .

`hierStates` A hierarchical model structure [Node](#) for the states. See [fithMM](#).

`hierDist` A hierarchical data structure [Node](#) for the data streams. See [fithMM](#).

## Details

If design matrix includes non-factor covariates, then natural scale parameters are assumed to correspond to the mean value(s) for the covariate(s) (if  $\text{nrow}(\text{data}) > 1$ ) and `getParDM` simply returns one possible solution to the system of linear equations defined by `Par`, `DM`, and any other constraints using singular value decomposition. This can be helpful for exploring relationships between the natural and working scale parameters when covariates are included, but `getParDM` will not necessarily return “good” starting values (i.e., `Par0`) for [fithMM](#) or [MIfithMM](#).

## Value

A list of parameter values that can be used as starting values (`Par0`) in [fithMM](#) or [MIfithMM](#)

## See Also

[getPar](#), [getPar0](#), [fithMM](#), [MIfithMM](#)

## Examples

```
# data is a momentuHMMDData object, automatically loaded with the package
data <- example$m$data
stepDist <- "gamma"
angleDist <- "vm"
nbStates <- 2
stepPar0 <- c(15,50,10,20) # natural scale mean_1, mean_2, sd_1, sd_2
anglePar0 <- c(0.7,1.5) # natural scale concentration_1, concentration_2

# get working parameters for 'DM' that constrains step length mean_1 < mean_2
stepDM <- matrix(c(1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1),4,4,
  dimnames=list(NULL,c("mean:(Intercept)", "mean_2",
    "sd_1:(Intercept)", "sd_2:(Intercept)")))
stepworkBounds <- matrix(c(-Inf,Inf),4,2,byrow=TRUE,
  dimnames=list(colnames(stepDM),c("lower", "upper")))
stepworkBounds["mean_2", "lower"] <- 0 #coefficient for 'mean_2' constrained to be positive
wPar0 <- getParDM(nbStates=2, dist=list(step=stepDist),
  Par=list(step=stepPar0),
  DM=list(step=stepDM), workBounds=list(step=stepworkBounds))

## Not run:
# Fit HMM using wPar0 as initial values for the step data stream
mPar <- fithMM(data, nbStates=2, dist=list(step=stepDist, angle=angleDist),
  Par0=list(step=wPar0$step, angle=anglePar0),
  DM=list(step=stepDM), workBounds=list(step=stepworkBounds))

## End(Not run)
```

```

# get working parameters for 'DM' using 'cov1' and 'cov2' covariates
stepDM2 <- list(mean=~cov1,sd=~cov2)
wPar20 <- getParDM(data,nbStates=2,dist=list(step=stepDist),
                  Par=list(step=stepPar0),
                  DM=list(step=stepDM2))

## Not run:
# Fit HMM using wPar20 as initial values for the step data stream
mPar2 <- fitHMM(data,nbStates=2,dist=list(step=stepDist,angle=angleDist),
                Par0=list(step=wPar20$step,angle=anglePar0),
                DM=list(step=stepDM2))

## End(Not run)

```

---

getTrProbs

*Transition probability matrix*


---

### Description

Computation of the transition probability matrix for each time step as a function of the covariates and the regression parameters.

### Usage

```

getTrProbs(data, ...)

## Default S3 method:
getTrProbs(
  data,
  nbStates,
  beta,
  workBounds = NULL,
  formula = ~1,
  mixtures = 1,
  betaRef = NULL,
  stateNames = NULL,
  getCI = FALSE,
  covIndex = NULL,
  alpha = 0.95,
  ...
)

## S3 method for class 'hierarchical'
getTrProbs(
  data,
  hierStates,
  hierBeta,

```

```

workBounds = NULL,
hierFormula = NULL,
mixtures = 1,
hierDist,
getCI = FALSE,
covIndex = NULL,
alpha = 0.95,
...
)

```

## Arguments

data	<p><a href="#">momentuHMM</a> object, <a href="#">momentuHierHMM</a> object, <a href="#">miSum</a> object, <a href="#">miHMM</a> object, <a href="#">momentuHMMData</a> object, <a href="#">momentuHierHMMData</a> object, or a data frame containing the covariate values.</p> <p>If a data frame is provided, then either nbStates must be specified (for a regular HMM) or hierStates and hierDist must be specified (for a hierarchical HMM).</p>
...	further arguments passed to or from other methods; most are ignored if data is a <a href="#">momentuHMM</a> or <a href="#">momentuHierHMM</a> object
nbStates	Number of states. Ignored unless data is a data frame.
beta	Matrix of regression coefficients for the transition probabilities
workBounds	An optional named list of 2-column matrices specifying bounds on the working scale of the transition probability parameters ('beta' and, for recharge models, 'g0' and 'theta'). workBounds\$beta must be a k x 2 matrix, where k=length(beta). The first column pertains to the lower bound and the second column the upper bound. Ignored unless data is a data frame.
formula	Regression formula for the transition probability covariates. Ignored unless data is a data frame.
mixtures	Number of mixtures for the state transition probabilities. Ignored unless data is a data frame.
betaRef	Indices of reference elements for t.p.m. multinomial logit link. Ignored unless data is a data frame.
stateNames	Optional character vector of length nbStates indicating state names. Ignored unless data is a data frame.
getCI	Logical indicating whether to calculate standard errors and logit-transformed confidence intervals based on fitted <a href="#">momentuHMM</a> or <a href="#">momentuHierHMM</a> object. Default: FALSE.
covIndex	Integer vector indicating specific rows of the data to be used in the calculations. This can be useful for reducing unnecessarily long computation times (particularly when getCI=TRUE), e.g., when formula includes factor covariates (such as ID) but no temporal covariates. Ignored if data is not a <a href="#">momentuHMM</a> , <a href="#">momentuHierHMM</a> , <a href="#">miSum</a> , or <a href="#">miHMM</a> object.
alpha	Significance level of the confidence intervals (if getCI=TRUE). Default: 0.95 (i.e. 95% CIs).

hierStates	A hierarchical model structure <a href="#">Node</a> for the states ('state'). See <a href="#">fitHMM</a> .
hierBeta	A hierarchical data structure <a href="#">Node</a> for the matrix of regression coefficients for the transition probabilities at each level of the hierarchy, including initial values ('beta'), parameter equality constraints ('betaCons'), fixed parameters ('fixPar'), and working scale bounds ('workBounds'). See details.
hierFormula	A hierarchical formula structure for the transition probability covariates for each level of the hierarchy ('formula'). See <a href="#">fitHMM</a> .
hierDist	A hierarchical data structure <a href="#">Node</a> for the data streams ('dist'). See <a href="#">fitHMM</a> .

## Value

If mixtures=1, an array of dimension nbStates x nbStates x nrow(data) containing the t.p.m for each observation in data. If mixtures>1, a list of length mixtures, where each element is an array of dimension nbStates x nbStates x nrow(data) containing the t.p.m for each observation in data.

If getCI=TRUE then a list of arrays is returned (with elements est, se, lower, and upper).

If a hierarchical HMM structure is provided, then a hierarchical data structure containing the state transition probabilities for each time step at each level of the hierarchy ('gamma') is returned.

## Examples

```
m <- example$m
trProbs <- getTrProbs(m)

# equivalent
trProbs <- getTrProbs(m$data,nbStates=2,beta=m$mle$beta,formula=m$conditions$formula)

## Not run:
# calculate SEs and 95% CIs
trProbsSE <- getTrProbs(m, getCI=TRUE)

# plot estimates and CIs for each state transition
par(mfrow=c(2,2))
for(i in 1:2){
  for(j in 1:2){
    plot(trProbsSE$est[i,j,],type="l",
         ylim=c(0,1), ylab=paste(i,"->",j))
    arrows(1:dim(trProbsSE$est)[3],
          trProbsSE$lower[i,j,],
          1:dim(trProbsSE$est)[3],
          trProbsSE$upper[i,j,],
          length=0.025, angle=90, code=3, col=gray(.5), lwd=1.3)
  }
}

# limit calculations to first 10 observations
trProbsSE_10 <- getTrProbs(m, getCI=TRUE, covIndex=1:10)

## End(Not run)
```

---

HMMfits	<i>Constructor of HMMfits objects</i>
---------	---------------------------------------

---

**Description**

Constructor of HMMfits objects

**Usage**

```
HMMfits(m)
```

**Arguments**

`m` A list of [momentuHMM](#) objects.  
HMMfits objects are returned by [MifitHMM](#) when arguments `fit=TRUE` and `poolEstimates=FALSE`.

**Value**

An object HMMfits.

---

<code>is.crwData</code>	<i>Is crwData</i>
-------------------------	-------------------

---

**Description**

Check that an object is of class [crwData](#). Used in [MifitHMM](#).

**Usage**

```
is.crwData(x)
```

**Arguments**

`x` An R object

**Value**

TRUE if `x` is of class [crwData](#), FALSE otherwise.

---

is.crwHierData	<i>Is crwHierData</i>
----------------	-----------------------

---

**Description**

Check that an object is of class `crwHierData`. Used in `MIfitHMM`.

**Usage**

```
is.crwHierData(x)
```

**Arguments**

x                    An R object

**Value**

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

---

is.crwHierSim	<i>Is crwHierSim</i>
---------------	----------------------

---

**Description**

Check that an object is of class `crwHierSim`.

**Usage**

```
is.crwHierSim(x)
```

**Arguments**

x                    An R object

**Value**

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

---

is.crwSim	<i>Is crwSim</i>
-----------	------------------

---

**Description**

Check that an object is of class `crwSim`.

**Usage**

```
is.crwSim(x)
```

**Arguments**

x                    An R object

**Value**

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

---

is.HMMfits	<i>Is HMMfits</i>
------------	-------------------

---

**Description**

Check that an object is of class `HMMfits`.

**Usage**

```
is.HMMfits(x)
```

**Arguments**

x                    An R object

**Value**

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

---

`is.miHMM`*Is miHMM*

---

**Description**

Check that an object is of class `miHMM`.

**Usage**

```
is.miHMM(x)
```

**Arguments**

`x` An R object

**Value**

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

---

`is.miSum`*Is miSum*

---

**Description**

Check that an object is of class `miSum`.

**Usage**

```
is.miSum(x)
```

**Arguments**

`x` An R object

**Value**

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



---

is.momentuHierHMM      *Is momentuHierHMM*

---

**Description**

Check that an object is of class `momentuHierHMM`. Used in `CIreal`, `CIbeta`, `plotPR`, `plotStates`, `pseudoRes`, `stateProbs`, and `viterbi`.

**Usage**

```
is.momentuHierHMM(x)
```

**Arguments**

x                      An R object

**Value**

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

---

is.momentuHierHMMData      *Is momentuHierHMMData*

---

**Description**

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

**Usage**

```
is.momentuHierHMMData(x)
```

**Arguments**

x                      An R object

**Value**

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

is.momentuHMM            *Is momentuHMM*

---

**Description**

Check that an object is of class `momentuHMM`. Used in `CIreal`, `CIbeta`, `plotPR`, `plotStates`, `pseudoRes`, `stateProbs`, and `viterbi`.

**Usage**

```
is.momentuHMM(x)
```

**Arguments**

x                      An R object

**Value**

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

---

is.momentuHMMData        *Is momentuHMMData*

---

**Description**

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

**Usage**

```
is.momentuHMMData(x)
```

**Arguments**

x                      An R object

**Value**

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

---

logAlpha	<i>Forward log-probabilities</i>
----------	----------------------------------

---

**Description**

Used in [stateProbs](#) and [pseudoRes](#).

**Usage**

```
logAlpha(m)
```

**Arguments**

`m` A [momentuHMM](#), [miHMM](#), or [miSum](#) object.

**Value**

A list of length `model$conditions$mixtures` where each element is a matrix of forward log-probabilities for each mixture.

**Examples**

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

---

logBeta	<i>Backward log-probabilities</i>
---------	-----------------------------------

---

**Description**

Used in [stateProbs](#).

**Usage**

```
logBeta(m)
```

**Arguments**

`m` A [momentuHMM](#), [miHMM](#), or [miSum](#) object.

**Value**

A list of length `model$conditions$mixtures` where each element is a matrix of backward log-probabilities for each mixture.

**Examples**

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

lb <- momentuHMM::logBeta(m)

## End(Not run)
```

---

MIfitHMM

*Fit HMMs to multiple imputation data*


---

**Description**

Fit a (multivariate) hidden Markov model to multiple imputation data. Multiple imputation is a method for accommodating missing data, temporal-irregularity, or location measurement error in hidden Markov models, where pooled parameter estimates reflect uncertainty attributable to observation error.

**Usage**

```
MIfitHMM(miData, ...)

## Default S3 method:
MIfitHMM(
  miData,
  nSims,
  ncores = 1,
  poolEstimates = TRUE,
  alpha = 0.95,
  na.rm = FALSE,
  nbStates,
  dist,
  Par0,
  beta0 = NULL,
  delta0 = NULL,
  estAngleMean = NULL,
  circularAngleMean = NULL,
  formula = ~1,
  formulaDelta = NULL,
  stationary = FALSE,
  mixtures = 1,
```

```
    formulaPi = NULL,  
    nlmPar = NULL,  
    fit = TRUE,  
    useInitial = FALSE,  
    DM = NULL,  
    userBounds = NULL,  
    workBounds = NULL,  
    betaCons = NULL,  
    betaRef = NULL,  
    deltaCons = NULL,  
    mvnCoords = NULL,  
    stateNames = NULL,  
    knownStates = NULL,  
    fixPar = NULL,  
    retryFits = 0,  
    retrySD = NULL,  
    optMethod = "nlm",  
    control = list(),  
    prior = NULL,  
    modelName = NULL,  
    covNames = NULL,  
    spatialCovs = NULL,  
    centers = NULL,  
    centroids = NULL,  
    angleCovs = NULL,  
    altCoordNames = NULL,  
    method = "IS",  
    parIS = 1000,  
    dfSim = Inf,  
    grid.eps = 1,  
    crit = 2.5,  
    scaleSim = 1,  
    quad.ask = FALSE,  
    force.quad = TRUE,  
    fullPost = TRUE,  
    dfPostIS = Inf,  
    scalePostIS = 1,  
    thetaSamp = NULL,  
    ...  
)  
  
## S3 method for class 'hierarchical'  
MifitHMM(  
  miData,  
  nSims,  
  ncores = 1,  
  poolEstimates = TRUE,  
  alpha = 0.95,
```

```
na.rm = FALSE,
hierStates,
hierDist,
Par0,
hierBeta = NULL,
hierDelta = NULL,
estAngleMean = NULL,
circularAngleMean = NULL,
hierFormula = NULL,
hierFormulaDelta = NULL,
mixtures = 1,
formulaPi = NULL,
nlmPar = NULL,
fit = TRUE,
useInitial = FALSE,
DM = NULL,
userBounds = NULL,
workBounds = NULL,
betaCons = NULL,
deltaCons = NULL,
mvnCoords = NULL,
knownStates = NULL,
fixPar = NULL,
retryFits = 0,
retrySD = NULL,
optMethod = "nlm",
control = list(),
prior = NULL,
modelName = NULL,
covNames = NULL,
spatialCovs = NULL,
centers = NULL,
centroids = NULL,
angleCovs = NULL,
altCoordNames = NULL,
method = "IS",
parIS = 1000,
dfSim = Inf,
grid.eps = 1,
crit = 2.5,
scaleSim = 1,
quad.ask = FALSE,
force.quad = TRUE,
fullPost = TRUE,
dfPostIS = Inf,
scalePostIS = 1,
thetaSamp = NULL,
...
```

)

**Arguments**

miData	A <code>crwData</code> object, a <code>crwHierData</code> object, a <code>crwSim</code> object, a <code>crwHierSim</code> object, a list of <code>momentuHMMData</code> objects, or a list of <code>momentuHierHMMData</code> objects.
...	further arguments passed to or from other methods
nSims	Number of imputations in which to fit the HMM using <code>fithMM</code> . If <code>miData</code> is a list of <code>momentuHMMData</code> objects, <code>nSims</code> cannot exceed the length of <code>miData</code> .
ncores	Number of cores to use for parallel processing. Default: 1 (no parallel processing).
poolEstimates	Logical indicating whether or not to calculate pooled parameter estimates across the <code>nSims</code> imputations using <code>MIPool</code> . Default: TRUE.
alpha	Significance level for calculating confidence intervals of pooled estimates when <code>poolEstimates=TRUE</code> (see <code>MIPool</code> ). Default: 0.95.
na.rm	Logical indicating whether or not to exclude model fits with NA parameter estimates or standard errors from pooling when <code>poolEstimates=TRUE</code> (see <code>MIPool</code> ). Default: FALSE.
nbStates	Number of states of the HMM. See <code>fithMM</code> .
dist	A named list indicating the probability distributions of the data streams. See <code>fithMM</code> .
Par0	A named list containing vectors of initial state-dependent probability distribution parameters for each data stream specified in <code>dist</code> . See <code>fithMM</code> . <code>Par0</code> may also be a list of length <code>nSims</code> , where each element is a named list containing vectors of initial state-dependent probability distribution parameters for each imputation. Note that if <code>useInitial=TRUE</code> then <code>Par0</code> is ignored after the first imputation.
beta0	Initial matrix of regression coefficients for the transition probabilities. See <code>fithMM</code> . <code>beta0</code> may also be a list of length <code>nSims</code> , where each element is an initial matrix of regression coefficients for the transition probabilities for each imputation.
delta0	Initial values for the initial distribution of the HMM. See <code>fithMM</code> . <code>delta0</code> may also be a list of length <code>nSims</code> , where each element is the initial values for the initial distribution of the HMM for each imputation.
estAngleMean	An optional named list indicating whether or not to estimate the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy'). See <code>fithMM</code> .
circularAngleMean	An optional named list indicating whether to use circular-linear or circular-circular regression on the mean of circular distributions ('vm' and 'wrpcauchy') for turning angles. See <code>fithMM</code> .
formula	Regression formula for the transition probability covariates. See <code>fithMM</code> .
formulaDelta	Regression formula for the initial distribution. See <code>fithMM</code> .
stationary	FALSE if there are time-varying covariates in <code>formula</code> or any covariates in <code>formulaDelta</code> . If TRUE, the initial distribution is considered equal to the stationary distribution. See <code>fithMM</code> .

<code>mixtures</code>	Number of mixtures for the state transition probabilities (i.e. discrete random effects <i>sensu</i> DeRuiter et al. 2017). Default: <code>mixtures=1</code> .
<code>formulaPi</code>	Regression formula for the mixture distribution probabilities. See <code>fitHMM</code> .
<code>nlmPar</code>	List of parameters to pass to the optimization function <code>nlm</code> (which should be either <code>print.level</code> , <code>gradtol</code> , <code>stepmax</code> , <code>steptol</code> , <code>iterlim</code> , or <code>hessian</code> – see <code>nlm</code> 's documentation for more detail). For <code>print.level</code> , 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. Ignored unless <code>optMethod="nlm"</code> .
<code>fit</code>	TRUE if the HMM should be fitted to the data, FALSE otherwise. See <code>fitHMM</code> . If <code>fit=FALSE</code> and <code>miData</code> is a <code>crwData</code> object, then <code>MIfitHMM</code> returns a list containing a <code>momentuHMMData</code> object (if <code>nSims=1</code> ) or, if <code>nSims&gt;1</code> , a <code>crwSim</code> object.
<code>useInitial</code>	Logical indicating whether or not to use parameter estimates for the first model fit as initial values for all subsequent model fits. If <code>ncores&gt;1</code> then the first model is fit on a single core and then used as the initial values for all subsequent model fits on each core (in this case, the progress of the initial model fit can be followed using the <code>print.level</code> option in <code>nlmPar</code> ). Default: FALSE. Ignored if <code>nSims&lt;2</code> .
<code>DM</code>	An optional named list indicating the design matrices to be used for the probability distribution parameters of each data stream. See <code>fitHMM</code> .
<code>userBounds</code>	An optional named list of 2-column matrices specifying bounds on the natural (i.e. real) scale of the probability distribution parameters for each data stream. See <code>fitHMM</code> .
<code>workBounds</code>	An optional named list of 2-column matrices specifying bounds on the working scale of the probability distribution, transition probability, and initial distribution parameters. See <code>fitHMM</code> .
<code>betaCons</code>	Matrix of the same dimension as <code>beta0</code> composed of integers identifying any equality constraints among the t.p.m. parameters. See <code>fitHMM</code> .
<code>betaRef</code>	Numeric vector of length <code>nbStates</code> indicating the reference elements for the t.p.m. multinomial logit link. See <code>fitHMM</code> .
<code>deltaCons</code>	Matrix of the same dimension as <code>delta0</code> composed of integers identifying any equality constraints among the initial distribution working scale parameters. Ignored unless a formula is provided in <code>formulaDelta</code> . See <code>fitHMM</code> .
<code>mvnCoords</code>	Character string indicating the name of location data that are to be modeled using a multivariate normal distribution. For example, if <code>mu="mvnorm2"</code> was included in <code>dist</code> and <code>(mu.x, mu.y)</code> are location data, then <code>mvnCoords="mu"</code> needs to be specified in order for these data to be properly treated as locations in functions such as <code>plot.momentuHMM</code> , <code>plot.miSum</code> , <code>plot.miHMM</code> , <code>plotSpatialCov</code> , and <code>MIpool</code> .
<code>stateNames</code>	Optional character vector of length <code>nbStates</code> indicating state names.
<code>knownStates</code>	Vector of values of the state process which are known prior to fitting the model (if any). See <code>fitHMM</code> . If <code>miData</code> is a list of <code>momentuHMMData</code> objects, then <code>knownStates</code> can alternatively be a list of vectors containing the known values for the state process for each element of <code>miData</code> .



<code>fixPar</code>	An optional list of vectors indicating parameters which are assumed known prior to fitting the model. See <code>fitHMM</code> .
<code>retryFits</code>	Non-negative integer indicating the number of times to attempt to iteratively fit the model using random perturbations of the current parameter estimates as the initial values for likelihood optimization. See <code>fitHMM</code> .
<code>retrySD</code>	An optional list of scalars or vectors indicating the standard deviation to use for normal perturbations of each working scale parameter when <code>retryFits&gt;0</code> . See <code>fitHMM</code> .
<code>optMethod</code>	The optimization method to be used. Can be “nlm” (the default; see <code>nlm</code> ), “Nelder-Mead” (see <code>optim</code> ), or “SANN” (see <code>optim</code> ).
<code>control</code>	A list of control parameters to be passed to <code>optim</code> (ignored unless <code>optMethod=“Nelder-Mead”</code> or <code>optMethod=“SANN”</code> ).
<code>prior</code>	A function that returns the log-density of the working scale parameter prior distribution(s). See <code>fitHMM</code> .
<code>modelName</code>	An optional character string providing a name for the fitted model. If provided, <code>modelName</code> will be returned in <code>print.momentuHMM</code> , <code>AIC.momentuHMM</code> , <code>AICweights</code> , and other functions.
<code>covNames</code>	Names of any covariates in <code>miData\$crwPredict</code> (if <code>miData</code> is a <code>crwData</code> object; otherwise <code>covNames</code> is ignored). See <code>prepData</code> . If <code>miData</code> is a <code>crwData</code> object, any covariate in <code>miData\$crwPredict</code> that is used in <code>formula</code> , <code>formulaDelta</code> , <code>formulaPi</code> , or <code>DM</code> must be included in <code>covNames</code> .
<code>spatialCovs</code>	List of raster layer(s) for any spatial covariates. See <code>prepData</code> .
<code>centers</code>	2-column matrix providing the x-coordinates (column 1) and y-coordinates (column 2) for any activity centers (e.g., potential centers of attraction or repulsion) from which distance and angle covariates will be calculated based on realizations of the position process. See <code>prepData</code> . Ignored unless <code>miData</code> is a <code>crwData</code> object.
<code>centroids</code>	List where each element is a data frame containing the x-coordinates (‘x’), y-coordinates (‘y’), and times (with user-specified name, e.g., ‘time’) for centroids (i.e., dynamic activity centers where the coordinates can change over time) from which distance and angle covariates will be calculated based on the location data. See <code>prepData</code> . Ignored unless <code>miData</code> is a <code>crwData</code> object.
<code>angleCovs</code>	Character vector indicating the names of any circular-circular regression angular covariates in <code>miData\$crwPredict</code> that need conversion from standard direction (in radians relative to the x-axis) to turning angle (relative to previous movement direction) See <code>prepData</code> . Ignored unless <code>miData</code> is a <code>crwData</code> or <code>crwHierData</code> object.
<code>altCoordNames</code>	Character string indicating an alternative name for the returned location data. See <code>prepData</code> . Ignored unless <code>miData</code> is a <code>crwData</code> or <code>crwHierData</code> object.
<code>method</code>	Method for obtaining weights for movement parameter samples. See <code>crwSimulator</code> . Ignored unless <code>miData</code> is a <code>crwData</code> object.
<code>parIS</code>	Size of the parameter importance sample. See <code>crwSimulator</code> . Ignored unless <code>miData</code> is a <code>crwData</code> object.
<code>dfSim</code>	Degrees of freedom for the t approximation to the parameter posterior. See ‘df’ argument in <code>crwSimulator</code> . Ignored unless <code>miData</code> is a <code>crwData</code> object.

grid.eps	Grid size for method="quadrature". See <a href="#">crwSimulator</a> . Ignored unless miData is a <a href="#">crwData</a> object.
crit	Criterion for deciding "significance" of quadrature points (difference in log-likelihood). See <a href="#">crwSimulator</a> . Ignored unless miData is a <a href="#">crwData</a> object.
scaleSim	Scale multiplier for the covariance matrix of the t approximation. See 'scale' argument in <a href="#">crwSimulator</a> . Ignored unless miData is a <a href="#">crwData</a> object.
quad.ask	Logical, for method='quadrature'. Whether or not the sampler should ask if quadrature sampling should take place. It is used to stop the sampling if the number of likelihood evaluations would be extreme. Default: FALSE. Ignored if ncores>1.
force.quad	A logical indicating whether or not to force the execution of the quadrature method for large parameter vectors. See <a href="#">crwSimulator</a> . Default: TRUE. Ignored unless miData is a <a href="#">crwData</a> object and method='`quadrature`'.
fullPost	Logical indicating whether to draw parameter values as well to simulate full posterior. See <a href="#">crwPostIS</a> . Ignored unless miData is a <a href="#">crwData</a> object.
dfPostIS	Degrees of freedom for multivariate t distribution approximation to parameter posterior. See 'df' argument in <a href="#">crwPostIS</a> . Ignored unless miData is a <a href="#">crwData</a> object.
scalePostIS	Extra scaling factor for t distribution approximation. See 'scale' argument in <a href="#">crwPostIS</a> . Ignored unless miData is a <a href="#">crwData</a> object.
thetaSamp	If multiple parameter samples are available in <a href="#">crwSimulator</a> objects, setting thetaSamp=n will use the nth sample. Defaults to the last. See <a href="#">crwSimulator</a> and <a href="#">crwPostIS</a> . Ignored unless miData is a <a href="#">crwData</a> object.
hierStates	A hierarchical model structure <a href="#">Node</a> for the states. See <a href="#">fithMM</a> .
hierDist	A hierarchical data structure <a href="#">Node</a> for the data streams. See <a href="#">fithMM</a> .
hierBeta	A hierarchical data structure <a href="#">Node</a> for the matrix of initial values for the regression coefficients of the transition probabilities at each level of the hierarchy ('beta'). See <a href="#">fithMM</a> .
hierDelta	A hierarchical data structure <a href="#">Node</a> for the matrix of initial values for the regression coefficients of the initial distribution at each level of the hierarchy ('delta'). See <a href="#">fithMM</a> .
hierFormula	A hierarchical formula structure for the transition probability covariates for each level of the hierarchy. See <a href="#">fithMM</a> .
hierFormulaDelta	A hierarchical formula structure for the initial distribution covariates for each level of the hierarchy ('formulaDelta'). Default: NULL (no covariate effects and fixPar\$delta is specified on the working scale). See <a href="#">fithMM</a> .

## Details

miData can either be a [crwData](#) or [crwHierData](#) object (as returned by [crawlWrap](#)), a [crwSim](#) or [crwHierSim](#) object (as returned by MIfitHMM when fit=FALSE), or a list of [momentuHMMData](#) or [momentuHierHMMData](#) objects (e.g., each element of the list as returned by [prepData](#)).

If miData is a [crwData](#) (or [crwHierData](#)) object, MIfitHMM uses a combination of [crwSimulator](#), [crwPostIS](#), [prepData](#), and [fithMM](#) to draw nSims realizations of the position process and fit the

specified HMM to each imputation of the data. The vast majority of MIfitHMM arguments are identical to the corresponding arguments from these functions.

If `miData` is a `crwData` or `crwHierData` object, `nSims` determines both the number of realizations of the position process to draw (using `crwSimulator` and `crwPostIS`) as well as the number of HMM fits.

If `miData` is a `crwSim` (or `crwHierSim`) object or a list of `momentuHMMData` (or `momentuHierHMMData`) object(s), the specified HMM will simply be fitted to each of the `momentuHMMData` (or `momentuHierHMMData`) objects and all arguments related to `crwSimulator`, `crwPostIS`, or `prepData` are ignored.

## Value

If `nSims`>1, `poolEstimates`=TRUE, and `fit`=TRUE, a `miHMM` object, i.e., a list consisting of:

`miSum`                    `miSum` object returned by `MIPool`.  
`HMMfits`                List of length `nSims` comprised of `momentuHMM` objects.

If `poolEstimates`=FALSE and `fit`=TRUE, a list of length `nSims` consisting of `momentuHMM` objects is returned.

However, if `fit`=FALSE and `miData` is a `crwData` object, then MIfitHMM returns a `crwSim` object, i.e., a list containing `miData` (a list of `momentuHMMData` objects) and `crwSimulator` (a list of `crwSimulator` objects), and most other arguments related to `fitHMM` are ignored.

## References

Hooten M.B., Johnson D.S., McClintock B.T., Morales J.M. 2017. Animal Movement: Statistical Models for Telemetry Data. CRC Press, Boca Raton.

McClintock B.T. 2017. Incorporating telemetry error into hidden Markov movement models using multiple imputation. Journal of Agricultural, Biological, and Environmental Statistics.

## See Also

`crawlWrap`, `crwPostIS`, `crwSimulator`, `fitHMM`, `getParDM`, `MIPool`, `prepData`

## Examples

```
# Don't run because it takes too long on a single core
## Not run:
# extract simulated obsData from example data
obsData <- miExample$obsData

# error ellipse model
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)

# create crwData object by fitting crwMLE models to obsData and predict locations
# at default intervals for both individuals
crwOut <- crawlWrap(obsData=obsData,
  theta=c(4,0), fixPar=c(1,1,NA,NA),
  err.model=err.model)
```

```

# HMM specifications
nbStates <- 2
stepDist <- "gamma"
angleDist <- "vm"
mu0 <- c(20,70)
sigma0 <- c(10,30)
kappa0 <- c(1,1)
stepPar0 <- c(mu0,sigma0)
anglePar0 <- c(-pi/2,pi/2,kappa0)
formula <- ~cov1+cos(cov2)
nbCovs <- 2
beta0 <- matrix(c(rep(-1.5,nbStates*(nbStates-1)),rep(0,nbStates*(nbStates-1)*nbCovs)),
                nrow=nbCovs+1,byrow=TRUE)

# first fit HMM to best predicted position process
bestData<-prepData(crwOut,covNames=c("cov1","cov2"))
bestFit<-fitHMM(bestData,
                nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
                Par0=list(step=stepPar0,angle=anglePar0),beta0=beta0,
                formula=formula,estAngleMean=list(angle=TRUE))

print(bestFit)

# extract estimates from 'bestFit'
bPar0 <- getPar(bestFit)

# Fit nSims=5 imputations of the position process
miFits<-MIfitHMM(miData=crwOut,nSims=5,
                 nbStates=nbStates,dist=list(step=stepDist,angle=angleDist),
                 Par0=bPar0$Par,beta0=bPar0$beta,delta0=bPar0$delta,
                 formula=formula,estAngleMean=list(angle=TRUE),
                 covNames=c("cov1","cov2"))

# print pooled estimates
print(miFits)

## End(Not run)

```

---

miHMM

*Constructor of miHMM objects*


---

## Description

Constructor of miHMM objects

## Usage

```
miHMM(m)
```

**Arguments**

`m` A list with attributes `miSum` (a `miSum` object) and `HMMfits` (a list of `momentuHMM` objects). `miHMM` objects are returned by `MIfitHMM` when arguments `fit=TRUE`, `nSims>1`, and `poolEstimates=TRUE`.

**Value**

An object `miHMM`.

---

<code>MIpool</code>	<i>Calculate pooled parameter estimates and states across multiple imputations</i>
---------------------	--

---

**Description**

Calculate pooled parameter estimates and states across multiple imputations

**Usage**

```
MIpool(im, alpha = 0.95, ncores = 1, covs = NULL, na.rm = FALSE)
```

**Arguments**

`im` List comprised of `momentuHMM` or `momentuHierHMM` objects

`alpha` Significance level for calculating confidence intervals of pooled estimates (including location error ellipses). Default: 0.95.

`ncores` Number of cores to use for parallel processing. Default: 1 (no parallel processing).

`covs` Data frame consisting of a single row indicating the covariate values to be used in the calculation of pooled natural parameters. For any covariates that are not specified using `covs`, the means of the covariate(s) across the imputations are used (unless the covariate is a factor, in which case the first factor in the data is used). By default, no covariates are specified.

`na.rm` Logical indicating whether or not to exclude model fits with NA parameter estimates or standard errors from pooling. Default: FALSE.

**Details**

Pooled estimates, standard errors, and confidence intervals are calculated using standard multiple imputation formulas. Working scale parameters are pooled using `MIcombine` and t-distributed confidence intervals. Natural scale parameters and normally-distributed confidence intervals are calculated by transforming the pooled working scale parameters and, if applicable, are based on covariate means across all imputations (and/or values specified in `covs`).

The calculation of pooled error ellipses uses `dataEllipse` from the `car` package. The suggested package `car` is not automatically imported by `momentuHMM` and must be installed in order to calculate error ellipses. A warning will be triggered if the `car` package is required but not installed.

Note that pooled estimates for `timeInStates` and `stateProbs` do not include within-model uncertainty and are based entirely on across-model variability.

## Value

A `miSum` object, i.e., a list comprised of model and pooled parameter summaries, including data (averaged across imputations), conditions, `Par`, and `MIcombine` (as returned by `MIcombine` for working parameters).

`miSum$Par` is a list comprised of:

<code>beta</code>	Pooled estimates for the working parameters
<code>real</code>	Estimates for the natural parameters based on pooled working parameters and covariate means (or covs) across imputations (if applicable)
<code>timeInStates</code>	The proportion of time steps assigned to each state
<code>states</code>	The most frequent state assignment for each time step based on the <code>viterbi</code> algorithm for each model fit
<code>stateProbs</code>	Pooled state probability estimates for each time step
<code>mixtureProbs</code>	Pooled mixture probabilities for each individual (only applies if <code>mixtures&gt;1</code> )
<code>hierStateProbs</code>	Pooled state probability estimates for each time step at each level of the hierarchy (only applies if <code>im</code> is comprised of <code>momentuHierHMM</code> objects)

## Examples

```
## Not run:
# Extract data and crawl inputs from miExample
obsData <- miExample$obsData

# error ellipse model
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)

# Fit crawl to obsData
crwOut <- crawlWrap(obsData, theta=c(4,0), fixPar=c(1,1,NA,NA),
                    err.model=err.model)

# Fit four imputations
bPar <- miExample$bPar
HMMfits <- MIfitHMM(crwOut, nSims=4, poolEstimates=FALSE,
                    nbStates=2, dist=list(step="gamma", angle="vm"),
                    Par0=bPar$Par, beta0=bPar$beta,
                    formula=~cov1+cos(cov2),
                    estAngleMean=list(angle=TRUE),
                    covNames=c("cov1", "cov2"))

# Pool estimates
miSum <- MIpool(HMMfits)
```

```
print(miSum)
## End(Not run)
```

---

miSum	<i>Constructor of miSum objects</i>
-------	-------------------------------------

---

### Description

Constructor of miSum objects

### Usage

```
miSum(m)
```

### Arguments

m	A list of attributes required for multiple imputation summaries: data (averaged across imputations), Par (the pooled estimates of the parameters of the model), conditions (conditions used to fit the model), and MIcombine (as returned by <a href="#">MIcombine</a> for the working parameters).
---	---

### Value

An object miSum.

---

mixtureProbs	<i>Mixture probabilities</i>
--------------	------------------------------

---

### Description

For a fitted model, this function computes the probability of each individual being in a particular mixture

### Usage

```
mixtureProbs(m, getCI = FALSE, alpha = 0.95)
```

### Arguments

m	<a href="#">momentuHMM</a> or <a href="#">momentuHierHMM</a> object
getCI	Logical indicating whether to calculate standard errors and logit-transformed confidence intervals for fitted <a href="#">momentuHMM</a> or <a href="#">momentuHierHMM</a> object. Default: FALSE.
alpha	Significance level of the confidence intervals (if getCI=TRUE). Default: 0.95 (i.e. 95% CIs).

**Details**

When `getCI=TRUE`, it can take a while for large data sets and/or a large number of mixtures because the model likelihood for each individual must be repeatedly evaluated in order to numerically approximate the SEs.

**Value**

The matrix of individual mixture probabilities, with element `[i,j]` the probability of individual `i` being in mixture `j`

**References**

Maruotti, A., and T. Ryden. 2009. A semiparametric approach to hidden Markov models under longitudinal observations. *Statistics and Computing* 19: 381-393.

**Examples**

```
## Not run:
nObs <- 100
nbAnimals <- 20
dist <- list(step="gamma",angle="vm")
Par <- list(step=c(100,1000,50,100),angle=c(0,0,0.1,2))

# create sex covariate
cov <- data.frame(sex=factor(rep(c("F","M"),each=nObs*nbAnimals/2)))
formulaPi <- ~ sex + 0

# Females more likely in mixture 1, males more likely in mixture 2
beta <- list(beta=matrix(c(-1.5,-0.5,-1.5,-3),2,2),
             pi=matrix(c(-2,2),2,1,dimnames=list(c("sexF","sexM"),"mix2")))

data.mix<-simData(nbAnimals=nbAnimals,obsPerAnimal=nObs,nbStates=2,dist=dist,Par=Par,
                 beta=beta,formulaPi=formulaPi,mixtures=2,covs=cov)

Par0 <- list(step=Par$step, angle=Par$angle[3:4])
m.mix <- fithMM(data.mix, nbStates=2, dist=dist, Par0 = Par0,
               beta0=beta,formulaPi=formulaPi,mixtures=2)

mixProbs <- mixtureProbs(m.mix, getCI=TRUE)

## End(Not run)
```

---

momentuHierHMM

*Constructor of momentuHierHMM objects*


---

**Description**

Constructor of momentuHierHMM objects



**Usage**

```
momentuHierHMM(m)
```

**Arguments**

**m** A list of attributes of the fitted model: `mle` (the maximum likelihood estimates of the parameters of the model), `data` (the `fitHMM` data), `mod` (the object returned by the `fitHMM` numerical optimizer `nlm` or `optim`), `conditions` (conditions used to fit the model: `hierStates`, `hierDist`, `zeroInflation`, `estAngleMean`, `circularAngleMean` `stationary`, `formula`, `userBounds`, `bounds`, `workBounds`, `DM`, etc.), `stateNames`, and `rawCovs` (optional – only if there are transition probability matrix covariates in the data).

**Value**

An object `momentuHierHMM`.

---

`momentuHierHMMData`     *Constructor of momentuHierHMMData objects*

---

**Description**

Constructor of `momentuHierHMMData` objects

**Usage**

```
momentuHierHMMData(data)
```

**Arguments**

**data** A dataframe containing: `ID` (the ID(s) of the observed animal(s)), `level` (the level of the hierarchy for each observation), and the data streams such as `step` (the step lengths, if any), `angle` (the turning angles, if any), `x` (either easting or longitude, if any), `y` (either northing or latitude, if any), and `covariates` (if any).

**Value**

An object `momentuHierHMMData`.

---

momentuHMM	<i>Constructor of momentuHMM objects</i>
------------	--

---

**Description**

Constructor of momentuHMM objects

**Usage**

```
momentuHMM(m)
```

**Arguments**

m	A list of attributes of the fitted model: mle (the maximum likelihood estimates of the parameters of the model), data (the fitHMM data), mod (the object returned by the fitHMM numerical optimizer nlm or optim), conditions (conditions used to fit the model: dist, zeroInflation, estAngleMean, circularAngleMean stationary, formula, userBounds, bounds, workBounds, DM, etc.), stateNames, and rawCovs (optional – only if there are transition probability matrix covariates in the data).
---	--

**Value**

An object momentuHMM.

---

momentuHMMData	<i>Constructor of momentuHMMData objects</i>
----------------	--

---

**Description**

Constructor of momentuHMMData objects

**Usage**

```
momentuHMMData(data)
```

**Arguments**

data	A dataframe containing: ID (the ID(s) of the observed animal(s)) and the data streams such as step (the step lengths, if any), angle (the turning angles, if any), x (either easting or longitude, if any), y (either northing or latitude, if any), and covariates (if any).
------	---

**Value**

An object momentuHMMData.

---

n2w                                      *Scaling function: natural to working parameters.*

---

### Description

Scales each data stream probability distribution parameter from its natural interval to the set of real numbers, to allow for unconstrained optimization. Used during the optimization of the log-likelihood. Parameters of any data streams for which a design matrix is specified are ignored.

### Usage

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

### Arguments

par	Named list of vectors containing the initial parameter values for each data stream.
bounds	Named list of 2-column matrices specifying bounds on the natural (i.e, real) scale of the probability distribution parameters for each data stream.
beta	List 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	Named list indicating whether or not to estimate the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy').
DM	An optional named list indicating the design matrices to be used for the probability distribution parameters of each data stream. Each element of DM can either be a named list of linear regression formulas or a matrix.
Bndind	Named list indicating whether DM is NULL with default parameter bounds for each data stream.
dist	A named list indicating the probability distributions of the data streams.

### Value

A vector of unconstrained parameters.

### Examples

```
## Not run:  
m<-example$m  
nbStates <- 2  
nbCovs <- 2  
parSize <- list(step=2,angle=2)  
par <- list(step=c(t(m$mle$step)),angle=c(t(m$mle$angle)))  
bounds <- m$conditions$bounds  
beta <- matrix(rnorm(6),ncol=2,nrow=3)  
delta <- c(0.6,0.4)
```

```

#working parameters
wpar <- momentuHMM:::w2n(par,bounds,list(beta=beta),log(delta[-1]/delta[1]),nbStates,
m$conditions$estAngleMean,NULL,m$conditions$Bndind,
m$conditions$dist)

#natural parameter
p <- momentuHMM:::w2n(wpar,bounds,parSize,nbStates,nbCovs,m$conditions$estAngleMean,
m$conditions$circularAngleMean,lapply(m$conditions$dist,function(x) x=="vmConsensus"),
m$conditions$stationary,m$conditions$fullDM,
m$conditions$DMind,1,m$conditions$dist,m$conditions$Bndind,
matrix(1,nrow=length(unique(m$data$ID)),ncol=1),covsDelta=m$covsDelta,
workBounds=m$conditions$workBounds)

## End(Not run)

```

---

nLogLike

*Negative log-likelihood function*


---

## Description

Negative log-likelihood function

## Usage

```

nLogLike(
  optPar,
  nbStates,
  formula,
  bounds,
  parSize,
  data,
  dist,
  covs,
  estAngleMean,
  circularAngleMean,
  consensus,
  zeroInflation,
  oneInflation,
  stationary = FALSE,
  fullDM,
  DMind,
  Bndind,
  knownStates,
  fixPar,
  wparIndex,
  nc,
  meanind,

```

```

    covsDelta,
    workBounds,
    prior = NULL,
    betaCons = NULL,
    betaRef,
    deltaCons = NULL,
    optInd = NULL,
    recovs = NULL,
    g0covs = NULL,
    mixtures = 1,
    covsPi,
    recharge = NULL,
    aInd
)

```

### Arguments

optPar	Vector of working parameters.
nbStates	Number of states of the HMM.
formula	Regression formula for the transition probability covariates.
bounds	Named list of 2-column matrices specifying bounds on the natural (i.e, real) scale of the probability distribution parameters for each data stream.
parSize	Named list indicating the number of natural parameters of the data stream probability distributions
data	An object momentuHMMData.
dist	Named list indicating the probability distributions of the data streams.
covs	data frame containing the beta model covariates (if any)
estAngleMean	Named list indicating whether or not to estimate the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy').
circularAngleMean	Named list indicating whether to use circular-linear or circular-circular regression on the mean of circular distributions ('vm' and 'wrpcauchy') for turning angles. See <a href="#">fitHMM</a> .
consensus	Named list indicating whether to use the circular-circular regression consensus model
zeroInflation	Named list of logicals indicating whether the probability distributions of the data streams are zero-inflated.
oneInflation	Named list of logicals indicating whether the probability distributions of the data streams are one-inflated.
stationary	FALSE if there are time-varying covariates in formula or any covariates in formulaDelta. If TRUE, the initial distribution is considered equal to the stationary distribution. Default: FALSE.
fullDM	Named list containing the full (i.e. not shorthand) design matrix for each data stream.

DMind	Named list indicating whether fullDM includes individual- and/or temporal-covariates for each data stream specifies (-1,1) bounds for the concentration parameters instead of the default [0,1) bounds.
Bndind	Named list indicating whether DM is NULL with default parameter bounds for each data stream.
knownStates	Vector of values of the state process which are known prior to fitting the model (if any).
fixPar	Vector of working parameters which are assumed known prior to fitting the model (NA indicates parameters is to be estimated).
wparIndex	Vector of indices for the elements of fixPar that are not NA.
nc	indicator for zeros in fullDM
meanind	index for circular-circular regression mean angles with at least one non-zero entry in fullDM
covsDelta	data frame containing the delta model covariates (if any)
workBounds	named list of 2-column matrices specifying bounds on the working scale of the probability distribution, transition probability, and initial distribution parameters
prior	A function that returns the log-density of the working scale parameter prior distribution(s)
betaCons	Matrix of the same dimension as beta $\theta$ composed of integers identifying any equality constraints among the t.p.m. parameters.
betaRef	Indices of reference elements for t.p.m. multinomial logit link.
deltaCons	Matrix of the same dimension as del $\theta$ composed of integers identifying any equality constraints among the initial distribution working scale parameters.
optInd	indices of constrained parameters
recovs	data frame containing the recharge model theta covariates (if any)
g0covs	data frame containing the recharge model g0 covariates (if any)
mixtures	Number of mixtures for the state transition probabilities
covsPi	data frame containing the pi model covariates
recharge	recharge model specification (only used for hierarchical models)
aInd	vector of indices of first observation for each animal

**Value**

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

**Examples**

```
## Not run:
# data is a momentuHMMData object (as returned by prepData), automatically loaded with the package
data <- example$m$data
m <- example$m
Par <- getPar(m)
nbStates <- length(m$stateNames)
```

```

inputs <- momentuHMM:::checkInputs(nbStates,m$conditions$dist,Par$Par,m$conditions$estAngleMean,
  m$conditions$circularAngleMean,m$conditions$zeroInflation,m$conditions$oneInflation,
  m$conditions$DM,m$conditions$userBounds,
  m$stateNames)

wpar <- momentuHMM:::n2w(Par$Par,m$conditions$bounds,list(beta=Par$beta),
  log(Par$delta[-1]/Par$delta[1]),nbStates,m$conditions$estAngleMean,
  m$conditions$DM,m$conditions$Bndind,
  m$conditions$dist)

l <- momentuHMM:::nLogLike(wpar,nbStates,m$conditions$formula,m$conditions$bounds,
  inputs$p$parSize,data,inputs$dist,model.matrix(m$conditions$formula,data),
  m$conditions$estAngleMean,m$conditions$circularAngleMean,inputs$consensus,
  m$conditions$zeroInflation,m$conditions$oneInflation,m$conditions$stationary,
  m$conditions$fullDM,m$conditions$DMind,
  m$conditions$Bndind,m$knownStates,unlist(m$conditions$fixPar),
  m$conditions$wparIndex,covsDelta=m$covsDelta,workBounds=m$conditions$workBounds,
  betaRef=m$conditions$betaRef,covsPi=m$covsPi)

## End(Not run)

```

---

nLogLike\_rcpp

*Negative log-likelihood*


---

## Description

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

## Usage

```

nLogLike_rcpp(
  nbStates,
  covs,
  data,
  dataNames,
  dist,
  Par,
  aInd,
  zeroInflation,
  oneInflation,
  stationary,
  knownStates,
  betaRef,
  mixtures
)

```

**Arguments**

nbStates	Number of states,
covs	Covariates,
data	A <code>momentuHMMData</code> object of the observations,
dataNames	Character vector containing the names of the data streams,
dist	Named list indicating the probability distributions of the data streams.
Par	Named list containing the state-dependent parameters of the data streams, matrix of regression coefficients for the transition probabilities ('beta'), and initial distribution ('delta').
aInd	Vector of indices of the rows at which the data switches to another animal
zeroInflation	Named list of logicals indicating whether the probability distributions of the data streams are zero-inflated.
oneInflation	Named list of logicals indicating whether the probability distributions of the data streams are one-inflated.
stationary	false if there are time-varying covariates in formula or any covariates in formulaDelta. 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.
betaRef	Indices of reference elements for t.p.m. multinomial logit link.
mixtures	Number of mixtures for the state transition probabilities

**Value**

Negative log-likelihood

---

parDef *Parameters definition*

---

**Description**

Parameters definition

**Usage**

```
parDef(
  dist,
  nbStates,
  estAngleMean,
  zeroInflation,
  oneInflation,
  DM,
  userBounds = NULL
)
```



**Arguments**

dist	Named list indicating the probability distributions of the data streams.
nbStates	Number of states of the HMM.
estAngleMean	Named list indicating whether or not to estimate the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy').
zeroInflation	Named list of logicals indicating whether the probability distributions of the data streams should be zero-inflated.
oneInflation	Named list of logicals indicating whether the probability distributions of the data streams are one-inflated.
DM	An optional named list indicating the design matrices to be used for the probability distribution parameters of each data stream. Each element of DM can either be a named list of linear regression formulas or a matrix.
userBounds	An optional named list of 2-column matrices specifying bounds on the natural (i.e, real) scale of the probability distribution parameters for each data stream. For example, for a 2-state model using the wrapped Cauchy ('wrpcauchy') distribution for a data stream named 'angle' with estAngleMean\$angle=TRUE), userBounds=list(angle=matrix(c(-pi, -pi, -1, -1, pi, pi, 1, 1), 4, 2)) specifies (-1,1) bounds for the concentration parameters instead of the default [0,1) bounds.

**Value**

A list of:

parSize	Named list indicating the number of natural parameters of the data stream probability distributions.
bounds	Named list of 2-column matrices specifying bounds on the natural (i.e, real) scale of the probability distribution parameters for each data stream.
parNames	Names of parameters of the probability distribution for each data stream.
Bndind	Named list indicating whether DM is NULL with default parameter bounds for each data stream.

**Examples**

```
## Not run:
pD<-momentuHMM:::parDef(list(step="gamma",angle="wrpcauchy"),
  nbStates=2,list(step=FALSE,angle=FALSE),list(step=FALSE,angle=FALSE),
  list(step=FALSE,angle=FALSE),NULL,NULL)

## End(Not run)
```

---

plot.crwData	<i>Plot</i> crwData
--------------	---------------------

---

### Description

Plot observed locations, error ellipses (if applicable), predicted locations, and prediction intervals from [crwData](#) or [crwHierData](#) object.

### Usage

```
## S3 method for class 'crwData'
plot(
  x,
  animals = NULL,
  compact = FALSE,
  ask = TRUE,
  plotEllipse = TRUE,
  crawlPlot = FALSE,
  ...
)
```

### Arguments

x	An object <a href="#">crwData</a> or <a href="#">crwHierData</a> (as returned by <a href="#">crawlWrap</a> ).
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). Ignored unless <a href="#">crwPredictPlot</a> =FALSE.
ask	If TRUE, the execution pauses between each plot.
plotEllipse	If TRUE (the default) then error ellipses are plotted (if applicable). Ignored unless <a href="#">crwPredictPlot</a> =FALSE.
crawlPlot	Logical indicating whether or not to create individual plots using <a href="#">crwPredictPlot</a> . See <a href="#">crwPredictPlot</a> for details.
...	Further arguments for passing to <a href="#">crwPredictPlot</a>

### Details

In order for error ellipses to be plotted, the names for the semi-major axis, semi-minor axis, and orientation in `x$crwPredict` must respectively be `error_semimajor_axis`, `error_semiminor_axis`, and `error_ellipse_orientation`.

If the [crwData](#) (or [crwHierData](#)) object was created using data generated by [simData](#) (or [simHierData](#)) or [simObsData](#), then the true locations (`mux,muy`) are also plotted.

### See Also

[crwPredictPlot](#)

**Examples**

```
## Not run:
# extract simulated obsData from example data
obsData <- miExample$obsData

# error ellipse model
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)

# create crwData object
crwOut <- crawlWrap(obsData=obsData,
                    theta=c(4,0),fixPar=c(1,1,NA,NA),
                    err.model=err.model)

plot(crwOut,compact=TRUE,ask=FALSE,plotEllipse=FALSE)

## End(Not run)
```

---

plot.miHMM

*Plot* miHMM

---

**Description**

For multiple imputation analyses, plot the pooled data stream densities over histograms of the data, probability distribution parameters and transition probabilities as functions of the covariates, and maps of the animals' tracks colored by the decoded states.

**Usage**

```
## S3 method for class 'miHMM'
plot(
  x,
  animals = NULL,
  covs = NULL,
  ask = TRUE,
  breaks = "Sturges",
  hist.ylim = NULL,
  sepAnimals = FALSE,
  sepStates = FALSE,
  col = NULL,
  cumul = TRUE,
  plotTracks = TRUE,
  plotCI = FALSE,
  alpha = 0.95,
  plotStationary = FALSE,
  plotEllipse = TRUE,
  ...
)
```

**Arguments**

x	Object miHMM (as returned by <a href="#">MFitHMM</a> )
animals	Vector of indices or IDs of animals for which information will be plotted. Default: NULL ; all animals are plotted.
covs	Data frame consisting of a single row indicating the covariate values to be used in plots. If none are specified, the means of any covariates appearing in the model are used (unless covariate is a factor, in which case the first factor appearing in the data is used).
ask	If TRUE, the execution pauses between each plot.
breaks	Histogram parameter. See <a href="#">hist</a> documentation.
hist.ylim	Parameter ylim for the step length histograms. See <a href="#">hist</a> documentation. Default: NULL ; the function sets default values.
sepAnimals	If TRUE, the data is split by individuals in the histograms. Default: FALSE.
sepStates	If TRUE, the data is split by states in the histograms. Default: FALSE.
col	Vector or colors for the states (one color per state).
cumul	If TRUE, the sum of weighted densities is plotted (default).
plotTracks	If TRUE, the Viterbi-decoded tracks are plotted (default).
plotCI	Logical indicating whether to include confidence intervals in natural parameter plots (default: FALSE)
alpha	Significance level of the confidence intervals (if plotCI=TRUE). Default: 0.95 (i.e. 95% CIs).
plotStationary	Logical indicating whether to plot the stationary state probabilities as a function of any covariates (default: FALSE)
plotEllipse	Logical indicating whether to plot error ellipses around imputed location means. Default: TRUE.
...	Additional arguments passed to <code>graphics::plot</code> and <code>graphics::hist</code> functions. These can currently include <code>asp</code> , <code>cex</code> , <code>cex.axis</code> , <code>cex.lab</code> , <code>cex.legend</code> , <code>cex.main</code> , <code>legend.pos</code> , and <code>lwd</code> . See <a href="#">par</a> . <code>legend.pos</code> can be a single keyword from the list “bottomright”, “bottom”, “bottomleft”, “left”, “topleft”, “top”, “topright”, “right”, and “center”. Note that <code>asp</code> and <code>cex</code> only apply to plots of animal tracks.

**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 each imputation). 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**

```
## Not run:
# Extract data from miExample
obsData <- miExample$obsData

# error ellipse model
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)

# Fit crawl to obsData
crwOut <- crawlWrap(obsData, theta=c(4,0), fixPar=c(1,1,NA,NA),
                    err.model=err.model)

# Fit four imputations
bPar <- miExample$bPar
HMMfits <- MIfitHMM(crwOut, nSims=4, poolEstimates=FALSE,
                   nbStates=2, dist=list(step="gamma", angle="vm"),
                   Par0=bPar$Par, beta0=bPar$beta,
                   formula=~cov1+cos(cov2),
                   estAngleMean=list(angle=TRUE),
                   covNames=c("cov1", "cov2"))

miHMM <- momentuHMM:::miHMM(list(miSum=MIpool(HMMfits), HMMfits=HMMfits))
plot(miHMM)

## End(Not run)
```

---

plot.miSum

*Plot miSum*


---

**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 'miSum'
plot(
  x,
  animals = NULL,
  covs = NULL,
  ask = TRUE,
  breaks = "Sturges",
  hist.ylim = NULL,
  sepAnimals = FALSE,
  sepStates = FALSE,
  col = NULL,
  cumul = TRUE,
```

```

    plotTracks = TRUE,
    plotCI = FALSE,
    alpha = 0.95,
    plotStationary = FALSE,
    plotEllipse = TRUE,
    ...
)

```

## Arguments

x	Object miSum (as return by <a href="#">MIpool</a> )
animals	Vector of indices or IDs of animals for which information will be plotted. Default: NULL ; all animals are plotted.
covs	Data frame consisting of a single row indicating the covariate values to be used in plots. If none are specified, the means of any covariates appearing in the model are used (unless covariate is a factor, in which case the first factor appearing in the data is used).
ask	If TRUE, the execution pauses between each plot.
breaks	Histogram parameter. See <a href="#">hist</a> documentation.
hist.ylim	Parameter ylim for the step length histograms. See <a href="#">hist</a> documentation. Default: NULL ; the function sets default values.
sepAnimals	If TRUE, the data is split by individuals in the histograms. Default: FALSE.
sepStates	If TRUE, the data is split by states in the histograms. Default: FALSE.
col	Vector or colors for the states (one color per state).
cumul	If TRUE, the sum of weighted densities is plotted (default).
plotTracks	If TRUE, the Viterbi-decoded tracks are plotted (default).
plotCI	Logical indicating whether to include confidence intervals in natural parameter plots (default: FALSE)
alpha	Significance level of the confidence intervals (if plotCI=TRUE). Default: 0.95 (i.e. 95% CIs).
plotStationary	Logical indicating whether to plot the stationary state probabilities as a function of any covariates (default: FALSE)
plotEllipse	Logical indicating whether to plot error ellipses around imputed location means. Default: TRUE.
...	Additional arguments passed to <code>graphics::plot</code> and <code>graphics::hist</code> functions. These can currently include <code>asp</code> , <code>cex</code> , <code>cex.axis</code> , <code>cex.lab</code> , <code>cex.legend</code> , <code>cex.main</code> , <code>legend.pos</code> , and <code>lwd</code> . See <a href="#">par</a> . <code>legend.pos</code> can be a single keyword from the list “bottomright”, “bottom”, “bottomleft”, “left”, “topleft”, “top”, “topright”, “right”, and “center”. Note that <code>asp</code> and <code>cex</code> only apply to plots of animal tracks.

## 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 each imputation). 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

```
## Not run:
# Extract data from miExample
obsData <- miExample$obsData

# error ellipse model
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)

# Fit crawl to obsData
crwOut <- crawlWrap(obsData, theta=c(4,0), fixPar=c(1,1,NA,NA),
                    err.model=err.model)

# Fit four imputations
bPar <- miExample$bPar
HMMfits <- MIfitHMM(crwOut, nSims=4, poolEstimates=FALSE,
                   nbStates=2, dist=list(step="gamma", angle="vm"),
                   Par0=bPar$Par, beta0=bPar$beta,
                   formula=~cov1+cos(cov2),
                   estAngleMean=list(angle=TRUE),
                   covNames=c("cov1", "cov2"))

# Pool estimates
miSum <- MIpool(HMMfits)
plot(miSum)

## End(Not run)
```

---

```
plot.momentuHMM
```

```
Plot momentuHMM
```

---

## 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 'momentuHMM'
plot(
  x,
```

```

animals = NULL,
covs = NULL,
ask = TRUE,
breaks = "Sturges",
hist.ylim = NULL,
sepAnimals = FALSE,
sepStates = FALSE,
col = NULL,
cumul = TRUE,
plotTracks = TRUE,
plotCI = FALSE,
alpha = 0.95,
plotStationary = FALSE,
...
)

```

### Arguments

<code>x</code>	Object <code>momentuHMM</code>
<code>animals</code>	Vector of indices or IDs of animals for which information will be plotted. Default: <code>NULL</code> ; all animals are plotted.
<code>covs</code>	Data frame consisting of a single row indicating the covariate values to be used in plots. If none are specified, the means of any covariates appearing in the model are used (unless <code>covariate</code> is a factor, in which case the first factor in the data is used).
<code>ask</code>	If <code>TRUE</code> , the execution pauses between each plot.
<code>breaks</code>	Histogram parameter. See <code>hist</code> documentation.
<code>hist.ylim</code>	An optional named list of vectors specifying <code>ylim=c(ymin,ymax)</code> for the data stream histograms. See <code>hist</code> documentation. Default: <code>NULL</code> ; the function sets default values for all data streams.
<code>sepAnimals</code>	If <code>TRUE</code> , the data is split by individuals in the histograms. Default: <code>FALSE</code> .
<code>sepStates</code>	If <code>TRUE</code> , the data is split by states in the histograms. Default: <code>FALSE</code> .
<code>col</code>	Vector or colors for the states (one color per state).
<code>cumul</code>	If <code>TRUE</code> , the sum of weighted densities is plotted (default).
<code>plotTracks</code>	If <code>TRUE</code> , the Viterbi-decoded tracks are plotted (default).
<code>plotCI</code>	Logical indicating whether to include confidence intervals in natural parameter plots (default: <code>FALSE</code> )
<code>alpha</code>	Significance level of the confidence intervals (if <code>plotCI=TRUE</code> ). Default: 0.95 (i.e. 95% CIs).
<code>plotStationary</code>	Logical indicating whether to plot the stationary state probabilities as a function of any covariates (default: <code>FALSE</code> ). Ignored unless covariate are included in formula.
<code>...</code>	Additional arguments passed to <code>graphics::plot</code> and <code>graphics::hist</code> functions. These can currently include <code>asp</code> , <code>cex</code> , <code>cex.axis</code> , <code>cex.lab</code> , <code>cex.legend</code> ,



cex.main, legend.pos, and lwd. See [par](#). legend.pos can be a single keyword from the list “bottomright”, “bottom”, “bottomleft”, “left”, “topleft”, “top”, “topright”, “right”, and “center”. Note that asp and cex only apply to plots of animal tracks.

### 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.

Confidence intervals for natural parameters are calculated from the working parameter point and covariance estimates using finite-difference approximations of the first derivative for the transformation (see [grad](#)). For example, if  $dN$  is the numerical approximation of the first derivative of the transformation  $N = \exp(x_1 * B_1 + x_2 * B_2)$  for covariates  $(x_1, x_2)$  and working parameters  $(B_1, B_2)$ , then  $\text{var}(N) = dN \%*\% \text{Sigma} \%*\% dN$ , where  $\text{Sigma} = \text{cov}(B_1, B_2)$ , and normal confidence intervals can be constructed as  $N +/- \text{qnorm}(1 - (1 - \alpha) / 2) * \text{se}(N)$ .

### Examples

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

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

---

plot.momentuHMMDData    *Plot momentuHMMDData or momentuHierHMMDData*

---

### Description

Plot momentuHMMDData or momentuHierHMMDData

### Usage

```
## S3 method for class 'momentuHMMDData'
plot(
  x,
  dataNames = c("step", "angle"),
  animals = NULL,
  compact = FALSE,
  ask = TRUE,
  breaks = "Sturges",
  ...
)
```

**Arguments**

x	An object <code>momentuHMMData</code> or <code>momentuHierHMMData</code>
dataNames	Names of the variables to plot. Default is <code>dataNames=c("step", "angle")</code> .
animals	Vector of indices or IDs of animals for which information will be plotted. Default: <code>NULL</code> ; all animals are plotted.
compact	<code>TRUE</code> for a compact plot (all individuals at once), <code>FALSE</code> otherwise (default – one individual at a time).
ask	If <code>TRUE</code> , the execution pauses between each plot.
breaks	Histogram parameter. See <code>hist</code> documentation.
...	Currently unused. For compatibility with generic method.

**Examples**

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

plot(data, dataNames=c("step", "angle", "cov1", "cov2"),
      compact=TRUE, breaks=20, ask=FALSE)
```

---

plotPR

*Plot pseudo-residuals*


---

**Description**

Plots time series, qq-plots (against the standard normal distribution) using `qqPlot`, and sample ACF functions of the pseudo-residuals for each data stream

**Usage**

```
plotPR(m, lag.max = NULL, ncores = 1)
```

**Arguments**

m	A <code>momentuHMM</code> , <code>momentuHierHMM</code> , <code>miHMM</code> , <code>HMMfits</code> , or <code>miSum</code> object.
lag.max	maximum lag at which to calculate the acf. See <code>acf</code> .
ncores	number of cores to use for parallel processing

**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  $+\text{Inf}$  (check Section 6.2 of reference for more information on the computation of pseudo-residuals).

- If some data streams are zero-inflated and/or one-inflated, 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).
- For multiple imputation analyses, if `m` is a `miHMM` object or a list of `momentuHMM` objects, then the pseudo-residuals are individually calculated and plotted for each model fit. Note that pseudo-residuals for `miSum` objects (as returned by `MIPool`) are based on pooled parameter estimates and the means of the data values across all imputations (and therefore may not be particularly meaningful).

## 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 momentuHMM 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 plots coordinates in longitude and latitude (not UTM), so if data coordinates are not provided in longitude and latitude, then the coordinate reference system must be provided using the `projargs` argument. This function 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,
  shape = 16,
  states = NULL,
  animals = NULL,
  ask = TRUE,
```

```

    return = FALSE,
    stateNames = NULL,
    projargs = NULL
  )

```

## Arguments

data	Data frame or <code>momentuHMMData</code> object, with necessary fields 'x' (longitudinal direction) and 'y' (latitudinal direction). A <code>momentuHMM</code> , <code>miHMM</code> , or <code>miSum</code> object is also permitted, from which the data will be extracted. If <code>states=NULL</code> and a <code>momentuHMM</code> , <code>miHMM</code> , or <code>miSum</code> object is provided, the decoded states are automatically plotted.
zoom	The zoom level, as defined for <code>get_map</code> . Integer value between 3 (continent) and 21 (building).
location	Location of the center of the map to be plotted (this must be in the same coordinate reference system as data).
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 <code>geom_point</code> .
size	Size argument for <code>geom_point</code> .
shape	Shape argument for <code>geom_point</code> . If <code>states</code> is provided, then shape must either be a scalar or a vector of length <code>length(unique(states))</code> . If <code>states=NULL</code> , then shape must either be a scalar or a vector consisting of a value for each individual to be plotted.
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 <code>ggplot</code> object (which can be edited and plotted manually). If FALSE, the function automatically plots the map (default).
stateNames	Optional character vector of length <code>max(states)</code> indicating state names. Ignored unless <code>states</code> is provided.
projargs	A character string of PROJ.4 projection arguments indicating the coordinate reference system for data and location coordinates (if not longitude and latitude). A <code>CRS</code> object is also permitted. If <code>projargs</code> is provided, the coordinates will be internally transformed to longitude and latitude for plotting.

## 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>

---

plotSpatialCov      *Plot observations on raster image*

---

## Description

Plot tracking data over a raster layer.

## Usage

```
plotSpatialCov(  
  data,  
  spatialCov,  
  segments = TRUE,  
  compact = TRUE,  
  col = NULL,  
  alpha = 1,  
  size = 1,  
  shape = 16,  
  states = NULL,  
  animals = NULL,  
  ask = TRUE,  
  return = FALSE,  
  stateNames = NULL  
)
```

## Arguments

data	Data frame or <a href="#">momentuHMMData</a> object, with necessary fields 'x' (longitudinal direction) and 'y' (latitudinal direction). A <a href="#">momentuHMM</a> , <a href="#">miHMM</a> , or <a href="#">miSum</a> object is also permitted, from which the data will be extracted. If states=NULL and a <a href="#">momentuHMM</a> , <a href="#">miHMM</a> , or <a href="#">miSum</a> object is provided, the decoded states are automatically plotted.
spatialCov	<a href="#">raster</a> object of the RasterLayer class on which to plot the location data
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 <a href="#">geom_point</a> .
size	Size argument for <a href="#">geom_point</a> .

shape	Shape argument for <code>geom_point</code> . If <code>states</code> is provided, then shape must either be a scalar or a vector of length <code>length(unique(states))</code> . If <code>states=NULL</code> , then shape must either be a scalar or a vector consisting of a value for each individual to be plotted.
states	A sequence of integers, corresponding to the decoded states for these data. If specified, the observations are colored by states.
animals	Vector of indices or IDs of animals/tracks to be plotted. Default: <code>NULL</code> ; all animals are plotted.
ask	If <code>TRUE</code> , the execution pauses between each plot.
return	If <code>TRUE</code> , the function returns a <code>ggplot</code> object (which can be edited and plotted manually). If <code>FALSE</code> , the function automatically plots the map (default).
stateNames	Optional character vector of length <code>max(states)</code> indicating state names. Ignored unless <code>states</code> is provided.

### Examples

```
## Not run:
stepDist <- "gamma"
angleDist <- "vm"

# plot simulated data over forest raster automatically loaded with the package
spatialCov<-list(forest=forest)
data <- simData(nbAnimals=2,nbStates=2,dist=list(step=stepDist,angle=angleDist),
               Par=list(step=c(100,1000,50,100),angle=c(0,0,0.1,5)),
               beta=matrix(c(5,-10,-25,50),nrow=2,ncol=2,byrow=TRUE),
               formula=~forest,spatialCovs=spatialCov,
               obsPerAnimal=225,states=TRUE)

plotSpatialCov(data,forest,states=data$states)

## End(Not run)
```

---

plotStates

*Plot states*

---

### Description

Plot the states and states probabilities.

### Usage

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

### Arguments

m	A <code>momentuHMM</code> , <code>momentuHierHMM</code> , <code>miHMM</code> , or <code>miSum</code> object
animals	Vector of indices or IDs of animals for which states will be plotted.
ask	If <code>TRUE</code> , the execution pauses between each plot.

**Examples**

```
# m is a momentuHMM 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	<i>Plot stationary state probabilities</i>
----------------	--

---

**Description**

Plot stationary state probabilities

**Usage**

```
plotStationary(
  model,
  covs = NULL,
  col = NULL,
  plotCI = FALSE,
  alpha = 0.95,
  return = FALSE,
  ...
)
```

**Arguments**

model	momentuHMM, momentuHierHMM, miHMM, or miSum object
covs	Optional data frame consisting of a single row indicating the covariate values to be used in plots. If none are specified, the means of any covariates appearing in the model are used (unless covariate is a factor, in which case the first factor in the data is used).
col	Vector or colors for the states (one color per state).
plotCI	Logical indicating whether to include confidence intervals in plots (default: FALSE)
alpha	Significance level of the confidence intervals (if plotCI=TRUE). Default: 0.95 (i.e. 95% CIs).
return	Logical indicating whether to return a list containing estimates, SEs, CIs, and covariate values used to create the plots for each mixture and state. Ignored if plotCI=FALSE. Default: FALSE.
...	Additional arguments passed to graphics::plot. These can currently include cex.axis, cex.lab, cex.legend, cex.main, legend.pos, and lwd. See <a href="#">par</a> . legend.pos can be a single keyword from the list “bottomright”, “bottom”, “bottomleft”, “left”, “topleft”, “top”, “topright”, “right”, and “center”.

**Examples**

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

plotStationary(m)
```

---

```
prepData
```

---

*Preprocessing of the data streams and covariates*

---

**Description**

Preprocessing of the data streams, including calculation of step length, turning angle, and covariates from location data to be suitable for analysis using [fitHMM](#).

**Usage**

```
prepData(data, ...)

## Default S3 method:
prepData(
  data,
  type = c("UTM", "LL"),
  coordNames = c("x", "y"),
  covNames = NULL,
  spatialCovs = NULL,
  centers = NULL,
  centroids = NULL,
  angleCovs = NULL,
  altCoordNames = NULL,
  ...
)

## S3 method for class 'hierarchical'
prepData(
  data,
  type = c("UTM", "LL"),
  coordNames = c("x", "y"),
  covNames = NULL,
  spatialCovs = NULL,
  centers = NULL,
  centroids = NULL,
  angleCovs = NULL,
  altCoordNames = NULL,
  hierLevels,
  coordLevel,
  ...
)
```



**Arguments**

data	Either a data frame of data streams or a <code>crwData</code> (or <code>crwHierData</code> ) object (as returned by <code>crawlWrap</code> ). If data is a data frame, it can optionally include a field ID (identifiers for the observed individuals), coordinates from which step length ('step') and turning angle ('angle') are calculated, and any covariates (with names matching <code>covNames</code> and/or <code>angleCovs</code> ). If step length and turning angle are to be calculated from coordinates, the <code>coordNames</code> argument must identify the names for the x- (longitudinal) and y- (latitudinal) coordinates, and, for hierarchical data, the <code>coordLevel</code> argument must identify the level of the hierarchy at which the location data are obtained. With the exception of ID, <code>coordNames</code> , and, for hierarchical data, <code>level</code> , all variables in data are treated as data streams unless identified as covariates in <code>covNames</code> and/or <code>angleCovs</code> .
...	further arguments passed to or from other methods
type	'UTM' if easting/northing provided (the default), 'LL' if longitude/latitude. If type='LL' then step lengths are calculated in kilometers and turning angles are based on initial bearings (see <code>turnAngle</code> ). Ignored if data is a <code>crwData</code> object.
coordNames	Names of the columns of coordinates in the data data frame. Default: <code>c("x", "y")</code> . If <code>coordNames=NULL</code> then step lengths, turning angles, and location covariates (i.e., those specified by <code>spatialCovs</code> , <code>centers</code> , and <code>angleCovs</code> ) are not calculated. Ignored if data is a <code>crwData</code> object.
covNames	Character vector indicating the names of any covariates in data dataframe. Any variables in data (other than ID) that are not identified in <code>covNames</code> and/or <code>angleCovs</code> are assumed to be data streams (i.e., missing values will not be accounted for).
spatialCovs	List of <code>raster</code> objects for spatio-temporally referenced covariates. Covariates specified by <code>spatialCovs</code> are extracted from the raster layer(s) based on the location data (and the z values for a raster <code>stack</code> or <code>brick</code> ) for each time step. If an element of <code>spatialCovs</code> is a raster <code>stack</code> or <code>brick</code> , then z values must be set using <code>raster::setZ</code> and data must include column(s) of the corresponding z value(s) for each observation (e.g., 'time').
centers	2-column matrix providing the x-coordinates (column 1) and y-coordinates (column 2) for any activity centers (e.g., potential centers of attraction or repulsion) from which distance and angle covariates will be calculated based on the location data. If no row names are provided, then generic names are generated for the distance and angle covariates (e.g., 'center1.dist', 'center1.angle', 'center2.dist', 'center2.angle'); otherwise the covariate names are derived from the row names of <code>centers</code> as <code>paste0(rep(rownames(centers), each=2), c(".dist", ".angle"))</code> . As with covariates identified in <code>angleCovs</code> , note that the angle covariates for each activity center are calculated relative to the previous movement direction (instead of standard direction relative to the x-axis); this is to allow the mean turning angle to be modelled as a function of these covariates using circular-circular regression in <code>fitHMM</code> or <code>MIfitHMM</code> .
centroids	List where each element is a data frame containing the x-coordinates ('x'), y-coordinates ('y'), and times (with user-specified name, e.g., 'time') for centroids (i.e., dynamic activity centers where the coordinates can change over time) from which distance and angle covariates will be calculated based on the location

data. If any centroids are specified, then data must include a column indicating the time of each observation, and this column name must match the corresponding user-specified name of the time column in centroids (e.g. 'time'). Times can be numeric or POSIXt. If no list names are provided, then generic names are generated for the distance and angle covariates (e.g., 'centroid1.dist', 'centroid1.angle', 'centroid2.dist', 'centroid2.angle'); otherwise the covariate names are derived from the list names of centroids as `paste0(rep(names(centroids), each=2), c(".dist", ".angle"))`. As with covariates identified in `angleCovs`, note that the angle covariates for each centroid are calculated relative to the previous movement direction (instead of standard direction relative to the x-axis); this is to allow the mean turning angle to be modelled as a function of these covariates using circular-circular regression in `fitHMM` or `MifitHMM`.

<code>angleCovs</code>	Character vector indicating the names of any circular-circular regression angular covariates in data or <code>spatialCovs</code> that need conversion from standard direction (in radians relative to the x-axis) to turning angle (relative to previous movement direction) using <code>circAngles</code> .
<code>altCoordNames</code>	Character string indicating an alternative name for the returned location data. If provided, then <code>prepData</code> will return easting (or longitude) coordinate names as <code>paste0(altCoordNames, ".x")</code> and northing (or latitude) as <code>paste0(altCoordNames, ".y")</code> instead of x and y, respectively. This can be useful for location data that are intended to be modeled using a bivariate normal distribution (see <code>fitHMM</code> ). Ignored unless <code>coordNames</code> are provided.
<code>hierLevels</code>	Character vector indicating the levels of the hierarchy and their order, from top (coarsest scale) to bottom (finest scale), that are included in <code>data\$level</code> . For example, for a 2-level hierarchy then <code>hierLevels=c("1", "2i", "2")</code> indicates <code>data\$level</code> for each observation can be one of three factor levels: "1" (coarse scale), "2i" (initial fine scale), and "2" (fine scale). Ignored if data is a <code>crwHierData</code> object.
<code>coordLevel</code>	Character string indicating the level of the hierarchy for the location data. If specified, then data must include a 'level' field indicating the level of the hierarchy for each observation. Ignored if <code>coordNames</code> is NULL or data is a <code>crwHierData</code> object.

### Details

- If data is a `crwData` (or `crwHierData`) object, the `momentuHMMData` (or `momentuHierHMMData`) object created by `prepData` includes step lengths and turning angles calculated from the best predicted locations (i.e., `crwData$crwPredict$mu.x` and `crwData$crwPredict$mu.y`). Prior to using `prepData`, additional data streams or covariates unrelated to location (including z-values associated with `spatialCovs` raster stacks or bricks) can be merged with the `crwData` (or `crwHierData`) object using `crawlMerge`.
- For hierarchical data, data must include a 'level' field indicating the level of the hierarchy for each observation, and, for location data identified by `coordNames`, the `coordLevel` argument must indicate the level of the hierarchy at which the location data are obtained.

### Value

An object `momentuHMMData` or `momentuHierHMMData`, i.e., a dataframe of:

ID	The ID(s) of the observed animal(s)
...	Data streams (e.g., 'step', 'angle', etc.)
x	Either easting or longitude (if coordNames is specified or data is a crwData object)
y	Either northing or latitude (if coordNames is specified or data is a crwData object)
...	Covariates (if any)

**See Also**

[crawlMerge](#), [crawlWrap](#), [crwData](#)

[crwHierData](#)

**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)
cov1 <- rnorm(10)

data <- data.frame(coord1=coord1, coord2=coord2, cov1=cov1)
d <- prepData(data, coordNames=c("coord1", "coord2"), covNames="cov1")

# include additional data stream named 'omega'
omega <- rbeta(10,1,1)
data <- data.frame(coord1=coord1, coord2=coord2, omega=omega, cov1=cov1)
d <- prepData(data, coordNames=c("coord1", "coord2"), covNames="cov1")

# include 'forest' example raster layer as covariate
data <- data.frame(coord1=coord1*1000, coord2=coord2*1000)
spatialCov <- list(forest=forest)
d <- prepData(data, coordNames=c("coord1", "coord2"), spatialCovs=spatialCov)

# include 2 activity centers
data <- data.frame(coord1=coord1, coord2=coord2, cov1=cov1)
d <- prepData(data, coordNames=c("coord1", "coord2"), covNames="cov1",
              centers=matrix(c(0,10,0,10), 2, 2, dimnames=list(c("c1", "c2"), NULL)))

# include centroid
data <- data.frame(coord1=coord1, coord2=coord2, cov1=cov1, time=1:10)
d <- prepData(data, coordNames=c("coord1", "coord2"), covNames="cov1",
              centroid=list(centroid=data.frame(x=coord1+rnorm(10),
                                                y=coord2+rnorm(10),
                                                time=1:10)))

# Include angle covariate that needs conversion to
# turning angle relative to previous movement direction
u <- rnorm(10) # horizontal component
v <- rnorm(10) # vertical component
cov2 <- atan2(v,u)
data <- data.frame(coord1=coord1, coord2=coord2, cov1=cov1, cov2=cov2)

```

```
d <- prepData(data, coordNames=c("coord1", "coord2"), covNames="cov1",
              angleCovs="cov2")
```

---

```
print.miHMM          Print miHMM
```

---

## Description

Print miHMM

## Usage

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

## Arguments

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

## Examples

```
## Not run:
# Extract data from miExample
obsData <- miExample$obsData

# error ellipse model
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)

# Fit crawl to obsData
crwOut <- crawlWrap(obsData, theta=c(4,0), fixPar=c(1,1,NA,NA),
                   err.model=err.model)

# Fit four imputations
bPar <- miExample$bPar
HMMfits <- MIfitHMM(crwOut, nSims=4, poolEstimates=FALSE,
                  nbStates=2, dist=list(step="gamma", angle="vm"),
                  Par0=bPar$Par, beta0=bPar$beta,
                  formula=~cov1+cos(cov2),
                  estAngleMean=list(angle=TRUE),
                  covNames=c("cov1", "cov2"))

miHMM <- momentuHMM:::miHMM(list(miSum=MIPool(HMMfits), HMMfits=HMMfits))
print(miHMM)

## End(Not run)
```

---

print.miSum	<i>Print</i> miSum
-------------	--------------------

---

## Description

Print miSum

## Usage

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

## Arguments

x	A miSum object.
...	Currently unused. For compatibility with generic method.

## Examples

```
## Not run:  
# Extract data from miExample  
obsData <- miExample$obsData  
  
# error ellipse model  
err.model <- list(x= ~ ln.sd.x - 1, y = ~ ln.sd.y - 1, rho = ~ error.corr)  
  
# Fit crawl to obsData  
crwOut <- crawlWrap(obsData, theta=c(4,0), fixPar=c(1,1,NA,NA),  
                    err.model=err.model)  
  
# Fit four imputations  
bPar <- miExample$bPar  
HMMfits <- MIfitHMM(crwOut, nSims=4, poolEstimates=FALSE,  
                   nbStates=2, dist=list(step="gamma", angle="vm"),  
                   Par0=bPar$Par, beta0=bPar$beta,  
                   formula=~cov1+cos(cov2),  
                   estAngleMean=list(angle=TRUE),  
                   covNames=c("cov1", "cov2"))  
  
# Pool estimates  
miSum <- MIpool(HMMfits)  
print(miSum)  
  
## End(Not run)
```

---

```
print.momentuHMM      Print momentuHMM
```

---

### Description

Print momentuHMM

### Usage

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

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

### Arguments

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

### Examples

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

print(m)
```

---

```
pseudoRes            Pseudo-residuals
```

---

### Description

The pseudo-residuals of momentuHMM models, as described in Zucchini and McDonad (2009).

### Usage

```
pseudoRes(m, ncores = 1)
```

### Arguments

`m`                    A `momentuHMM`, `miHMM`, `HMMfits`, or `miSum` object.  
`ncores`              number of cores to use for parallel processing

## 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  $+\text{Inf}$  (check Section 6.2 of reference for more information on the computation of pseudo-residuals).

A continuity adjustment (adapted from Harte 2017) is made for discrete probability distributions. When the data are near the boundary (e.g. 0 for “pois”; 0 and 1 for “bern”), then the pseudo residuals can be a poor indicator of lack of fit.

For multiple imputation analyses, if  $m$  is a `miHMM` object or a list of `momentuHMM` objects, then the pseudo-residuals are individually calculated for each model fit. Note that pseudo-residuals for `miSum` objects (as returned by `MIPool`) are based on pooled parameter estimates and the means of the data values across all imputations (and therefore may not be particularly meaningful).

## Value

If  $m$  is a `momentuHMM`, `miHMM`, or `miSum` object, a list of pseudo-residuals for each data stream (e.g., ‘stepRes’, ‘angleRes’) is returned. If  $m$  is a list of `momentuHMM` objects, then a list of length `length(m)` is returned where each element is a list of pseudo-residuals for each data stream.

## References

Harte, D. 2017. HiddenMarkov: Hidden Markov Models. R package version 1.8-8.

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

## Examples

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

---

randomEffects

*Random effects estimation*

---

## Description

Approximate individual-level random effects estimation for state transition probabilities based on Burnham & White (2002)

**Usage**

```
randomEffects(
  m,
  Xformula = ~1,
  alpha = 0.95,
  ncores = 1,
  nlmPar = list(),
  fit = TRUE,
  retryFits = 0,
  retrySD = NULL,
  optMethod = "nlm",
  control = list(),
  modelName = NULL,
  ...
)
```

**Arguments**

<code>m</code>	A <a href="#">momentuHMM</a> object.
<code>Xformula</code>	Formula for the design matrix of the random effects model. The default <code>Xformula=~1</code> specifies an intercept-only model with no additional individual covariate effects.
<code>alpha</code>	Significance level of the confidence intervals. Default: 0.95 (i.e. 95% CIs).
<code>ncores</code>	number of cores to use for parallel processing
<code>nlmPar</code>	List of parameters to pass to the optimization function <code>nlm</code> . See <a href="#">fithMM</a> .
<code>fit</code>	TRUE if the HMM should be re-fitted at the shrinkage estimates, FALSE otherwise.
<code>retryFits</code>	Non-negative integer indicating the number of times to attempt to iteratively fit the model using random perturbations of the current parameter estimates as the initial values for likelihood optimization. See <a href="#">fithMM</a> .
<code>retrySD</code>	An optional list of scalars or vectors indicating the standard deviation to use for normal perturbations of each working scale parameter when <code>retryFits&gt;0</code> . See <a href="#">fithMM</a> .
<code>optMethod</code>	The optimization method to be used. See <a href="#">fithMM</a> .
<code>control</code>	A list of control parameters to be passed to <code>optim</code> (ignored unless <code>optMethod="Nelder-Mead"</code> or <code>optMethod="SANN"</code> ).
<code>modelName</code>	An optional character string providing a name for the fitted model. See <a href="#">fithMM</a> .
<code>...</code>	further arguments passed to or from other methods. Not currently used.

**Value**

A `randomEffects` model similar to a [momentuHMM](#) object, but including the additional random effect components:

<code>varcomp</code>	A list of length <code>nbStates*(nbStates-1)</code> with each element containing the random effect mean coefficient(s) ( <code>mu</code> ), random effect variance ( <code>sigma</code> ), and logit-scale shrinkage estimates for the state transition probability parameters ( <code>ztilde</code> ).
----------------------	---



traceG            The trace of the projection matrix for each random effect.

## References

Burnham, K.P. and White, G.C. 2002. Evaluation of some random effects methodology applicable to bird ringing data. *Journal of Applied Statistics* 29: 245-264.

McClintock, B.T. 2021. Worth the effort? A practical examination of random effects in hidden Markov models for animal telemetry data. *Methods in Ecology and Evolution* doi: [10.1111/2041-210X.13619](https://doi.org/10.1111/2041-210X.13619).

## Examples

```
## Not run:
# simulated data with normal random effects
# and binary individual covariate

nbAnimals <- 5 # should be larger for random effects estimation
obsPerAnimal <- 110
indCov <- rbinom(nbAnimals,1,0.5) # individual covariate
betaCov <- c(-0.5,0.5) # covariate effects
mu <- c(-0.1,0.1) # mean for random effects
sigma <- c(0.2,0.4) # sigma for random effects
beta0 <- cbind(rnorm(nbAnimals,mu[1],sigma[1]),
              rnorm(nbAnimals,mu[2],sigma[2]))

reData <- simData(nbAnimals=nbAnimals,obsPerAnimal=obsPerAnimal,nbStates=2,
                 dist=list(step="gamma"),formula=~0+ID+indCov,
                 Par=list(step=c(1,10,1,2)),
                 beta=rbind(beta0,betaCov),
                 covs=data.frame(indCov=rep(indCov,each=obsPerAnimal)))

# fit null model
nullFit <- fitHMM(reData,nbStates=2,
                 dist=list(step="gamma"),
                 Par0=list(step=c(1,10,1,2)))

# fit covariate model
covFit <- fitHMM(reData,nbStates=2,
                 dist=list(step="gamma"),formula=~indCov,
                 Par0=list(step=c(1,10,1,2)),
                 beta0=rbind(mu,betaCov))

# fit fixed effects model
fixFit <- fitHMM(reData,nbStates=2,
                 dist=list(step="gamma"),formula=~0+ID,
                 Par0=list(step=c(1,10,1,2)),
                 beta0=beta0)

# fit random effect model
reFit <- randomEffects(fixFit)

# fit random effect model with individual covariate
```

```

reCovFit <- randomEffects(fixFit, Xformula=~indCov)

# compare by AICc
AIC(nullFit,covFit,fixFit,reFit,reCovFit, n=nrow(reData))

## End(Not run)

```

---

setModelName	<i>Set modelName for a momentuHMM, miHMM, HMMfits, or miSum object</i>
--------------	--

---

### Description

Set modelName for a momentuHMM, miHMM, HMMfits, or miSum object

### Usage

```
setModelName(model, modelName)
```

### Arguments

model            [momentuHMM](#), [miHMM](#), [HMMfits](#), or [miSum](#) object  
modelName        Character string providing a name for the model. See [fitHMM](#) and [MIfitHMM](#).

### Value

model object with new modelName field

### Examples

```

m <- example$m
mName <- setModelName(m, modelName="example")

```

---

setStateNames	<i>Set stateNames for a momentuHMM, miHMM, HMMfits, or miSum object</i>
---------------	---

---

### Description

Set stateNames for a momentuHMM, miHMM, HMMfits, or miSum object

### Usage

```
setStateNames(model, stateNames)
```

**Arguments**

model                    [momentuHMM](#), [miHMM](#), [HMMfits](#), or [miSum](#) object  
stateNames              Character string providing state names for the model. See [fiHMM](#) and [MIfiHMM](#).

**Value**

model object with new stateNames field

**Examples**

```
m <- example$m
mName <- setStateNames(m, stateNames=c("encamped", "exploratory"))
```

---

simData

*Simulation tool*


---

**Description**

Simulates data from a (multivariate) hidden Markov model. Movement data are assumed to be in Cartesian coordinates (not longitude/latitude) and can be generated with or without observation error attributable to temporal irregularity or location measurement error.

**Usage**

```
simData(
  nbAnimals = 1,
  nbStates = 2,
  dist,
  Par,
  beta = NULL,
  delta = NULL,
  formula = ~1,
  formulaDelta = NULL,
  mixtures = 1,
  formulaPi = NULL,
  covs = NULL,
  nbCovs = 0,
  spatialCovs = NULL,
  zeroInflation = NULL,
  oneInflation = NULL,
  circularAngleMean = NULL,
  centers = NULL,
  centroids = NULL,
  angleCovs = NULL,
  obsPerAnimal = c(500, 1500),
  initialPosition = c(0, 0),
```

```
DM = NULL,
userBounds = NULL,
workBounds = NULL,
betaRef = NULL,
mvnCoords = NULL,
stateNames = NULL,
model = NULL,
states = FALSE,
retrySims = 0,
lambda = NULL,
errorEllipse = NULL,
ncores = 1
)

simHierData(
  nbAnimals = 1,
  hierStates,
  hierDist,
  Par,
  hierBeta = NULL,
  hierDelta = NULL,
  hierFormula = NULL,
  hierFormulaDelta = NULL,
  mixtures = 1,
  formulaPi = NULL,
  covs = NULL,
  nbHierCovs = NULL,
  spatialCovs = NULL,
  zeroInflation = NULL,
  oneInflation = NULL,
  circularAngleMean = NULL,
  centers = NULL,
  centroids = NULL,
  angleCovs = NULL,
  obsPerLevel,
  initialPosition = c(0, 0),
  DM = NULL,
  userBounds = NULL,
  workBounds = NULL,
  mvnCoords = NULL,
  model = NULL,
  states = FALSE,
  retrySims = 0,
  lambda = NULL,
  errorEllipse = NULL,
  ncores = 1
)
```

**Arguments**

nbAnimals	Number of observed individuals to simulate.
nbStates	Number of behavioural states to simulate.
dist	A named list indicating the probability distributions of the data streams. Currently supported distributions are 'bern', 'beta', 'cat', 'exp', 'gamma', 'lnorm', 'logis', 'negbinom', 'norm', 'mvnorm2' (bivariate normal distribution), 'mvnorm3' (trivariate normal distribution), 'pois', 'rw_norm' (normal random walk), 'rw_mvnorm2' (bivariate normal random walk), 'rw_mvnorm3' (trivariate normal random walk), 'vm', 'vmConsensus', 'weibull', and 'wrpcauchy'. For example, <code>dist=list(step='gamma', angle='vm')</code> indicates 3 data streams ('step', 'angle', and 'dives') and their respective probability distributions ('gamma', 'vm', and 'pois').
Par	A named list containing vectors of initial state-dependent probability distribution parameters for each data stream specified in <code>dist</code> . The parameters should be in the order expected by the pdfs of <code>dist</code> , and any zero-mass and/or one-mass parameters should be the last (if both are present, then zero-mass parameters must precede one-mass parameters).  If DM is not specified for a given data stream, then Par is on the natural (i.e., real) scale of the parameters. However, if DM is specified for a given data stream, then Par must be on the working (i.e., beta) scale of the parameters, and the length of Par must match the number of columns in the design matrix. See details below.
beta	Matrix of regression parameters for the transition probabilities (more information in "Details").
delta	Initial value for the initial distribution of the HMM. Default: <code>rep(1/nbStates, nbStates)</code> . If <code>formulaDelta</code> includes a formula, then <code>delta</code> must be specified as a $k \times (nbStates-1)$ matrix, where $k$ is the number of covariates and the columns correspond to states 2:nbStates. See details below.
formula	Regression formula for the transition probability covariates. Default: <code>~1</code> (no covariate effect). In addition to allowing standard functions in R formulas (e.g., <code>cos(cov)</code> , <code>cov1*cov2</code> , <code>I(cov^2)</code> ), special functions include <code>cosinor(cov, period)</code> for modeling cyclical patterns, spline functions ( <code>bs</code> , <code>ns</code> , <code>bSpline</code> , <code>cSpline</code> , <code>iSpline</code> , and <code>mSpline</code> ), and state- or parameter-specific formulas (see details). Any formula terms that are not state- or parameter-specific are included on all of the transition probabilities.
formulaDelta	Regression formula for the initial distribution. Default: NULL (no covariate effects and <code>delta</code> is specified on the real scale). Standard functions in R formulas are allowed (e.g., <code>cos(cov)</code> , <code>cov1*cov2</code> , <code>I(cov^2)</code> ). When any formula is provided, then <code>delta</code> must be specified on the working scale.
mixtures	Number of mixtures for the state transition probabilities (i.e. discrete random effects *sensu* DeRuiter et al. 2017). Default: <code>mixtures=1</code> .
formulaPi	Regression formula for the mixture distribution probabilities. Default: NULL (no covariate effects; both <code>beta\$pi</code> and <code>fixPar\$pi</code> are specified on the real scale). Standard functions in R formulas are allowed (e.g., <code>cos(cov)</code> , <code>cov1*cov2</code> , <code>I(cov^2)</code> ). When any formula is provided, then both <code>beta\$pi</code> and <code>fixPar\$pi</code> are specified on the working scale. Note that only the covariate values corresponding to the first time step for each individual ID are used (i.e. time-varying covariates cannot be used for the mixture probabilities).

covs	Covariate values to include in the simulated data, as a dataframe. The names of any covariates specified by covs can be included in formula and/or DM. Covariates can also be simulated according to a standard normal distribution, by setting covs to NULL (the default), and specifying nbCovs>0.
nbCovs	Number of covariates to simulate (0 by default). Does not need to be specified if covs is specified. Simulated covariates are provided generic names (e.g., 'cov1' and 'cov2' for nbCovs=2) and can be included in formula and/or DM.
spatialCovs	List of <a href="#">raster</a> objects for spatio-temporally referenced covariates. Covariates specified by spatialCovs are extracted from the raster layer(s) based on any simulated location data (and the z values for a raster <a href="#">stack</a> or <a href="#">brick</a> ) for each time step. If an element of spatialCovs is a raster <a href="#">stack</a> or <a href="#">brick</a> , then z values must be set using raster::setZ and covs must include column(s) of the corresponding z value(s) for each observation (e.g., 'time'). The names of the raster layer(s) can be included in formula and/or DM. Note that simData usually takes longer to generate simulated data when spatialCovs is specified.
zeroInflation	A named list of logicals indicating whether the probability distributions of the data streams should be zero-inflated. If zeroInflation is TRUE for a given data stream, then values for the zero-mass parameters should be included in the corresponding element of Par.
oneInflation	A named list of logicals indicating whether the probability distributions of the data streams should be one-inflated. If oneInflation is TRUE for a given data stream, then values for the one-mass parameters should be included in the corresponding element of Par.
circularAngleMean	<p>An optional named list indicating whether to use circular-linear (FALSE) or circular-circular (TRUE) regression on the mean of circular distributions ('vm' and 'wrpcauchy') for turning angles. For example, circularAngleMean=list(angle=TRUE) indicates the angle mean is be estimated for 'angle' using circular-circular regression. Whenever circular-circular regression is used for an angular data stream, a corresponding design matrix (DM) must be specified for the data stream, and the previous movement direction (i.e., a turning angle of zero) is automatically used as the reference angle (i.e., the intercept). Default is NULL, which assumes circular-linear regression is used for any angular distributions. Any circularAngleMean elements corresponding to data streams that do not have angular distributions are ignored. circularAngleMean is also ignored for any 'vmConsensus' data streams (because the consensus model is a circular-circular regression model).</p> <p>Alternatively, circularAngleMean can be specified as a numeric scalar, where the value specifies the coefficient for the reference angle (i.e., directional persistence) term in the circular-circular regression model. For example, setting circularAngleMean to 0 specifies a circular-circular regression model with no directional persistence term (thus specifying a biased random walk instead of a biased correlated random walk). Setting circularAngleMean to 1 is equivalent to setting it to TRUE, i.e., a circular-circular regression model with a coefficient of 1 for the directional persistence reference angle.</p>
centers	2-column matrix providing the x-coordinates (column 1) and y-coordinates (column 2) for any activity centers (e.g., potential centers of attraction or repul-

sion) from which distance and angle covariates will be calculated based on the simulated location data. These distance and angle covariates can be included in formula and DM using the row names of centers. If no row names are provided, then generic names are generated for the distance and angle covariates (e.g., 'center1.dist', 'center1.angle', 'center2.dist', 'center2.angle'); otherwise the covariate names are derived from the row names of centers as `paste0(rep(rownames(centers), each=2), c(".dist", ".angle"))`. Note that the angle covariates for each activity center are calculated relative to the previous movement direction instead of standard directions relative to the x-axis; this is to allow turning angles to be simulated as a function of these covariates using circular-circular regression.

centroids	List where each element is a data frame consisting of at least <code>max(unlist(obsPerAnimal))</code> rows that provides the x-coordinates ('x') and y-coordinates ('y') for centroids (i.e., dynamic activity centers where the coordinates can change for each time step) from which distance and angle covariates will be calculated based on the simulated location data. These distance and angle covariates can be included in formula and DM using the names of centroids. If no list names are provided, then generic names are generated for the distance and angle covariates (e.g., 'centroid1.dist', 'centroid1.angle', 'centroid2.dist', 'centroid2.angle'); otherwise the covariate names are derived from the list names of centroids as <code>paste0(rep(names(centroids), each=2), c(".dist", ".angle"))</code> . Note that the angle covariates for each centroid are calculated relative to the previous movement direction instead of standard directions relative to the x-axis; this is to allow turning angles to be simulated as a function of these covariates using circular-circular regression.
angleCovs	Character vector indicating the names of any circular-circular regression angular covariates in covs or spatialCovs that need conversion from standard direction (in radians relative to the x-axis) to turning angle (relative to previous movement direction) using <code>circAngles</code> .
obsPerAnimal	Either 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 observations generated for each animal are uniformly picked from this interval. Alternatively, obsPerAnimal can be specified as a list of length nbAnimals with each element providing the number of observations (if single value) or the bounds (if vector of two values) for each individual. Default: <code>c(500, 1500)</code> .
initialPosition	2-vector providing the x- and y-coordinates of the initial position for all animals. Alternatively, initialPosition can be specified as a list of length nbAnimals with each element a 2-vector providing the x- and y-coordinates of the initial position for each individual. Default: <code>c(0, 0)</code> . If mvnCoord corresponds to a data stream with "mvnnorm3" or "rw_mvnorm3" probability distributions, then initialPosition must be composed of 3-vector(s) for the x-, y-, and z-coordinates.
DM	An optional named list indicating the design matrices to be used for the probability distribution parameters of each data stream. Each element of DM can either be a named list of regression formulas or a "pseudo" design matrix. For example, for a 2-state model using the gamma distribution for a data stream named 'step',

`DM=list(step=list(mean=~cov1, sd=~1))` specifies the mean parameters as a function of the covariate 'cov1' for each state. This model could equivalently be specified as a 4x6 "pseudo" design matrix using character strings for the covariate: `DM=list(step=matrix(c(1,0,0,0, 'cov1', 0,0,0,0,1,0,0,0, 'cov1', 0,0,0,0,1,0,0,0,0,1), where the 4 rows correspond to the state-dependent parameters (mean_1,mean_2,sd_1,sd_2) and the 6 columns correspond to the regression coefficients.`

Design matrices specified using formulas allow standard functions in R formulas (e.g., `cos(cov)`, `cov1*cov2`, `I(cov^2)`). Special formula functions include `cosinor(cov,period)` for modeling cyclical patterns, spline functions (`bs`, `ns`, `bSpline`, `cSpline`, `iSpline`, and `mSpline`), `angleFormula(cov, strength, by)` for the angle mean of circular-circular regression models, and state-specific formulas (see details). Any formula terms that are not state-specific are included on the parameters for all `nbStates` states.

<code>userBounds</code>	An optional named list of 2-column matrices specifying bounds on the natural (i.e., real) scale of the probability distribution parameters for each data stream. For example, for a 2-state model using the wrapped Cauchy ('wrpcauchy') distribution for a data stream named 'angle' with <code>estAngleMean\$angle=TRUE</code> , <code>userBounds=list(angle=matrix(c(-pi,-pi,-1,-1,pi,pi,1,1),4,2,dimnames=list(c("mean_1", "sd_1", "sd_2", "cov1"),1:4)))</code> specifies (-1,1) bounds for the concentration parameters instead of the default [0,1) bounds.
<code>workBounds</code>	An optional named list of 2-column matrices specifying bounds on the working scale of the probability distribution, transition probability, and initial distribution parameters. For each matrix, the first column pertains to the lower bound and the second column the upper bound. For data streams, each element of <code>workBounds</code> should be a k x 2 matrix with the same name of the corresponding element of <code>Par</code> , where k is the number of parameters. For transition probability parameters, the corresponding element of <code>workBounds</code> must be a k x 2 matrix named "beta", where <code>k=length(beta)</code> . For initial distribution parameters, the corresponding element of <code>workBounds</code> must be a k x 2 matrix named "delta", where <code>k=length(delta)</code> . <code>workBounds</code> is ignored for any given data stream unless <code>DM</code> is also specified.
<code>betaRef</code>	Numeric vector of length <code>nbStates</code> indicating the reference elements for the t.p.m. multinomial logit link. Default: <code>NULL</code> , in which case the diagonal elements of the t.p.m. are the reference. See <a href="#">fithMM</a> .
<code>mvnCoords</code>	Character string indicating the name of location data that are to be simulated using a multivariate normal distribution. For example, if <code>mu="rw_mvnorm2"</code> was included in <code>dist</code> and ( <code>mu.x</code> , <code>mu.y</code> ) are intended to be location data, then <code>mvnCoords="mu"</code> needs to be specified in order for these data to be treated as such.
<code>stateNames</code>	Optional character vector of length <code>nbStates</code> indicating state names.
<code>model</code>	A <a href="#">momentuHMM</a> , <a href="#">momentuHierHMM</a> , <a href="#">miHMM</a> , or <a href="#">miSum</a> object. This option can be used to simulate from a fitted model. Default: <code>NULL</code> . Note that, if this argument is specified, most other arguments will be ignored – except for <code>nbAnimals</code> , <code>obsPerAnimal</code> , <code>states</code> , <code>initialPosition</code> , <code>lambda</code> , <code>errorEllipse</code> , and, if covariate values different from those in the data should be specified, <code>covs</code> , <code>spatialCovs</code> , <code>centers</code> , and <code>centroids</code> . It is not appropriate to simulate move-



	ment data from a model that was fitted to latitude/longitude data (because <code>simData</code> assumes Cartesian coordinates).
<code>states</code>	TRUE if the simulated states should be returned, FALSE otherwise (default).
<code>retrySims</code>	Number of times to attempt to simulate data within the spatial extent of <code>spatialCovs</code> . If <code>retrySims=0</code> (the default), an error is returned if the simulated tracks(s) move beyond the extent(s) of the raster layer(s). Instead of relying on <code>retrySims</code> , in many cases it might be better to simply expand the extent of the raster layer(s) and/or adjust the step length and turning angle probability distributions. Ignored if <code>spatialCovs=NULL</code> .
<code>lambda</code>	Observation rate for location data. If NULL (the default), location data are obtained at regular intervals. Otherwise <code>lambda</code> is the rate parameter of the exponential distribution for the waiting times between successive location observations, i.e., $1/\lambda$ is the expected time between successive location observations. Only the 'step' and 'angle' data streams are subject to temporal irregularity; any other data streams are observed at temporally-regular intervals. Ignored unless a valid distribution for the 'step' data stream is specified.
<code>errorEllipse</code>	List providing the upper bound for the semi-major axis ( $M$ ; on scale of $x$ - and $y$ -coordinates), semi-minor axis ( $m$ ; on scale of $x$ - and $y$ -coordinates), and orientation ( $r$ ; in degrees) of location error ellipses. If NULL (the default), no location measurement error is simulated. If <code>errorEllipse</code> is specified, then each observed location is subject to bivariate normal errors as described in McClintock et al. (2015), where the components of the error ellipse for each location are randomly drawn from $\text{runif}(1, \min(\text{errorEllipse}\$M), \max(\text{errorEllipse}\$M))$ , $\text{runif}(1, \min(\text{errorEllipse}\$m), \max(\text{errorEllipse}\$m))$ , and $\text{runif}(1, \min(\text{errorEllipse}\$r), \max(\text{errorEllipse}\$r))$ . If only a single value is provided for any of the error ellipse elements, then the corresponding component is fixed to this value for each location. Only the 'step' and 'angle' data streams are subject to location measurement error; any other data streams are observed without error. Ignored unless a valid distribution for the 'step' data stream is specified.
<code>ncores</code>	Number of cores to use for parallel processing. Default: 1 (no parallel processing).
<code>hierStates</code>	A hierarchical model structure <a href="#">Node</a> for the states ('state'). See details.
<code>hierDist</code>	A hierarchical data structure <a href="#">Node</a> for the data streams ('dist'). Currently supported distributions are 'bern', 'beta', 'exp', 'gamma', 'lnorm', 'norm', 'mvnorm2' (bivariate normal distribution), 'mvnorm3' (trivariate normal distribution), 'pois', 'rw_norm' (normal random walk), 'rw_mvnorm2' (bivariate normal random walk), 'rw_mvnorm3' (trivariate normal random walk), 'vm', 'vmConsensus', 'weibull', and 'wrpcauchy'. See details.
<code>hierBeta</code>	A hierarchical data structure <a href="#">Node</a> for the matrix of initial values for the regression coefficients of the transition probabilities at each level of the hierarchy ('beta'). See <a href="#">fithMM</a> .
<code>hierDelta</code>	A hierarchical data structure <a href="#">Node</a> for the matrix of initial values for the regression coefficients of the initial distribution at each level of the hierarchy ('delta'). See <a href="#">fithMM</a> .
<code>hierFormula</code>	A hierarchical formula structure for the transition probability covariates for each level of the hierarchy ('formula'). Default: NULL (only hierarchical-level effects,

with no covariate effects). Any formula terms that are not state- or parameter-specific are included on all of the transition probabilities within a given level of the hierarchy. See details.

hierFormulaDelta	A hierarchical formula structure for the initial distribution covariates for each level of the hierarchy ('formulaDelta'). Default: NULL (no covariate effects and fixPar\$delta is specified on the working scale).
nbHierCovs	A hierarchical data structure <code>Node</code> for the number of covariates ('nbCovs') to simulate for each level of the hierarchy (0 by default). Does not need to be specified if covs is specified. Simulated covariates are provided generic names (e.g., 'cov1.1' and 'cov1.2' for nbHierCovs\$level1\$nbCovs=2) and can be included in hierFormula and/or DM.
obsPerLevel	A hierarchical data structure <code>Node</code> indicating the number of observations for each level of the hierarchy ('obs'). For each level, the 'obs' field can either be 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 observations generated per level for each animal are uniformly picked from this interval. Alternatively, obsPerLevel can be specified as a list of length nbAnimals with each element providing the hierarchical data structure for the number of observations for each level of the hierarchy for each animal, where the 'obs' field can either be the number of observations (if single value) or the bounds of the number of observations (if vector of two values) for each individual.

## Details

- `simHierData` is very similar to `simData` except that instead of simply specifying the number of states (`nbStates`), distributions (`dist`), observations (`obsPerAnimal`), covariates (`nbCovs`), and a single t.p.m. formula (`formula`), the `hierStates` argument specifies the hierarchical nature of the states, the `hierDist` argument specifies the hierarchical nature of the data streams, the `obsPerLevel` argument specifies the number of observations for each level of the hierarchy, the `nbHierCovs` argument specifies the number of covariates for each level of the hierarchy, and the `hierFormula` argument specifies a t.p.m. formula for each level of the hierarchy. All of the hierarchical arguments in `simHierData` are specified as `Node` objects from the `data.tree` package.
- x- and y-coordinate location data are generated only if valid 'step' and 'angle' data streams are specified. Valid distributions for 'step' include 'gamma', 'weibull', 'exp', and 'lnorm'. Valid distributions for 'angle' include 'vm' and 'wrpcauchy'. If only a valid 'step' data stream is specified, then only x-coordinates are generated.
- If DM is specified for a particular data stream, then the initial values are specified on the working (i.e., beta) scale of the parameters. The working scale of each parameter is determined by the link function used. The function `getParDM` is intended to help with obtaining initial values on the working scale when specifying a design matrix and other parameter constraints.
- Simulated data that are temporally regular (i.e., `lambda=NULL`) and without location measurement error (i.e., `errorEllipse=NULL`) are returned as a `momentuHMMDData` (or `momentuHierHMMDData`) object suitable for analysis using `fitHMM`.

- Simulated location data that are temporally-irregular (i.e.,  $\lambda > 0$ ) and/or with location measurement error (i.e., `errorEllipse!=NULL`) are returned as a data frame suitable for analysis using `crawlWrap`.
- 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 formula 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.
- State-specific formulas can be specified in DM using special formula functions. These special functions can take the names `paste0("state", 1:nbStates)` (where the integer indicates the state-specific formula). For example, `DM=list(step=list(mean=~cov1+state1(cov2), sd=~cov2+state2(cov1)))` includes `cov1` on the mean parameter for all states, `cov2` on the mean parameter for state 1, `cov2` on the sd parameter for all states, and `cov1` on the sd parameter for state 2.
- State- and parameter-specific formulas can be specified for transition probabilities in formula using special formula functions. These special functions can take the names `paste0("state", 1:nbStates)` (where the integer indicates the current state from which transitions occur), `paste0("toState", 1:nbStates)` (where the integer indicates the state to which transitions occur), or `paste0("betaCol", nbStates*(nbStates-1))` (where the integer indicates the column of the beta matrix). For example with `nbStates=3`, `formula=~cov1+betaCol1(cov2)+state3(cov3)+toState1(cov4)` includes `cov1` on all transition probability parameters, `cov2` on the beta column corresponding to the transition from state 1->2, `cov3` on transition probabilities from state 3 (i.e., beta columns corresponding to state transitions 3->1 and 3->2), and `cov4` on transition probabilities to state 1 (i.e., beta columns corresponding to state transitions 2->1 and 3->1).
- Cyclical relationships (e.g., hourly, monthly) may be simulated using the `cosinor(x, period)` special formula function for covariate `x` and sine curve period of time length `period`. For example, if the data are hourly, a 24-hour cycle can be simulated using `~cosinor(cov1, 24)`, where the covariate `cov1` is a repeating series of integers `0, 1, ..., 23, 0, 1, ..., 23, 0, 1, ...` (note that `simData` will not do this for you, the appropriate covariate must be specified using the `covs` argument; see example below). The `cosinor(x, period)` function converts `x` to 2 covariates `cosinorCos(x)=cos(2*pi*x/period)` and `cosinorSin(x)=sin(2*pi*x/period)` for inclusion in the model (i.e., 2 additional parameters per state). The amplitude of the sine wave is thus `sqrt(B_cos^2 + B_sin^2)`, where `B_cos` and `B_sin` are the working parameters corresponding to `cosinorCos(x)` and `cosinorSin(x)`, respectively (e.g., see Cornelissen 2014).

When the circular-circular regression model is used, the special function `angleFormula(cov, strength, by)` can be used in DM for the mean of angular distributions (i.e. `'vm'`, `'vmConsensus'`, and `'wrpcauchy'`), where `cov` is an angle covariate (e.g. wind direction), `strength` is a positive real covariate (e.g. wind speed), and `by` is an optional factor variable for individual- or group-level effects (e.g. ID, sex). This allows angle covariates to be weighted based on their strength or importance at time step `t` as in Rivest et al. (2016).

- 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'` (or `'obsPerLevel'` for `simHierData`), 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).
- For `simData`, when covariates are not included in `formulaDelta` (i.e. `formulaDelta=NULL`), then `delta` is specified as a vector of length `nbStates` that sums to 1. When covariates are in-

cluded in `formulaDelta`, then `delta` must be specified as a  $k \times (\text{nbStates}-1)$  matrix of working parameters, where  $k$  is the number of regression coefficients and the columns correspond to states 2:nbStates. For example, in a 3-state HMM with `formulaDelta=~cov1+cov2`, the matrix `delta` has three rows (intercept + two covariates) and 2 columns (corresponding to states 2 and 3). The initial distribution working parameters are transformed to the real scale as  $\exp(\text{covsDelta} \times \text{Delta}) / \text{rowSums}(\exp(\text{covsDelta} \times \text{Delta}))$ , where `covsDelta` is the  $N \times k$  design matrix, `Delta=cbind(rep(0,k),delta)` is a  $k \times \text{nbStates}$  matrix of working parameters, and  $N=\text{length}(\text{unique}(\text{data}\$ID))$ .

- For `simHierData`, `delta` must be specified as a  $k \times (\text{nbStates}-1)$  matrix of working parameters, where  $k$  is the number of regression coefficients and the columns correspond to states 2:nbStates.

## Value

If the simulated data are temporally regular (i.e., `lambda=NULL`) with no measurement error (i.e., `errorEllipse=NULL`), an object `momentuHMMData` (or `momentuHierHMMData`), i.e., a dataframe of:

ID	The ID(s) of the observed animal(s)
...	Data streams as specified by <code>dist</code> (or <code>hierDist</code> )
x	Either easting or longitude (if data streams include valid non-negative distribution for 'step')
y	Either northing or latitude (if data streams include valid non-negative distribution for 'step')
...	Covariates (if any)

If simulated location data are temporally irregular (i.e., `lambda>0`) and/or include measurement error (i.e., `errorEllipse!=NULL`), a dataframe of:

time	Numeric time of each observed (and missing) observation
ID	The ID(s) of the observed animal(s)
x	Either easting or longitude observed location
y	Either northing or latitude observed location
...	Data streams that are not derived from location (if applicable)
...	Covariates at temporally-regular true ( <code>mux,muy</code> ) locations (if any)
mux	Either easting or longitude true location
muy	Either northing or latitude true location
error_semimajor_axis	error ellipse semi-major axis (if applicable)
error_semiminor_axis	error ellipse semi-minor axis (if applicable)
error_ellipse_orientation	error ellipse orientation (if applicable)
ln.sd.x	log of the square root of the x-variance of bivariate normal error (if applicable; required for error ellipse models in <code>crawlWrap</code> )

ln.sd.y	log of the square root of the y-variance of bivariate normal error (if applicable; required for error ellipse models in <a href="#">crawlWrap</a> )
error.corr	correlation term of bivariate normal error (if applicable; required for error ellipse models in <a href="#">crawlWrap</a> )

## References

- Cornelissen, G. 2014. Cosinor-based rhythmometry. *Theoretical Biology and Medical Modelling* 11:16.
- McClintock BT, London JM, Cameron MF, Boveng PL. 2015. Modelling animal movement using the Argos satellite telemetry location error ellipse. *Methods in Ecology and Evolution* 6(3):266-277.
- Rivest, LP, Duchesne, T, Nicosia, A, Fortin, D, 2016. A general angular regression model for the analysis of data on animal movement in ecology. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 65(3):445-463.
- Leos-Barajas, V., Gangloff, E.J., Adam, T., Langrock, R., van Beest, F.M., Nabe-Nielsen, J. and Morales, J.M. 2017. Multi-scale modeling of animal movement and general behavior data using hidden Markov models with hierarchical structures. *Journal of Agricultural, Biological and Environmental Statistics*, 22 (3), 232-248.

## See Also

[prepData](#), [simObsData](#)

## Examples

```
# 1. Pass a fitted model to simulate from
# (m is a momentuHMM 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)

## Not run:
# 2. Pass the parameters of the model to simulate from
stepPar <- c(1,10,1,5,0.2,0.3) # mean_1, mean_2, sd_1, sd_2, zeromass_1, zeromass_2
anglePar <- c(pi,0,0.5,2) # mean_1, mean_2, concentration_1, concentration_2
omegaPar <- c(1,10,10,1) # shape1_1, shape1_2, shape2_1, shape2_2
stepDist <- "gamma"
angleDist <- "vm"
omegaDist <- "beta"
data <- simData(nbAnimals=4,nbStates=2,dist=list(step=stepDist,angle=angleDist,omega=omegaDist),
               Par=list(step=stepPar,angle=anglePar,omega=omegaPar),nbCovs=2,
               zeroInflation=list(step=TRUE),
               obsPerAnimal=obsPerAnimal)

# 3. Include covariates
# (note that it is useless to specify "nbCovs", which are overruled
# by the number of columns of "cov")
cov <- data.frame(temp=log(rnorm(500,20,5)))
stepPar <- c(log(10),0.1,log(100),-0.1,log(5),log(25)) # working scale parameters for step DM
```

```

anglePar <- c(pi,0,0.5,2) # mean_1, mean_2, concentration_1, concentration_2
stepDist <- "gamma"
angleDist <- "vm"
data <- simData(nbAnimals=2,nbStates=2,dist=list(step=stepDist,angle=angleDist),
               Par=list(step=stepPar,angle=anglePar),
               DM=list(step=list(mean=~temp,sd=~1)),
               covs=cov,
               obsPerAnimal=obsPerAnimal)

# 4. Include example 'forest' spatial covariate raster layer
# nbAnimals and obsPerAnimal kept small to reduce example run time
spatialCov<-list(forest=forest)
data <- simData(nbAnimals=1,nbStates=2,dist=list(step=stepDist,angle=angleDist),
               Par=list(step=c(100,1000,50,100),angle=c(0,0,0.1,5)),
               beta=matrix(c(5,-10,-25,50),nrow=2,ncol=2,byrow=TRUE),
               formula=~forest,spatialCovs=spatialCov,
               obsPerAnimal=250,states=TRUE,
               retrySims=100)

# 5. Specify design matrix for 'omega' data stream
# natural scale parameters for step and angle
stepPar <- c(1,10,1,5) # shape_1, shape_2, scale_1, scale_2
anglePar <- c(pi,0,0.5,0.7) # mean_1, mean_2, concentration_1, concentration_2

# working scale parameters for omega DM
omegaPar <- c(log(1),0.1,log(10),-0.1,log(10),-0.1,log(1),0.1)

stepDist <- "weibull"
angleDist <- "wrpcauchy"
omegaDist <- "beta"

data <- simData(nbStates=2,dist=list(step=stepDist,angle=angleDist,omega=omegaDist),
               Par=list(step=stepPar,angle=anglePar,omega=omegaPar),nbCovs=2,
               DM=list(omega=list(shape1=~cov1,shape2=~cov2)),
               obsPerAnimal=obsPerAnimal,states=TRUE)

# 6. Include temporal irregularity and location measurement error
lambda <- 2 # expect 2 observations per time step
errorEllipse <- list(M=50,m=25,r=180)
obsData <- simData(model=m,obsPerAnimal=obsPerAnimal,
                  lambda=lambda, errorEllipse=errorEllipse)

# 7. Cosinor and state-dependent formulas
nbStates<-2
dist<-list(step="gamma")
Par<-list(step=c(100,1000,50,100))

# include 24-hour cycle on all transition probabilities
# include 12-hour cycle on transitions from state 2
formula=~cosinor(hour24,24)+state2(cosinor(hour12,12))

# specify appropriate covariates
covs<-data.frame(hour24=0:23,hour12=0:11)

```

```

beta<-matrix(c(-1.5,1,1,NA,NA,-1.5,-1,-1,1,1),5,2)
# row names for beta not required but can be helpful
rownames(beta)<-c("(Intercept)",
                 "cosinorCos(hour24, 24)",
                 "cosinorSin(hour24, 24)",
                 "cosinorCos(hour12, 12)",
                 "cosinorSin(hour12, 12)")
data.cos<-simData(nbStates=nbStates,dist=dist,Par=Par,
                 beta=beta,formula=formula,covs=covs)

# 8. Piecewise constant B-spline on step length mean and angle concentration
nObs <- 1000 # length of simulated track
cov <- data.frame(time=1:nObs) # time covariate for splines
dist <- list(step="gamma",angle="vm")
stepDM <- list(mean=~splines2::bSpline(time,df=2,degree=0),sd=~1)
angleDM <- list(mean=~1,concentration=~splines2::bSpline(time,df=2,degree=0))
DM <- list(step=stepDM,angle=angleDM)
Par <- list(step=c(log(1000),1,-1,log(100)),angle=c(0,log(10),2,-5))

data.spline<-simData(obsPerAnimal=nObs,nbStates=1,dist=dist,Par=Par,DM=DM,covs=cov)

# 9. Initial state (delta) based on covariate
nObs <- 100
dist <- list(step="gamma",angle="vm")
Par <- list(step=c(100,1000,50,100),angle=c(0,0,0.01,0.75))

# create sex covariate
cov <- data.frame(sex=factor(rep(c("F","M"),each=nObs))) # sex covariate
formulaDelta <- ~ sex + 0

# Female begins in state 1, male begins in state 2
delta <- matrix(c(-100,100),2,1,dimnames=list(c("sexF","sexM"),"state 2"))

data.delta<-simData(nbAnimals=2,obsPerAnimal=nObs,nbStates=2,dist=dist,Par=Par,
                  delta=delta,formulaDelta=formulaDelta,covs=cov,
                  beta=matrix(-1.5,1,2),states=TRUE)

## End(Not run)

```

---

simObsData

*Observation error simulation tool*


---

## Description

Simulates observed location data subject to temporal irregularity and/or location measurement error

## Usage

```
simObsData(data, lambda, errorEllipse, ...)
```

```
## S3 method for class 'momentuHMMDData'
simObsData(data, lambda, errorEllipse, ...)

## S3 method for class 'momentuHierHMMDData'
simObsData(data, lambda, errorEllipse, coordLevel, ...)
```

## Arguments

data	A <code>momentuHMMDData</code> or <code>momentuHierHMMDData</code> object with necessary fields 'x' (easting/longitudinal coordinates) and 'y' (northing/latitudinal coordinates)
lambda	Observation rate for location data. If NULL, location data are kept at temporally-regular intervals. Otherwise lambda is the rate parameter of the exponential distribution for the waiting times between successive location observations, i.e., 1/lambda is the expected time between successive location observations. Only the 'step' and 'angle' data streams (or multivariate normal data streams identified by <code>mvnCoords</code> ) are subject to temporal irregularity; any other data streams are kept at temporally-regular intervals. Ignored unless a valid distribution for the 'step' (or 'mvnCoord') data stream has been specified.
errorEllipse	List providing the bounds for the semi-major axis (M; on scale of x- and y-coordinates), semi-minor axis (m; on scale of x- and y-coordinates), and orientation (r; in degrees) of location error ellipses. If NULL, no location measurement error is simulated. If <code>errorEllipse</code> is specified, then each observed location is subject to bivariate normal errors as described in McClintock et al. (2015), where the components of the error ellipse for each location are randomly drawn from <code>runif(1, min(errorEllipse\$M), max(errorEllipse\$M))</code> , <code>runif(1, min(errorEllipse\$m), max(errorEllipse\$m))</code> , and <code>runif(1, min(errorEllipse\$r), max(errorEllipse\$r))</code> . If only a single value is provided for any of the error ellipse elements, then the corresponding component is fixed to this value for each location. Only the 'step' and 'angle' data streams are subject to location measurement error; any other data streams are observed without error. Ignored unless a valid distribution for the 'step' data stream is specified.
...	further arguments passed to or from other methods
coordLevel	Level of the hierarchy in which the location data are obtained

## Details

Simulated location data that are temporally-irregular (i.e., `lambda > 0`) and/or with location measurement error (i.e., `errorEllipse != NULL`) are returned as a data frame suitable for analysis using [crawlWrap](#).

## Value

A dataframe of:

time	Numeric time of each observed (and missing) observation
ID	The ID(s) of the observed animal(s)
x	Either easting or longitude observed location



y	Either northing or latitude observed location
...	Data streams that are not derived from location (if applicable)
...	Covariates at temporally-regular true (mux,muy) locations (if any)
mux	Either easting or longitude true location
muy	Either northing or latitude true location
error_semimajor_axis	error ellipse semi-major axis (if applicable)
error_semiminor_axis	error ellipse semi-minor axis (if applicable)
error_ellipse_orientation	error ellipse orientation (if applicable)
ln.sd.x	log of the square root of the x-variance of bivariate normal error (if applicable; required for error ellipse models in <a href="#">crawlWrap</a> )
ln.sd.y	log of the square root of the y-variance of bivariate normal error (if applicable; required for error ellipse models in <a href="#">crawlWrap</a> )
error.corr	correlation term of bivariate normal error (if applicable; required for error ellipse models in <a href="#">crawlWrap</a> )

## References

McClintock BT, London JM, Cameron MF, Boveng PL. 2015. Modelling animal movement using the Argos satellite telemetry location error ellipse. *Methods in Ecology and Evolution* 6(3):266-277.

## See Also

[crawlWrap](#), [prepData](#), [simData](#)

[simHierData](#)

## Examples

```
# extract momentuHMMData example
data <- example$m$data
lambda <- 2 # expect 2 observations per time step
errorEllipse <- list(M=c(0,50),m=c(0,50),r=c(0,180))
obsData1 <- simObsData(data,lambda=lambda,errorEllipse=errorEllipse)

errorEllipse <- list(M=50,m=50,r=180)
obsData2 <- simObsData(data,lambda=lambda,errorEllipse=errorEllipse)
```

---

stateProbs	<i>State probabilities</i>
------------	----------------------------

---

**Description**

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

**Usage**

```
stateProbs(m, hierarchical = FALSE)
```

**Arguments**

<code>m</code>	A <code>momentuHMM</code> or <code>momentuHierHMM</code> object.
<code>hierarchical</code>	Logical indicating whether or not to return a list of state probabilities for each level of a hierarchical HMM. Ignored unless <code>m</code> is a <code>momentuHierHMM</code> 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 momentuHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

sp <- stateProbs(m)
```

---

stationary	<i>Stationary state probabilities</i>
------------	---------------------------------------

---

**Description**

Calculates the stationary probabilities of each state based on covariate values.

**Usage**

```
stationary(model, covs, covIndex)
```

**Arguments**

model	momentuHMM, miHMM, or miSum object
covs	Either a data frame or a design matrix of covariates. If covs is not provided, then the stationary probabilities are calculated based on the covariate data for each time step.
covIndex	Integer vector indicating specific rows of the data to be used in the calculations. This can be useful for reducing unnecessarily long computation times, e.g., when formula includes factor covariates (such as ID) but no temporal covariates. Ignored unless covs is missing.

**Value**

A list of length `model$conditions$mixtures` where each element is a matrix of stationary state probabilities for each mixture. For each matrix, each row corresponds to a row of `covs`, and each column corresponds to a state.

**Examples**

```
# m is a momentuHMM 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))

# get stationary distribution for first 3 observations
stationary(m, covIndex = c(1,2,3))
```

---

```
summary.momentuHMMDData
```

*Summary momentuHMMDData*

---

**Description**

Summary momentuHMMDData

**Usage**

```
## S3 method for class 'momentuHMMDData'
summary(object, dataNames = c("step", "angle"), animals = NULL, ...)

## S3 method for class 'momentuHierHMMDData'
summary(object, dataNames = c("step", "angle", "level"), animals = NULL, ...)
```

**Arguments**

object	A <a href="#">momentuHMMData</a> or <a href="#">momentuHierHMMData</a> object.
dataNames	Names of the variables to summarize. Default is <code>dataNames=c("step", "angle")</code> .
animals	Vector of indices or IDs of animals for which data will be summarized. Default: NULL ; data for all animals are summarized.
...	Currently unused. For compatibility with generic method.

**Examples**

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

summary(data, dataNames=c("step", "angle", "cov1", "cov2"))
```

---

timeInStates	<i>Calculate proportion of time steps assigned to each state (i.e. “activity budgets”)</i>
--------------	--

---

**Description**

Calculate proportion of time steps assigned to each state (i.e. “activity budgets”)

**Usage**

```
timeInStates(m, by = NULL, alpha = 0.95, ncores = 1)

## S3 method for class 'momentuHMM'
timeInStates(m, by = NULL, alpha = 0.95, ncores = 1)

## S3 method for class 'HMMfits'
timeInStates(m, by = NULL, alpha = 0.95, ncores = 1)

## S3 method for class 'miHMM'
timeInStates(m, by = NULL, alpha = 0.95, ncores = 1)
```

**Arguments**

m	A <a href="#">momentuHMM</a> , <a href="#">miHMM</a> , or <a href="#">HMMfits</a> object.
by	A character vector indicating any groupings by which to calculate the proportions, such as individual (“ID”) or group-level (e.g. sex or age class) covariates. Default is NULL (no groupings are used).
alpha	Significance level for calculating confidence intervals of pooled estimates. Default: 0.95. Ignored unless m is a <a href="#">miHMM</a> or <a href="#">HMMfits</a> object.
ncores	Number of cores to use for parallel processing. Default: 1 (no parallel processing). Ignored unless m is a <a href="#">miHMM</a> or <a href="#">HMMfits</a> object.

**Value**

If `m` is a `momentuHMM` object, a data frame containing the estimated activity budgets for each state (grouped according to `by`). If `m` is a `miHMM` or `HMMfits` object, a list containing the activity budget estimates, standard errors, lower bounds, and upper bounds across all imputations.

**Examples**

```
# m is a momentuHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m
timeInStates(m)
timeInStates(m, by = "ID")
```

---

<code>trMatrix_rcpp</code>	<i>Transition probability matrix</i>
----------------------------	--------------------------------------

---

**Description**

Computation of the transition probability matrix, as a function of the covariates and the regression parameters. Written in C++. Used in `viterbi`.

**Usage**

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

**Arguments**

<code>nbStates</code>	Number of states
<code>beta</code>	Matrix of regression parameters
<code>covs</code>	Matrix of covariate values
<code>betaRef</code>	Indices of reference elements for t.p.m. multinomial logit link.

**Value**

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

---

turnAngle	<i>Turning angle</i>
-----------	----------------------

---

### Description

Used in [prepData](#) and [simData](#).

### Usage

```
turnAngle(x, y, z, type = "UTM", angleCov = FALSE)
```

### Arguments

x	First point
y	Second point
z	Third point
type	'UTM' if easting/northing provided (the default), 'LL' if longitude/latitude. If type='LL' then the <a href="#">geosphere</a> package must be installed.
angleCov	logical indicating to not return NA when x=y or y=z. Default: FALSE (i.e. NA is returned if x=y or y=z).

### Value

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

If type='LL' then turning angle is calculated based on initial bearings using [bearing](#).

### Examples

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

---

viterbi	<i>Viterbi algorithm</i>
---------	--------------------------

---

**Description**

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

**Usage**

```
viterbi(m, hierarchical = FALSE)
```

**Arguments**

<code>m</code>	An object <code>momentuHMM</code> or <code>momentuHierHMM</code>
<code>hierarchical</code>	Logical indicating whether or not to return a list of Viterbi-decoded states for each level of a hierarchical HMM. Ignored unless <code>m</code> is a <code>momentuHierHMM</code> object.

**Value**

The sequence of most probable states. If `hierarchical` is `TRUE`, then a list of the most probable states for each level of the hierarchy is returned.

**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 momentuHMM object (as returned by fitHMM), automatically loaded with the package
m <- example$m

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

---

w2n	<i>Scaling function: working to natural parameters</i>
-----	--

---

**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,
  circularAngleMean,
  consensus,
  stationary,
  fullDM,
  DMind,
  nbObs,
  dist,
  Bndind,
  nc,
  meanind,
  covsDelta,
  workBounds,
  covsPi
)
```

**Arguments**

wpar	Vector of working parameters.
bounds	Named list of 2-column matrices specifying bounds on the natural (i.e, real) scale of the probability distribution parameters for each data stream.
parSize	Named list indicating the number of natural parameters of the data stream probability distributions
nbStates	The number of states of the HMM.
nbCovs	The number of beta covariates.
estAngleMean	Named list indicating whether or not to estimate the angle mean for data streams with angular distributions ('vm' and 'wrpcauchy').
circularAngleMean	Named list indicating whether to use circular-linear or circular-circular regression on the mean of circular distributions ('vm' and 'wrpcauchy') for turning angles. See <a href="#">fitHMM</a> .
consensus	Named list indicating whether to use the circular-circular regression consensus model
stationary	FALSE if there are time-varying covariates in formula or any covariates in formulaDelta. If TRUE, the initial distribution is considered equal to the stationary distribution. Default: FALSE.
fullDM	Named list containing the full (i.e. not shorthand) design matrix for each data stream.



DMind	Named list indicating whether fullDM includes individual- and/or temporal-covariates for each data stream specifies (-1,1) bounds for the concentration parameters instead of the default [0,1) bounds.
nbObs	Number of observations in the data.
dist	Named list indicating the probability distributions of the data streams.
Bndind	Named list indicating whether DM is NULL with default parameter bounds for each data stream.
nc	indicator for zeros in fullDM
meanind	index for circular-circular regression mean angles with at least one non-zero entry in fullDM
covsDelta	data frame containing the delta model covariates
workBounds	named list of 2-column matrices specifying bounds on the working scale of the probability distribution, transition probability, and initial distribution parameters
covsPi	data frame containing the pi model covariates

### Value

A list of:

...	Matrices containing the natural parameters for each data stream (e.g., 'step', 'angle', etc.)
beta	Matrix of regression coefficients of the transition probabilities
delta	Initial distribution

### Examples

```
## Not run:
m<-example$m
nbStates <- 2
nbCovs <- 2
parSize <- list(step=2,angle=2)
par <- list(step=c(t(m$mle$step)),angle=c(t(m$mle$angle)))
bounds <- m$conditions$bounds
beta <- matrix(rnorm(6),ncol=2,nrow=3)
delta <- c(0.6,0.4)

#working parameters
wpar <- momentuHMM:::n2w(par,bounds,list(beta=beta),log(delta[-1]/delta[1]),nbStates,
m$conditions$estAngleMean,NULL,m$conditions$Bndind,
m$conditions$dist)

#natural parameter
p <- momentuHMM:::w2n(wpar,bounds,parSize,nbStates,nbCovs,m$conditions$estAngleMean,
m$conditions$circularAngleMean,lapply(m$conditions$dist,function(x) x=="vmConsensus"),
m$conditions$stationary,m$conditions$fullDM,
m$conditions$DMind,1,m$conditions$dist,m$conditions$Bndind,
matrix(1,nrow=length(unique(m$data$ID)),ncol=1),covsDelta=m$covsDelta,
workBounds=m$conditions$workBounds)
```

```
## End(Not run)
```

---

 XBloop\_rcpp

*Get XB*


---

### Description

Loop for computation of design matrix (X) times the working scale parameters (B). Written in C++. Used in [w2n](#).

### Usage

```
XBloop_rcpp(
  DM,
  Xvec,
  nbObs,
  nr,
  nc,
  circularAngleMean,
  consensus,
  rindex,
  cindex,
  nbStates,
  refCoeff = 1
)
```

### Arguments

DM	design matrix
Xvec	working parameters
nbObs	number of observations
nr	number of rows in design matrix
nc	number of column in design matrix
circularAngleMean	indicator for whether or not circular-circular regression model
consensus	indicator for whether or not circular-circular regression consensus model
rindex	row index for design matrix
cindex	column index for design matrix
nbStates	number of states
refCoeff	intercept coefficient for circular-circular regression model

### Value

XB matrix

# Index

- acf, [98](#)
- AIC.momentuHMM, [4](#), [39](#), [73](#)
- AICweights, [4](#), [5](#), [39](#), [73](#)
- allProbs, [7](#)
  
- bearing, [25](#), [134](#)
- brick, [105](#), [118](#)
- bs, [36](#), [37](#), [117](#), [120](#)
- bSpline, [36](#), [37](#), [117](#), [120](#)
  
- checkPar0, [7](#), [41](#)
- CIbeta, [11](#), [65](#), [66](#)
- circAngles, [12](#), [35](#), [40](#), [106](#), [119](#)
- CIreal, [13](#), [65](#), [66](#)
- crawlMerge, [15](#), [106](#), [107](#)
- crawlWrap, [15](#), [16](#), [20](#), [21](#), [30](#), [41](#), [74](#), [75](#), [90](#), [105](#), [107](#), [123–125](#), [128](#), [129](#)
- CRS, [100](#)
- crwData, [15](#), [19](#), [20](#), [22](#), [61](#), [71–75](#), [90](#), [105–107](#)
- crwHierData, [15](#), [19](#), [20](#), [21](#), [62](#), [71](#), [73–75](#), [90](#), [105–107](#)
- crwHierSim, [21](#), [62](#), [71](#), [74](#), [75](#)
- crwMLE, [17–19](#)
- crwPostIS, [74](#), [75](#)
- crwPredict, [18](#), [19](#)
- crwPredictPlot, [90](#)
- crwSim, [21](#), [63](#), [71](#), [72](#), [74](#), [75](#)
- crwSimulator, [21](#), [22](#), [73–75](#)
- cSpline, [36](#), [37](#), [117](#), [120](#)
  
- data.tree, [42](#), [122](#)
- dataEllipse, [78](#)
- dbern\_rcpp, [22](#)
- dbeta\_rcpp, [22](#)
- dcat\_rcpp, [23](#)
- dexp\_rcpp, [23](#)
- dgamma\_rcpp, [24](#)
- distAngle, [24](#)
- dlnorm\_rcpp, [25](#)
  
- dlogis\_rcpp, [26](#)
- dmvnorm\_rcpp, [26](#)
- dnbinom\_rcpp, [27](#)
- dnorm\_rcpp, [27](#)
- dpois\_rcpp, [28](#)
- dt\_rcpp, [28](#)
- dvm\_rcpp, [29](#)
- dweibull\_rcpp, [29](#)
- dwrpcauchy\_rcpp, [30](#)
  
- example (exampleData), [30](#)
- exampleData, [30](#)
- expandPar, [31](#)
  
- fillCols, [19](#)
- fitHMM, [7](#), [10](#), [12](#), [19](#), [30](#), [33](#), [47](#), [48](#), [50–53](#), [56](#), [57](#), [60](#), [65](#), [66](#), [71–75](#), [85](#), [104–106](#), [112](#), [114](#), [115](#), [120–122](#), [136](#)
- fitHMM.momentuHMMDData, [41](#)
- forest (exampleData), [30](#)
- formatHierHMM, [47](#)
  
- geom\_point, [100–102](#)
- geosphere, [134](#)
- get\_map, [100](#)
- getCovNames, [49](#)
- getDM\_rcpp, [49](#)
- getPar, [50](#), [53](#), [57](#)
- getPar0, [50](#), [51](#), [57](#)
- getParDM, [40](#), [43](#), [50](#), [53](#), [54](#), [75](#), [122](#)
- getTrProbs, [58](#)
- grad, [97](#)
  
- HMMfits, [5](#), [6](#), [61](#), [63](#), [98](#), [110](#), [114](#), [115](#), [132](#), [133](#)
  
- is.crwData, [61](#)
- is.crwHierData, [62](#)
- is.crwHierSim, [62](#)
- is.crwSim, [63](#)

- is.HMMfits, 63
- is.miHMM, 64
- is.miSum, 64
- is.momentuHierHMM, 65
- is.momentuHierHMMDData, 65
- is.momentuHMM, 66
- is.momentuHMMDData, 66
- iSpline, 36, 37, 117, 120
  
- logAlpha, 7, 67
- logBeta, 7, 67
  
- MIcombine, 77–79
- miExample (exampleData), 30
- MifitHMM, 7, 10, 12, 19–22, 30, 48, 50–53, 57, 61, 62, 68, 77, 92, 105, 106, 114, 115
- miHMM, 5, 6, 50, 52, 59, 64, 67, 75, 76, 98–103, 110, 111, 114, 115, 120, 131–133
- MIpool, 30, 38, 52, 71, 72, 75, 77, 94, 99, 111
- miSum, 7, 50, 52, 59, 64, 67, 75, 77, 78, 79, 98–103, 110, 111, 114, 115, 120, 131
- mixtureProbs, 79
- momentuHierHMM, 42, 52, 59, 65, 77–79, 80, 98, 102, 103, 120, 130, 135
- momentuHierHMMDData, 9, 34, 47, 55, 56, 59, 65, 71, 74, 75, 81, 106, 122, 124, 128, 132
- momentuHMM, 5–7, 30, 42, 49–52, 59, 61, 66, 67, 75, 77, 79, 82, 98–103, 110–112, 114, 115, 120, 131–133, 135
- momentuHMMDData, 9, 21, 22, 34, 55, 56, 59, 66, 71, 72, 74, 75, 82, 88, 100, 101, 106, 122, 124, 128, 132
- mSpline, 36, 37, 117, 120
  
- n2w, 83
- n1m, 36, 38, 72, 73, 112
- nLogLike, 84
- nLogLike\_rcpp, 87
- Node, 10, 14, 39, 42, 47, 48, 52, 57, 60, 74, 121, 122
- ns, 36, 37, 117, 120
  
- optim, 18, 38, 73, 112
  
- par, 92, 94, 97, 103
- parDef, 49, 88
- plot.crwData, 90
- plot.crwHierData (plot.crwData), 90
- plot.miHMM, 38, 72, 91
- plot.miSum, 38, 72, 93
- plot.momentuHierHMMDData (plot.momentuHMMDData), 97
- plot.momentuHMM, 38, 42, 72, 95
- plot.momentuHMMDData, 97
- plotPR, 65, 66, 98
- plotSat, 99
- plotSpatialCov, 38, 72, 101
- plotStates, 65, 66, 102
- plotStationary, 103
- prepData, 25, 34, 35, 40, 43, 73–75, 104, 125, 129, 134
- print.miHMM, 108
- print.miSum, 109
- print.momentuHierHMM (print.momentuHMM), 110
- print.momentuHMM, 39, 73, 110
- pseudoRes, 65–67, 110
  
- qqPlot, 98
  
- randomEffects, 111
- raster, 31, 101, 105, 118
- rlang, 4, 5
  
- seq.POSIXt, 17
- setModelName, 114
- setStateNames, 114
- simData, 17, 19, 25, 31, 34, 40, 43, 90, 115, 122, 129, 134
- simHierData, 34, 43, 90, 129
- simHierData (simData), 115
- simObsData, 90, 125, 127
- SpatialPointsDataFrame, 17
- spDistsN1, 25
- stack, 105, 118
- stateProbs, 65–67, 130
- stationary, 130
- summary.momentuHierHMMDData (summary.momentuHMMDData), 131
- summary.momentuHMMDData, 131
  
- timeInStates, 132
- trMatrix\_rcpp, 133
- turnAngle, 105, 134
  
- viterbi, 7, 65, 66, 78, 92, 95, 97, 133, 135
  
- w2n, 135, 138

`XBloop_rcpp`, [138](#)