Package: smam (via r-universe)

August 22, 2024

Title Statistical Modeling of Animal Movements

Version 0.7.2

Date 2024-01-10

Description Animal movement models including Moving-Resting Process with Embedded Brownian Motion (Yan et al., 2014, <doi:10.1007/s10144-013-0428-8>; Pozdnyakov et al., 2017, <doi:10.1007/s11009-017-9547-6>), Brownian Motion with Measurement Error (Pozdnyakov et al., 2014, <doi:10.1890/13-0532.1>), Moving-Resting-Handling Process with Embedded Brownian Motion (Pozdnyakov et al., 2020, <doi:10.1007/s11009-020-09774-1>), Moving-Resting Process with Measurement Error (Hu et al., 2021, <doi:10.1111/2041-210X.13694>), Moving-Moving Process with two Embedded Brownian Motions.

Depends R (>= 3.5.0)

License GPL (>= 3.0)

Encoding UTF-8

LazyData true

Imports nloptr, Matrix, stats, Rcpp, RcppParallel, doParallel, foreach, parallel, doSNOW, methods, numDeriv, EnvStats

Suggests R.rsp

LinkingTo Rcpp, RcppGSL, RcppParallel

SystemRequirements GNU GSL, GNU make

VignetteBuilder R.rsp

BugReports `https://github.com/ChaoranHu/smam/issues`

URL `https://github.com/ChaoranHu/smam`

RoxygenNote 7.2.3

NeedsCompilation yes

Repository https://chaoranhu.r-universe.dev

RemoteUrl https://github.com/chaoranhu/smam

RemoteRef HEAD

RemoteSha 0d1164958ab218cc64efb00445301b032bca94dd

Contents

`approxNormalOrder` *Auxiliary for Preparing Discrete Distribution used to approximating Standard Normal Distribution*

Description

Auxiliary for preparing discrete distribution used to approximate standard normal. This function generates order statistics of standard normal with same probability assigned. Then, the discrete distribution is standardized to variance one and mean zero.

Usage

```
approxNormalOrder(m)

approxNormalOrder2(m, width)
```

Arguments

`m`	int, the number of order statistics used
`width`	the width between two consecutive grid points.

Details

This function use `EnvStats::evNormOrdStats` to get the order statisics of standard normal distribution. The same probability is assigned for each order statistics.

Value

A numeric matrix with first column is support of discrete distribution and second column is corresponding p.m.f..

Author(s)

Chaoran Hu

See Also

`EnvStats::evNormOrdStats` for order statisics of standard normal. `fitMRMEapprox` for fit MRME with approximated measurement error.

`dtm` *Density for Time Spent in Moving or Resting*

Description

Density for time spent in moving or resting in a time interval, unconditional or conditional on the initial state.

Usage

```
dtm(w, t, lamM, lamR, s0 = NULL)

dtr(w, t, lamM, lamR, s0 = NULL)
```

Arguments

`w`	time points at which the density is to be evaluated
`t`	length of the time interval
`lamM`	rate parameter of the exponentially distributed duration in moving
`lamR`	rate parameter of the exponentially distributed duration in resting
`s0`	initial state. If `NULL`, the unconditional density is returned; otherwise, it is one of "m" or "s", standing for moving and resting, respectively, and the conditional density is returned given the initial state.

Details

`dtm` returns the density for time in moving; `dtr` returns the density for time in resting.

Value

a vector of the density evaluated at `w`.

Functions

- `dtr()`: Density of time spent in resting

References

Yan, J., Chen, Y., Lawrence-Apfel, K., Ortega, I. M., Pozdnyakov, V., Williams, S., and Meyer, T. (2014) A moving-resting process with an embedded Brownian motion for animal movements. Population Ecology. 56(2): 401–415.

Examples

```
lamM <- 1
lamR <- c(1/2, 1, 2)
lr <- length(lamR)
totalT <- 10
old.par <- par(no.readonly=TRUE)
par(mfrow=c(1, 2), mar=c(2.5, 2.5, 1.1, 0.1), mgp=c(1.5, 0.5, 0), las=1)
curve(dtm(x, totalT, 1, 1/2, "m"), 0, totalT, lty=1, ylim=c(0, 0.34),
      xlab="M(10)", ylab="density")
curve(dtm(x, totalT, 1, 1, "m"), 0, totalT, lty=2, add=TRUE)
curve(dtm(x, totalT, 1, 2, "m"), 0, totalT, lty=3, add=TRUE)
mtext(expression("S(0) = 1"))
legend("topleft", legend = expression(lambda[r] == 1/2, lambda[r] == 1,
       lambda[r] == 2), lty = 1:lr)
curve(dtm(x, totalT, 1, 1/2, "r"), 0, totalT, lty=1, ylim=c(0, 0.34),
      xlab="M(10)", ylab="density")
curve(dtm(x, totalT, 1, 1, "r"), 0, totalT, lty=2, add=TRUE)
curve(dtm(x, totalT, 1, 2, "r"), 0, totalT, lty=3, add=TRUE)
mtext(expression("S(0) = 0"))
legend("topleft", legend = expression(lambda[r] == 1/2, lambda[r] == 1,
      lambda[r] == 2), lty = 1:lr)
par(old.par)
```

`estimate` *Estimate Result of smam Estimators*

Description

‘estimate‘ function returns the estimate result of ‘smam::fitXXXX‘ from smam package.

Usage

```
estimate(x, ...)

## S3 method for class 'smam_mrme'
estimate(x, ...)

## S3 method for class 'smam_mr'
estimate(x, ...)

## S3 method for class 'smam_mm'
estimate(x, ...)

## S3 method for class 'smam_mrh'
estimate(x, ...)

## S3 method for class 'smam_bmme'
estimate(x, ...)
```

Arguments

`x`	a fitted object from one of ‘smam::fitXXXX‘ functions
`...`	other arguments

Examples

```
## time consuming example
#tgrid <- seq(0, 100, length=100)
#set.seed(123)
#dat <- rMRME(tgrid, 1, 0.5, 1, 0.01, "m")

## fit whole dataset to the MRME model
#fit <- fitMRME(dat, start=c(1, 0.5, 1, 0.01))
#fit

## get covariance matrix of estimators
#estimate(fit)
```

`estVarMRME_Godambe` *Variance matrix of estimators from moving-resting process with measurement error*

Description

’estVarMRME_Godambe’ uses Godambe information matrix to obtain variance matrix of estimators from ’fitMRME’. ’estVarMRME_pBootstrap’ uses parametric bootstrap to obtain variance matrix of estimators from ’fitMRME’. ’estVarMRMEnaive_Godambe’ use Godambe information matrix to obtain variance matrix of estimators from ’fitMRME_naive’. ’estVarMRMEnaive_pBootstrap’ uses parametric bootstrap to obtain variance matrix of estimators from ’fitMRME_naive’.

Usage

```
estVarMRME_Godambe(
  est_theta,
  data,
  nBS,
  numThreads = 1,
  gradMethod = "simple",
  integrControl = integr.control()
)

estVarMRME_pBootstrap(
  est_theta,
  data,
  nBS,
  detailBS = FALSE,
  numThreads = 1,
  integrControl = integr.control()
)

estVarMRMEnaive_Godambe(
  est_theta,
  data,
  nBS,
  numThreads = 1,
  gradMethod = "simple",
  integrControl = integr.control()
)

estVarMRMEnaive_pBootstrap(
  est_theta,
  data,
  nBS,
  detailBS = FALSE,
  numThreads = 1,
  integrControl = integr.control()
)
```

Arguments

`est_theta`	estimators of MRME model
`data`	data used to process estimation
`nBS`	number of bootstrap.
`numThreads`	the number of threads for parallel computation. If its value is greater than 1, then parallel computation will be processed. Otherwise, serial computation will be processed.
`gradMethod`	method used for numeric gradient (`numDeriv::grad`).

`integrControl`	a list of control parameters for the `integrate` function: rel.tol, abs.tol, subdivision.
`detailBS`	whether or not output estimation results during bootstrap, which can be used to generate bootstrap CI.

Value

variance-covariance matrix of estimators

Author(s)

Chaoran Hu

Examples

```
## Not run:
## time consuming example
tgrid <- seq(0, 10*100, length=100)
set.seed(123)
dat <- rMRME(tgrid, 1, 0.5, 1, 0.01, "m")

estVarMRME_Godambe(c(1, 0.5, 1, 0.01), dat, nBS = 10)
estVarMRME_pBootstrap(c(1, 0.5, 1, 0.01), dat, nBS = 10)
estVarMRMEnaive_Godambe(c(1, 0.5, 1, 0.01), dat, nBS = 10)
estVarMRMEnaive_pBootstrap(c(1, 0.5, 1, 0.01), dat, nBS = 10)

estVarMRME_Godambe(c(1, 0.5, 1, 0.01), dat, nBS = 10, numThreads = 6)
estVarMRME_pBootstrap(c(1, 0.5, 1, 0.01), dat, nBS = 10, numThreads = 6)
estVarMRMEnaive_Godambe(c(1, 0.5, 1, 0.01), dat, nBS = 10, numThreads = 6)
estVarMRMEnaive_pBootstrap(c(1, 0.5, 1, 0.01), dat, nBS = 10, numThreads = 6)
estVarMRMEnaive_pBootstrap(c(1, 0.5, 1, 0.01), dat, nBS = 10, numThreads = 6)

## End(Not run)
```

`f109` *GPS data of f109*

Description

A dataset of GPS coordinates of a mature female mountain lion living in the Gros Ventre Mountain Range near Jackson, Wyoming. The data were collected by a code-only GPS wildlife tracking collar from 2009 to 2012.

Usage

```
f109
```

Format

A data frame with 3919 rows and 4 variables:

date Date when the GPS coordinates were collected

cumTime Standardized time when the GPS coordinates were collected (unit: hour)

centerE Standardized UTM easting (unit: km)

centerN Standardized UTM northing (unit: km)

`f109raw` *GPS data of f109 (raw format)*

Description

The original format of ‘f109‘ dataset.

Usage

```
f109raw
```

Format

A data frame with 3917 rows and 3 variables:

t Date and time when the GPS coordinates were collected (unit: year)

dE Standardized UTM easting (unit: meter)

dN Standardized UTM northing (unit: meter)

`fitBMME` *Fit a Brownian Motion with Measurement Error*

Description

Given discretely observed animal movement locations, fit a Brownian motion model with measurement errors. Using `segment` to fit part of observations to the model. A practical application of this feature is seasonal analysis.

Usage

```
fitBMME(
  data,
  start = NULL,
  segment = NULL,
  method = "Nelder-Mead",
  optim.control = list()
)

fitBmme(data, start = NULL, method = "Nelder-Mead", optim.control = list())
```

Arguments

`data`	a data.frame whose first column is the observation time, and other columns are location coordinates. If `segment` is not `NULL`, additional column with the same name given by `segment` should be included. This additional column is used to indicate which part of observations shoule be used to fit model. The value of this column can be any integer with 0 means discarding this observation and non-0 means using this obversvation. Using different non-zero numbers indicate different segments. (See vignette for more details.)
`start`	starting value of the model, a vector of two component, one for sigma (sd of BM) and the other for delta (sd for measurement error). If unspecified (NULL), a moment estimator will be used assuming equal sigma and delta.
`segment`	character variable, name of the column which indicates segments, in the given `data.frame`. The default value, `NULL`, means using whole dataset to fit the model.
`method`	the method argument to feed `optim`.
`optim.control`	a list of control that is passed down to `optim`.

Details

The joint density of the increment data is multivariate normal with a sparse (tri-diagonal) covariance matrix. Sparse matrix operation from package Matrix is used for computing efficiency in handling large data.

Value

A list of the following components:

`estimate`	the estimated parameter vector
`var.est`	variance matrix of the estimator
`loglik`	loglikelihood evaluated at the estimate
`convergence`	convergence code from optim

References

Pozdnyakov V., Meyer, TH., Wang, Y., and Yan, J. (2013) On modeling animal movements using Brownian motion with measurement error. Ecology 95(2): p247–253. doi:doi:10.1890/13-0532.1.

See Also

`fitMR`

Examples

```
set.seed(123)
tgrid <- seq(0, 500, by = 1)
dat <- rBMME(tgrid, sigma = 1, delta = 0.5)

## using whole dataset to fit BMME
```

```
fit <- fitBMME(dat)
fit

## using part of dataset to fit BMME
batch <- c(rep(0, 100), rep(1, 200), rep(0, 50), rep(2, 100), rep(0, 51))
dat.segment <- cbind(dat, batch)
fit.segment <- fitBMME(dat.segment, segment = "batch")
head(dat.segment)
fit.segment
```

`fitMM` *Fit a Moving-Moving Model with 2 Embedded Brownian Motion*

Description

Fit a Moving-Moving Model with 2 Embedded Brownian Motion with animal movement data at discretely observation times by maximizing a full likelihood constructed from the marginal density of increment. 'estVarMM' uses parametric bootstrap to obtain variance matrix of estimators from 'fitMM'.

Usage

```
fitMM(
  data,
  start,
  logtr = FALSE,
  method = "Nelder-Mead",
  optim.control = list(),
  integrControl = integr.control()
)

estVarMM(
  est_theta,
  data,
  nBS,
  detailBS = FALSE,
  numThreads = 1,
  integrControl = integr.control()
)
```

Arguments

`data`	data used to process estimation
`start`	starting value of the model, a vector of four components in the order of rate for moving1, rate for moving2, and volatility1(larger), volatility2(smaller).
`logtr`	logical, if TRUE parameters are estimated on the log scale.

`method`	the method argument to feed `optim`.
`optim.control`	a list of control to be passed to `optim`.
`integrControl`	a list of control parameters for the `integrate` function: rel.tol, abs.tol, subdivision.
`est_theta`	estimators of MRME model
`nBS`	number of bootstrap.
`detailBS`	whether or not output estimation results during bootstrap, which can be used to generate bootstrap CI.
`numThreads`	the number of threads for parallel computation. If its value is greater than 1, then parallel computation will be processed. Otherwise, serial computation will be processed.

Value

a list of the following components:

`estimate`	the esimated parameter vector
`loglik`	maximized loglikelihood or composite loglikelihood evaluated at the estimate
`convergence`	convergence code from `optim`

References

Yan, J., Chen, Y., Lawrence-Apfel, K., Ortega, I. M., Pozdnyakov, V., Williams, S., and Meyer, T. (2014) A moving-resting process with an embedded Brownian motion for animal movements. Population Ecology. 56(2): 401–415.

Pozdnyakov, V., Elbroch, L., Labarga, A., Meyer, T., and Yan, J. (2017) Discretely observed Brownian motion governed by telegraph process: estimation. Methodology and Computing in Applied Probability. doi:10.1007/s11009-017-9547-6.

Examples

```
## Not run:
## time consuming example
tgrid <- seq(0, 100, length=100)
set.seed(123)
dat <- rMM(tgrid, 1, 0.1, 1, 0.1, "m1")

## fit whole dataset to the MR model
fit <- fitMM(dat, start=c(1, 0.1, 1, 0.1))
fit

var <- estVarMM(fit$estimate, dat, nBS = 10, numThreads = 6)
var

## End(Not run)
```

`fitMR` *Fit a Moving-Resting Model with Embedded Brownian Motion*

Description

Fit a Moving-Resting Model with Embedded Brownian Motion with animal movement data at discretely observation times by maximizing a composite likelihood constructed from the marginal density of increment. Using `segment` to fit part of observations to the model. A practical application of this feature is seasonal analysis.

Usage

```
fitMR(
  data,
  start,
  segment = NULL,
  likelihood = c("full", "composite"),
  logtr = FALSE,
  method = "Nelder-Mead",
  optim.control = list(),
  integrControl = integr.control()
)

fitMovRes(
  data,
  start,
  likelihood = c("full", "composite"),
  logtr = FALSE,
  method = "Nelder-Mead",
  optim.control = list(),
  integrControl = integr.control()
)
```

Arguments

`data`	a data.frame whose first column is the observation time, and other columns are location coordinates. If `segment` is not `NULL`, additional column with the same name given by `segment` should be included. This additional column is used to indicate which part of observations shoule be used to fit model. The value of this column can be any integer with 0 means discarding this observation and non-0 means using this obversvation. Using different non-zero numbers indicate different segments. (See vignette for more details.)
`start`	starting value of the model, a vector of three components in the order of rate for moving, rate for resting, and volatility.
`segment`	character variable, name of the column which indicates segments, in the given `data.frame`. The default value, `NULL`, means using whole dataset to fit the model.

`likelihood`	a character string specifying the likelihood type to maximize in estimation. This can be "full" for full likelihood or "composite’ for composite likelihood.
`logtr`	logical, if TRUE parameters are estimated on the log scale.
`method`	the method argument to feed `optim`.
`optim.control`	a list of control to be passed to `optim`.
`integrControl`	a list of control parameters for the `integrate` function: rel.tol, abs.tol, subdivision.

Value

a list of the following components:

`estimate`	the esimated parameter vector
`loglik`	maximized loglikelihood or composite loglikelihood evaluated at the estimate
`convergence`	convergence code from `optim`
`likelihood`	likelihood type (full or composite) from the input

References

Yan, J., Chen, Y., Lawrence-Apfel, K., Ortega, I. M., Pozdnyakov, V., Williams, S., and Meyer, T. (2014) A moving-resting process with an embedded Brownian motion for animal movements. Population Ecology. 56(2): 401–415.

Pozdnyakov, V., Elbroch, L., Labarga, A., Meyer, T., and Yan, J. (2017) Discretely observed Brownian motion governed by telegraph process: estimation. Methodology and Computing in Applied Probability. doi:10.1007/s11009-017-9547-6.

Examples

```
## Not run:
## time consuming example
tgrid <- seq(0, 10, length=500)
set.seed(123)
## make it irregularly spaced
tgrid <- sort(sample(tgrid, 30)) # change to 400 for a larger sample
dat <- rMR(tgrid, 1, 2, 25, ”m”)

## fit whole dataset to the MR model
fit.fl <- fitMR(dat, start=c(2, 2, 20), likelihood = ”full”)
fit.fl

fit.cl <- fitMR(dat, start=c(2, 2, 20), likelihood = ”composite”)
fit.cl

## fit part of dataset to the MR model
batch <- c(rep(0, 5), rep(1, 7), rep(0, 4), rep(2, 10), rep(0, 4))
dat.segment <- cbind(dat, batch)
fit.segment <- fitMR(dat.segment, start = c(2, 2, 20), segment = ”batch”,
                     likelihood = ”full”)
head(dat.segment)
```

```
fit.segment

## End(Not run)
```

`fitMRH` *Fit a Moving-Resting-Handling Model with Embedded Brownian Motion*

Description

Fit a Moving-Resting-Handling Model with Embedded Brownian Motion with animal movement data at discretely observation times by maximizing a full likelihood. Using `segment` to fit part of observations to the model. A practical application of this feature is seasonal analysis.

Usage

```
fitMRH(
  data,
  start,
  segment = NULL,
  numThreads = RcppParallel::defaultNumThreads() * 3/4,
  lower = c(0.001, 0.001, 0.001, 0.001, 0.001),
  upper = c(10, 10, 10, 10, 0.999),
  integrControl = integr.control(),
  print_level = 3
)
```

Arguments

`data`	a data.frame whose first column is the observation time, and other columns are location coordinates. If `segment` is not `NULL`, additional column with the same name given by `segment` should be included. This additional column is used to indicate which part of observations shoule be used to fit model. The value of this column can be any integer with 0 means discarding this observation and non-0 means using this obversvation. Using different non-zero numbers indicate different segments. (See vignette for more details.)
`start`	The initial value for optimization, in the order of rate of moving, rate of resting, rate of handling, volatility and switching probability.
`segment`	character variable, name of the column which indicates segments, in the given `data.frame`. The default value, `NULL`, means using whole dataset to fit the model.
`numThreads`	int, the number of threads allocated for parallel computation. The default setup is 3/4 available threads. If this parameter is less or equal to 1, the serial computation will be processed.
`lower`, `upper`	Lower and upper bound for optimization.
`integrControl`	Integration control vector includes rel.tol, abs.tol, and subdivisions.

`print_level`	print_level passed to nloptr::nloptr. Possible values: 0 (default): no output; 1: show iteration number and value of objective function; 2: 1 + show value of (in)equalities; 3: 2 + show value of controls.

Value

A list of estimation result with following components:

`estimate`	the estimated parameter vector
`loglik`	maximized loglikelihood or composite loglikelihood evaluated at the estimate
`convergence`	convergence code from `nloptr`
`data`	fitted data

Author(s)

Chaoran Hu

References

Pozdnyakov, V., Elbroch, L.M., Hu, C. et al. On Estimation for Brownian Motion Governed by Telegraph Process with Multiple Off States. Methodol Comput Appl Probab 22, 1275–1291 (2020). doi:10.1007/s11009-020-09774-1

See Also

`rMRH` for simulation.

Examples

```
## Not run:
## time consuming example
set.seed(06269)
tgrid <- seq(0, 400, by = 8)
dat <- rMRH(tgrid, 4, 0.5, 0.1, 5, 0.8, 'm')
fitMRH(dat, c(4, 0.5, 0.1, 5, 0.8)) ## parallel process
fitMRH(dat, c(4, 0.5, 0.1, 5, 0.8), numThreads = -1) ## serial process

## fit part of dataset to the MRH model
batch <- c(rep(0, 10), rep(1, 7), rep(0, 10), rep(2, 10), rep(0, 14))
dat.segment <- cbind(dat, batch)
fit.segment <- fitMRH(dat.segment, start = c(4, 0.5, 0.1, 5, 0.8), segment = "batch")
head(dat.segment)
fit.segment

## End(Not run)
```

`fitMRME` *Fit a Moving-Resting Model with Measurement Error*

Description

’fitMRME’ fits a Moving-Resting Model with Measurement Error. The measurement error is modeled by Guassian noise. Using `segment` to fit part of observations to the model. A practical application of this feature is seasonal analysis.

Usage

```
fitMRME(
  data,
  start,
  segment = NULL,
  lower = c(1e-06, 1e-06, 1e-06, 1e-06),
  upper = c(10, 10, 10, 10),
  print_level = 3,
  integrControl = integr.control()
)

fitMRME_naive(
  data,
  start,
  segment = NULL,
  lower = c(1e-06, 1e-06, 1e-06, 1e-06),
  upper = c(10, 10, 10, 10),
  integrControl = integr.control()
)

fitMRMEapprox(
  data,
  start,
  segment = NULL,
  approx_norm_even = approxNormalOrder(5),
  approx_norm_odd = approxNormalOrder(6),
  method = "Nelder-Mead",
  optim.control = list(),
  integrControl = integr.control()
)
```

Arguments

`data`	a data.frame whose first column is the observation time, and other columns are location coordinates. If `segment` is not `NULL`, additional column with the same name given by `segment` should be included. This additional column is used to indicate which part of observations shoule be used to fit model. The value

	of this column can be any integer with 0 means discarding this observation and non-0 means using this obversvation. Using different non-zero numbers indicate different segments. (See vignette for more details.)
`start`	starting value of the model, a vector of four components in the order of rate for moving, rate for resting, volatility, and s.d. of Guassian measurement error.
`segment`	character variable, name of the column which indicates segments, in the given `data.frame`. The default value, `NULL`, means using whole dataset to fit the model.
`lower`, `upper`	Lower and upper bound for optimization.
`print_level`	print_level passed to nloptr::nloptr. Possible values: 0 (default): no output; 1: show iteration number and value of objective function; 2: 1 + show value of (in)equalities; 3: 2 + show value of controls.
`integrControl`	a list of control parameters for the `integrate` function: rel.tol, abs.tol, subdivision.
`approx_norm_even`, `approx_norm_odd`	numeric matrixes specify the discrete distributions used to approximate standard normal distribution. The first column is support of discrete distribution and the second column is probability mass function. `approx_norm_even` is used to approximate even step error and `approx_norm_odd` is used to approximate odd step error. We mention that the supports of these two discrete distributions should not have any common elements.
`method`	the method argument to feed `optim`.
`optim.control`	a list of control to be passed to `optim`.

Value

a list of the following components:

`estimate`	the esimated parameter vector
`loglik`	maximized loglikelihood or composite loglikelihood evaluated at the estimate
`convergence`	convergence code from `optim`
`data`	fitted data

Author(s)

Chaoran Hu

References

Hu, C., Elbroch, L.M., Meyer, T., Pozdnyakov, V. and Yan, J. (2021), Moving-resting process with measurement error in animal movement modeling. Methods in Ecology and Evolution. doi:10.1111/2041-210X.13694

Examples

```
## time consuming example
#tgrid <- seq(0, 10*100, length=100)
#set.seed(123)
#dat <- rMRME(tgrid, 1, 0.5, 1, 0.01, "m")

## fit whole dataset to the MRME model
#fit <- fitMRME(dat, start=c(1, 0.5, 1, 0.01))
#fit

## fit whole dataset to the MRME model with naive composite likelihood
#fit.naive <- fitMRME_naive(dat, start=c(1, 0.5, 1, 0.01))
#fit.naive

## fit whole dataset to the MRME model with approximate error
#fit.approx <- fitMRMEapprox(dat, start=c(1, 0.5, 1, 0.01))
#fit.approx

## fit part of dataset to the MR model
#batch <- c(rep(0, 5), rep(1, 17), rep(0, 4), rep(2, 30), rep(0, 4), rep(3, 40))
#dat.segment <- cbind(dat, batch)
#fit.segment <- fitMRME(dat.segment, start = c(1, 0.5, 1, 0.01), segment = "batch")
#fit.segment.approx <- fitMRMEapprox(dat.segment, start = c(1, 0.5, 1, 0.01), segment = "batch")
#head(dat.segment)
#fit.segment
```

`fitStateMR` *Estimation of states at each time point with Moving-Resting Process*

Description

Estimate the state at each time point under the Moving-Resting process with Embedded Brownian Motion with animal movement data at discretely time points. See the difference between `fitStateMR` and `fitViterbiMR` in detail part. Using `fitPartialViterbiMR` to estimate the state within a small piece of time interval.

Usage

```
fitStateMR(data, theta, cutoff = 0.5, integrControl = integr.control())

fitViterbiMR(data, theta, cutoff = 0.5, integrControl = integr.control())

fitPartialViterbiMR(
  data,
  theta,
  cutoff = 0.5,
  startpoint,
```

```
  pathlength,
  integrControl = integr.control()
)
```

Arguments

`data`	a `data.frame` whose first column is the observation time, and other columns are location coordinates.
`theta`	the parameters for Moving-Resting model, in the order of rate of moving, rate of resting, volatility.
`cutoff`	the cut-off point for prediction.
`integrControl`	Integration control vector includes rel.tol, abs.tol, and subdivisions.
`startpoint`	Start time point of interested time interval.
`pathlength`	the length of interested time interval.

Details

`fitStateMR` estimates the most likely state by maximizing the probability of $Pr(S(t = t_k) = s_k|X)$, where X is the whole data and s_k is the possible sates at t_k (moving, resting).

`fitViterbiMR` estimates the most likely state path by maximizing $Pr(S(t = t_0) = s_0, S(t = t_1) = s_1, ..., S(t = t_n) = s_n|X)$, where X is the whole data and $s_0, s_1, ..., s_n$ is the possible state path.

`fitPartialViterbiMR` estimates the most likely state path of a small peice of time interval, by maximizing the probability of $Pr(S(t = t_k) = s_k, ..., S(t = t_{k+q-1}) = s_{k+q-1}|X)$, where k is the start time point and q is the length of interested time interval.

Value

A `data.frame` contains estimated results, with elements:

- original data be estimated.
- conditional probability of moving, resting (`p.m`, `p.r`), which is $Pr(S(t = t_k) = s_k|X)$ for `fitStateMR`; $log - Pr(s_0, ..., s_k|X_k)$ for `fitViterbiMR`, where X_k is $(X_0, ..., X_k)$; and $log - Pr(s_k, ..., s_{k+q-1}|X)$ for `fitPartialViterbiMR`.
- estimated states with 1-moving, 0-resting.

Author(s)

Chaoran Hu

See Also

`rMR` for simulation. `fitMR` for estimation of parameters.

Examples

```
set.seed(06269)
tgrid <- seq(0, 400, by = 8)
dat <- rMR(tgrid, 4, 3.8, 5, 'm')
fitStateMR(dat, c(4, 3.8, 5), cutoff = 0.5)
fitViterbiMR(dat, c(4, 3.8, 5), cutoff = 0.5)
fitPartialViterbiMR(dat, c(4, 3.8, 5), cutoff = 0.5, 20, 10)
```

`fitStateMRH` *Estimation of states at each time point with Moving-Resting-Handling Process*

Description

Estimate the state at each time point under the Moving-Resting-Handling process with Embedded Brownian Motion with animal movement data at discretely time points. See the difference between `fitStateMRH` and `fitViterbiMRH` in detail part. Using `fitPartialViterbiMRH` to estimate the state during a small piece of time interval.

Usage

```
fitStateMRH(data, theta, integrControl = integr.control())

fitViterbiMRH(data, theta, integrControl = integr.control())

fitPartialViterbiMRH(
  data,
  theta,
  startpoint,
  pathlength,
  integrControl = integr.control()
)
```

Arguments

`data`	a `data.frame` whose first column is the observation time, and other columns are location coordinates.
`theta`	the parameters for Moving-Resting-Handling model, in the order of rate of moving, rate of resting, rate of handling, volatility and switching probability.
`integrControl`	Integration control vector includes rel.tol, abs.tol, and subdivisions.
`startpoint`	Start time point of interested time interval.
`pathlength`	the length of interested time interval.

Details

`fitStateMRH` estimates the most likely state by maximizing the probability of $Pr(S(t = t_k) = s_k|X)$, where X is the whole data and s_k is the possible sates at t_k (moving, resting or handling).

`fitViterbiMRH` estimates the most likely state path by maximizing $Pr(S(t = t_0) = s_0, S(t = t_1) = s_1, ..., S(t = t_n) = s_n|X)$, where X is the whole data and $s_0, s_1, ..., s_n$ is the possible state path.

`fitPartialViterbiMRH` estimates the most likely state path of a small peice of time interval, by maximizing the probability of $Pr(S(t = t_k) = s_k, ..., S(t = t_{k+q-1}) = s_{k+q-1}|X)$, where k is the start time point and q is the length of interested time interval.

Value

A `data.frame` contains estimated results, with elements:

- original data be estimated.
- conditional probability of moving, resting, handling (`p.m`, `p.r`, `p.h`), which is $Pr(S(t = t_k) = s_k|X)$ for `fitStateMRH`; $log - Pr(s_0, ..., s_k|X_k)$ for `fitViterbiMRH`, where X_k is $(X_0, ..., X_k)$; and $log - Pr(s_k, ..., s_{k+q-1}|X)$ for `fitPartialViterbiMRH`.
- estimated states with 0-moving, 1-resting, 2-handling.

Author(s)

Chaoran Hu

References

Pozdnyakov, V., Elbroch, L.M., Hu, C., Meyer, T., and Yan, J. (2018+) On estimation for Brownian motion governed by telegraph process with multiple off states. <arXiv:1806.00849>

See Also

`rMRH` for simulation. `fitMRH` for estimation of parameters.

Examples

```
## Not run:
## time consuming example
set.seed(06269)
tgrid <- seq(0, 400, by = 8)
dat <- rMRH(tgrid, 4, 0.5, 0.1, 5, 0.8, 'm')
fitStateMRH(dat, c(4, 0.5, 0.1, 5, 0.8))
fitViterbiMRH(dat, c(4, 0.5, 0.1, 5, 0.8))
fitPartialViterbiMRH(dat, c(4, 0.5, 0.1, 5, 0.8), 20, 10)

## End(Not run)
```

`integr.control` *Auxiliary for Controlling Numerical Integration*

Description

Auxiliary function for the numerical integration used in the likelihood and composite likelihood functions. Typically only used internally by 'fitMR' and 'fitMRH'.

Usage

```
integr.control(
  rel.tol = .Machine$double.eps^0.25,
  abs.tol = rel.tol,
  subdivisions = 100L
)
```

Arguments

`rel.tol`	relative accuracy requested.
`abs.tol`	absolute accuracy requested.
`subdivisions`	the maximum number of subintervals.

Details

The arguments are the same as `integrate`, but passed down to the C API of Rdqags used by `integrate`.

Value

A list with components named as the arguments.

`rBB` *Sampling from a Brownian bridge path give a grid time*

Description

A Brownian bridge is a continuous-time stochastic process B(t) whose probability distribution is the conditional probability distribution of a standard Wiener process W(t) subject to the condition (when standardized) that W(T) = 0, so that the process is pinned to the same value at both t = 0 and t = T. The implementation here is a generalized Brownian bridge that allows start point and end point at different locations.

Usage

```
rBB(time, start_pt, end_pt, sigma)
```

Arguments

`time`	time points at which observations are to be simulated
`start_pt`	the start point location of Brownian bridge
`end_pt`	the end point location of Brownian brige
`sigma`	volatility parameter of the Brownian motion

Value

A `data.frame` whose first column is the time points and second column is coordinate of the locations.

Examples

```
## Brownian bridge starting from location 0 and ending at location 1
## with sigma 0.1 from time 0 to time 10
plot(rBB(seq(0, 10, length.out = 100), 0, 1, 0.1), type = ”l”)
```

`rBMME` *Sampling from Brown Motion with Measurement Error*

Description

Given the volatility parameters of a Brownian motion and normally distributed measurement errors, generate the process at discretely observed time points of a given dimension.

Usage

```
rBMME(time, dim = 2, sigma = 1, delta = 1)

rBmme(time, dim = 2, sigma = 1, delta = 1)
```

Arguments

`time`	vector of time points at which observations are to be sampled
`dim`	(integer) dimension of the Brownian motion
`sigma`	volatility parameter (sd) of the Brownian motion
`delta`	sd parameter of measurement error

Value

A `data.frame` whose first column is the time points and whose other columns are coordinates of the locations.

References

Pozdnyakov V., Meyer, TH., Wang, Y., and Yan, J. (2013) On modeling animal movements using Brownian motion with measurement error. Ecology 95(2): p247–253. doi:doi:10.1890/13-0532.1.

Examples

```
tgrid <- seq(0, 10, length = 1001)
## make it irregularly spaced
tgrid <- sort(sample(tgrid, 800))
dat <- rBMME(tgrid, 1, 1)
plot(dat[,1], dat[,2], xlab="t", ylab="X(t)", type="l")
```

`rMM` *Sampling from a Moving-Moving Process with 2 Embedded Brownian Motion*

Description

A moving-moving process consists of two states: moving (large) and moving (small). The transition between the two states is modeled by an alternating renewal process, with exponentially distributed duration. An animal moves according to two Brownian motions with different volatility parameters.

Usage

```
rMM(time, lamM1, lamM2, sigma1, sigma2, s0, dim = 2)
```

Arguments

`time`	time points at which observations are to be simulated
`lamM1`	rate parameter of the exponential duration while moving1
`lamM2`	rate parameter of the exponential duration while moving2
`sigma1`	volatility parameter of the Brownian motion while moving1
`sigma2`	volatility parameter of the Brownian motion while moving2
`s0`	the state at time 0, must be one of "m1" or "m2", for moving1 and moving2, respectively
`dim`	(integer) dimension of the Brownian motion

Value

A `data.frame` whose first column is the time points and whose other columns are coordinates of the locations.

References

Yan, J., Chen, Y., Lawrence-Apfel, K., Ortega, I. M., Pozdnyakov, V., Williams, S., and Meyer, T. (2014) A moving-resting process with an embedded Brownian motion for animal movements. Population Ecology. 56(2): 401–415.

Pozdnyakov, V., Elbroch, L., Labarga, A., Meyer, T., and Yan, J. (2017) Discretely observed Brownian motion governed by telegraph process: estimation. Methodology and Computing in Applied Probability. doi:10.1007/s11009-017-9547-6.

Examples

```
tgrid <- seq(0, 100, length=100)

dat <- rMM(tgrid, 1, 0.1, 1, 0.1, "m1")
plot(dat[,1], dat[,2], xlab="t", ylab="X(t)", type='l')
```

`rMR` *Sampling from a Moving-Resting Process with Embedded Brownian Motion*

Description

A moving-resting process consists of two states: moving and resting. The transition between the two states is modeled by an alternating renewal process, with exponentially distributed duration. An animal stays at the same location while resting, and moves according to a Brownian motion while moving.

Usage

```
rMR(time, lamM, lamR, sigma, s0, dim = 2, state = FALSE)

rMovRes(time, lamM, lamR, sigma, s0, dim = 2)

rMRME(time, lamM, lamR, sigma, sig_err, s0, dim = 2, state = FALSE)
```

Arguments

`time`	time points at which observations are to be simulated
`lamM`	rate parameter of the exponential duration while moving
`lamR`	rate parameter of the exponential duration while resting
`sigma`	volatility parameter of the Brownian motion while moving
`s0`	the state at time 0, must be one of "m" or "r", for moving and resting, respectively
`dim`	(integer) dimension of the Brownian motion
`state`	indicates whether the simulation show the states at given time points.
`sig_err`	s.d. of Gaussian white noise

Value

A `data.frame` whose first column is the time points and whose other columns are coordinates of the locations.

References

Yan, J., Chen, Y., Lawrence-Apfel, K., Ortega, I. M., Pozdnyakov, V., Williams, S., and Meyer, T. (2014) A moving-resting process with an embedded Brownian motion for animal movements. Population Ecology. 56(2): 401–415.

Pozdnyakov, V., Elbroch, L., Labarga, A., Meyer, T., and Yan, J. (2017) Discretely observed Brownian motion governed by telegraph process: estimation. Methodology and Computing in Applied Probability. doi:10.1007/s11009-017-9547-6.

Examples

```
tgrid <- seq(0, 10, length=1001)
## make it irregularly spaced
tgrid <- sort(sample(tgrid, 800))
dat <- rMR(tgrid, 1, 1, 1, "m")
plot(dat[,1], dat[,2], xlab="t", ylab="X(t)", type='l')

dat2 <- rMR(tgrid, 1, 1, 1, "m", state = TRUE)
head(dat2)

dat3 <- rMRME(tgrid, 1, 1, 1, 0.01, "m", state = TRUE)
head(dat3)
plot(dat3[,1], dat3[,3], xlab="t", ylab="Z(t)=X(t)+GWN(0.01)", type="l")
```

---

rMRB *Sampling from a Moving-Resting bridge*

Description

A moving-resting process consists of two states: moving and resting. The transition between the two states is modeled by an alternating renewal process, with exponentially distributed duration. An animal stays at the same location while resting, and moves according to a Brownian motion while moving. A moving-resting bridge is an extension of Brownian bridge that uses moving-resting process to bridge given starting point and ending point.

Usage

```
rMRB(time, start_pt, end_pt, lamM, lamR, sigma, s0)
```

Arguments

`time`	time points at which observations are to be simulated
`start_pt`	the start point location of Brownian bridge (numeric for one dimension, vector for multiple dimension)
`end_pt`	the end point location of Brownian brige (numeric for one dimension, vector for multiple dimension)
`lamM`	rate parameter of the exponential duration while moving
`lamR`	rate parameter of the exponential duration while resting
`sigma`	volatility parameter of the Brownian motion while moving
`s0`	the state at time 0, must be one of "m" or "r", for moving and resting, respectively

Value

A `data.frame` whose first column is the time points and whose other columns are coordinates of the locations.

Examples

```
tgrid <- seq(0, 10, length=1001)
## make it irregularly spaced
tgrid <- sort(sample(tgrid, 800))

## a 2-dim moving-resting bridge starting from c(0, 0)
## and ending at c(10, -10)
dat <- rMRB(tgrid, start_pt = c(0, 0), end_pt = c(10, -10),
            lamM = 1, lamR = 1, sigma = 1, "m")
par(mfrow = c(2, 1))
plot(dat[,1], dat[,2], xlab="t", ylab="X(t)", type='l')
plot(dat[,1], dat[,3], xlab="t", ylab="X(t)", type='l')
par(mfrow = c(1, 1))
```

`rMRH` *Sampling from a Moving-Resting-Handling Process with Embedded Brownian Motion*

Description

A moving-resting-handling process consists of three states: moving, resting and handling. The transition between the three states is modeled by an alternating renewal process, with expenentially distributed duration. An animal stays at the same location while resting and handling (the choice of resting and handling depends on Bernoulli distribution), and moves according to a Brownian motion while moving state. The sequence of states is moving, resting or staying, moving, resting or staying ... or versus

Usage

```
rMRH(time, lamM, lamR, lamH, sigma, p, s0, dim = 2, state = FALSE)
```

Arguments

`time`	time points at which observations are to be simulated
`lamM`	rate parameter of the exponential duration while moving
`lamR`	rate parameter of the exponential duration while resting
`lamH`	rate parameter of the exponential duration while handling
`sigma`	volatility parameter of the Brownian motion while moving
`p`	probability of choosing resting, and 1-p is probability of choosing handling
`s0`	the state at time 0, must be one of "m" (moving), "r" (resting) or "h" (handling).
`dim`	(integer) dimension of the Brownian motion
`state`	indicates whether the simulation show the states at given time points.

Value

A `data.frame` whose first column is the time points and whose other columns are coordinates of the locations. If `state` is `TRUE`, the second column will be the simulation state.

Author(s)

Chaoran Hu

References

Pozdnyakov, V., Elbroch, L.M., Hu, C., Meyer, T., and Yan, J. (2018+) On estimation for Brownian motion governed by telegraph process with multiple off states. <arXiv:1806.00849>

See Also

`fitMRH` for fitting model.

Examples

```
set.seed(06269)
tgrid <- seq(0, 8000, length.out=1001)
dat <- rMRH(time=tgrid, lamM=4, lamR=0.04, lamH=0.2,
            sigma=1000, p=0.5, s0="m", dim=2)
plot(dat$time, dat$X1, type='l')
plot(dat$time, dat$X2, type='l')
plot(dat$X1,   dat$X2, type='l')

set.seed(06269) ## show the usage of state
dat2 <- rMRH(time=tgrid, lamM=4, lamR=0.04, lamH=0.2,
             sigma=1000, p=0.5, s0="m", dim=2, state=TRUE)
head(dat)
head(dat2)
```

`seasonFilter` *Subsetting data during given season for each year (seasonal analysis toolbox)*

Description

Return subsets of data from each year, which is in given time interval between `startDate` and `endDate`.

Usage

```
seasonFilter(data, startDate, endDate)
```

Arguments

`data`	The data be filtered, which has the same format as the output from `transfData`.
`startDate, endDate`	Start point and end point of time interval during a year, which has the format "MM-DD".

Value

A `data.frame` with inputted data and additional column ’BATCH’ indicates which subset of inputted data is located within given time interval. In column ’BATCH’, different integers stands for different segments and 0 stands for outside given time interval.

Author(s)

Chaoran Hu

`smam` *smam: Statistical Modeling of Animal Movements*

Description

Animal movement models including Moving-Resting Process with Embedded Brownian Motion, Brownian Motion with Measurement Error, Moving-Resting-Handling Process with Embedded Brownian Motion, Moving-Resting Process with Measurement Error, Moving-Moving Process with two Embedded Brownian Motions.

Author(s)

- *maintainer, author* Chaoran Hu <`huchaoran.stat@gmail.com`>
- *author* Vladimir Pozdnyakov <`vladimir.pozdnyakov@uconn.edu`>
- *author* Jun Yan <`jun.yan@uconn.edu`>

`transfData` *Transfer raw dataset to the standard dataset (seasonal analysis toolbox)*

Description

Transfer the raw location dataset of animal to the standard dataset, which is acceptable in this packages. The raw dataset contains at least four components: 1. `t1`: data information. 2. `dt..hr.`: the difference of time between two sample points. 3. `e1`: the GPS coordinate of east-west. 4. `n1`: the GPS coordinate of north-south. (These weird variable names are from the original GPS data. We will change them in later version.)

Usage

```
transfData(data, dateFormat, roundValue = NULL, lengthUnit = "km")
```

Arguments

`data`	The raw dataset.
`dateFormat`	Charater string indicates the format of date variable.
`roundValue`	Round GPS coordinate to `roundValue` with unit meter. If NULL (default), no rounding will be processed.
`lengthUnit`	Charater string indicates the length unit of GPS coordinate, which can be "m" or "km"(default). Usually, we recommend not change the default setup of this parameter. Otherwise, numerical computation problem will happen.

Value

A `data.frame` containing the following components, which is standard format of dataset in this package:

- date: tells us the date of collecting this sample point.
- cumTime: cumulative time line. The collection of this data starts from time 0 in this time line. (Time unit is hours.)
- centerE: the centered east-west GPS coordinate with the center is the starting point (when `cumTime[1]`).
- centerN: the centered north-south GPS coordinate with the center is the starting point (when `cumTime[1]`).

Author(s)

Chaoran Hu

See Also

`as.Date` has parameter `format`, which is the same as the parameter `dateFormat` in this function.

---

vcov *Variance-Covariance Matrix of smam Estimators*

Description

This function calculates variance covariance matrix for estimators from smam package. Different methods will be used for different ‘smam‘ models.

Usage

```
## S3 method for class 'smam_mrme'
vcov(
  object,
  nBS = 25,
  detailBS = TRUE,
  numThreads = 5,
  gradMethod = "simple",
  vcovMethod = "pBootstrap",
  integrControl = integr.control(),
  ...
)

## S3 method for class 'smam_mm'
vcov(
  object,
  nBS = 25,
  detailBS = TRUE,
  numThreads = 5,
  integrControl = integr.control(),
  ...
)

## S3 method for class 'smam_mrh'
vcov(object, numThreads = 5, integrControl = integr.control(), ...)

## S3 method for class 'smam_mr'
vcov(object, ...)

## S3 method for class 'smam_bmme'
vcov(object, ...)
```

Arguments

`object`	a fitted object from one of ‘smam::fitXXXX‘ functions
`nBS`	number of bootstrap.
`detailBS`	whether or not output estimation results of bootstrap, which can be used to generate bootstrap CI. Required when ‘vcovMethod==’pBootstrap’‘.

`numThreads`	the number of threads for parallel computation. If its value is greater than 1, then parallel computation will be processed. Otherwise, serial computation will be processed.
`gradMethod`	method used for numeric gradient (`numDeriv::grad`). Required when ‘vcovMethod==’Godambe’‘.
`vcovMethod`	method of calculating variance covariance matrix. This should be one of ‘pBootstrap‘ (default) and ‘Godambe‘.
`integrControl`	a list of control parameters for the `integrate` function: rel.tol, abs.tol, subdivision.
`...`	Optional arguments that are not used

Examples

```
## time consuming example
#tgrid <- seq(0, 100, length=100)
#set.seed(123)
#dat <- rMRME(tgrid, 1, 0.5, 1, 0.01, "m")

## fit whole dataset to the MRME model
#fit <- fitMRME(dat, start=c(1, 0.5, 1, 0.01))
#fit

## get covariance matrix of estimators
#vcov(fit)
```

Index