Package ‘ctmm’

May 15, 2019

Encoding UTF-8

Version 0.5.6

Date 2019-05-14

Title Continuous-Time Movement Modeling

URL `https://github.com/ctmm-initiative/ctmm`,
`http://biology.umd.edu/movement.html`

Depends R (>= 3.5.0)

Imports data.table, expm, fasttime, Gmedian, graphics, grDevices, gsl, manipulate, MASS, methods, numDeriv, pbivnorm, pracma, raster, rgdal, shape, stats, utils

Suggests bit64, fftw, knitr, move, parallel, sp, suncalc

Description Functions for identifying, fitting, and applying continuous-space, continuous-time stochastic movement models to animal tracking data.
The package is described in Calabrese et al (2016) <doi:10.1111/2041-210X.12559> and its methods are based on those introduced in
Fleming & Calabrese et al (2014) <doi:10.1086/675504>,
Fleming et al (2014) <doi:10.1111/2041-210X.12176>,
Fleming et al (2015) <doi:10.1890/14-2010.1>,
Fleming et al (2016) <doi:10.1890/15-1607>,
Péron & Fleming et al (2016) <doi:10.1186/s40462-016-0084-7>,
Fleming & Calabrese (2016) <doi:10.1111/2041-210X.12673>,
Péron et al (2017) <doi:10.1002/ecm.1260>,
Fleming et al (2017) <doi:10.1016/j.ecoinf.2017.04.008>,
Fleming et al (2018) <doi:10.1002/eap.1704>,
and
Winner & Noonan et al (2018) <doi:10.1111/2041-210X.13027>.

License GPL-3

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 6.1.0

Author Christen H. Fleming [aut, cre],
Justin M. Calabrese [aut],
Xianghui Dong [ctb],
Kevin Winner [ctb],
Guillaume Péron [ctb],
Michael J. Noonan [ctb],
Bart Kranstauber [ctb],
Eliezer Gurarie [ctb],
Kamran Safi [ctb],
Paul C. Cross [dtc],
Thomas Mueller [dtc],
Rogério C. de Paula [dtc],
Thomas Akre [dtc],
Jonathan Drescher-Lehman [dtc]

Maintainer Christen H. Fleming <flemingc@si.edu>

Repository CRAN

Date/Publication 2019-05-15 11:40:07 UTC

R topics documented:

`ctmm-package` *Continuous-time movement modeling*

Description

Functions for identifying, fitting, and applying continuous-space, continuous-time stochastic movement models to animal tracking data. The package is described in Calabrese & Fleming (2016) <doi:10.1111/2041-210X.12559> and its methods are based on those introduced in Fleming & Calabrese et al (2014) <doi:10.1086/675504>, Fleming et al (2014) <doi:10.1111/2041-210X.12176>, Fleming et al (2015) <doi:10.1890/14-2010.1>, Fleming et al (2016) <doi:10.1890/15-1607>, Péron & Fleming et al (2016) <doi:10.1186/s40462-016-0084-7>, Fleming & Calabrese (2016) <doi:10.1111/2041-210X.12673>, Péron et al (2017) <doi:10.1002/ecm.1260>, Fleming et al (2017) <doi:10.1016/j.ecoinf.2017.04.008>, Fleming et al (2018) <doi:10.1002/eap.1704>, and Winner & Noonan et al (2018) <doi:10.1111/2041-210X.13027>.

Details

Package:	ctmm
Type:	Package
Version:	0.5.6
Date:	2019-05-14
License:	GPL-3

- CTMM Initiative
- CRAN package
- Github project
- Source packages
- Github reference
- Google group
- ctmm-FAQ

Author(s)

Christen H. Fleming and Justin M. Calabrese

Maintainer: Christen H. Fleming <flemingc@si.edu>

References

C. H. Fleming, J. M. Calabrese, T. Mueller, K.A. Olson, P. Leimgruber, W. F. Fagan. From fine-scale foraging to home ranges: A semi-variance approach to identifying movement modes across spatiotemporal scales. The American Naturalist, 183:5, E154-E167 (2014).

C. H. Fleming, J. M. Calabrese, T. Mueller, K. A. Olson, P. Leimgruber, W. F. Fagan. Non-Markovian maximum likelihood estimation of autocorrelated movement processes. Methods in Ecology and Evolution, 5:5 462-472 (2014).

C. H. Fleming, Y. Subasi, J. M. Calabrese. A maximum-entropy description of animal movement. Physical Review E, 91, 032107 (2015).

C. H. Fleming, W. F. Fagan, T. Mueller, K. A. Olson, P. Leimgruber, J. M. Calabrese. Rigorous home-range estimation with movement data: A new autocorrelated kernel-density estimator. Ecology, 96:5, 1182-1188 (2015).

J. M. Calabrese, C. H. Fleming. ctmm: an R package for analyzing animal relocation data as a continuous-time stochastic process. Methods in Ecology and Evolution, 7:9, 1124-1132 (2016).

C. H. Fleming, W. F. Fagan, T. Mueller, K. A. Olson, P. Leimgruber, J. M. Calabrese. Estimating where and how animals travel: An optimal framework for path reconstruction from autocorrelated tracking data. Ecology, 97:3, 576-582 (2016).

G. Péron, C. H. Fleming, R. C. de Paula, J. M. Calabrese. Uncovering periodic patterns of space use in animal tracking data with periodograms, including a new algorithm for the Lomb-Scargle periodogram and improved randomization tests. Movement Ecology, 4:19, DOI:10.1186/s40462-016-0084-7 (2016).

C. H. Fleming, J. M. Calabrese. A new kernel-density estimator for accurate home-range and species-range area estimation. Methods in Ecology and Evolution, 8:5, 571-579 (2016).

G. Péron, C. H. Fleming, R. C. de Paula, N. Mitchell, M. Strohbach, P. Leimgruber, J. M. Calabrese. Periodic continuous-time movement models uncover behavioral changes of wild canids along anthropization gradients. Ecological Monographs 87:3, 442-456 (2017)

C. H. Fleming, D. Sheldon, E. Gurarie, W. F. Fagan, S. LaPoint, J. M. Calabrese. Kálmán filters for continuous-time movement models. Ecological Informatics, 40, 8-21 (2017).

C. H. Fleming, D. Sheldon, W. F. Fagan, P. Leimgruber, T. Mueller, D. Nandintsetseg, M. J. Noonan, K. A. Olson, E. Setyawan, A. Sianipar, J. M. Calabrese. Correcting for missing and irregular data in home-range estimation. Ecological Applications, 28:4, 1003-1010 (2018).

K. Winner, M. J. Noonan, C. H. Fleming, K. Olson, T. Mueller, D. Sheldon, J. M. Calabrese. Statistical inference for home range overlap. Methods in Ecology and Evolution, DOI:10.1111/2041-210X.13027 (2018).

---

akde — *Calculate an autocorrelated kernel density estimate*

Description

This function calculates autocorrelated kernel density home-range estimates from `telemetry` data and a corresponding continuous-time movement model.

Usage

```
akde(data,CTMM,VMM=NULL,debias=TRUE,smooth=TRUE,error=0.001,res=10,grid=NULL,...)

## S3 method for class 'telemetry'
akde(data,CTMM,VMM=NULL,debias=TRUE,smooth=TRUE,error=0.001,res=10,grid=NULL,...)

## S3 method for class 'list'
akde(data,CTMM,VMM=NULL,debias=TRUE,smooth=TRUE,error=0.001,res=10,grid=NULL,...)

## S3 method for class 'UD'
mean(x,...)
```

Arguments

`data`	2D timeseries telemetry data represented as a `telemetry` object or list of objects.
`CTMM`	A `ctmm` movement model from the output of `ctmm.fit` or list of objects.
`VMM`	An optional vertical `ctmm` object for 3D home-range calculation.
`debias`	Debias the distribution for area estimation (AKDEc).
`smooth`	"Smooth" out errors from the data.
`error`	Target probability error.
`res`	Number of grid points along each axis, relative to the bandwidth.
`grid`	Optional grid specification with columns labeled `x` and `y`. Not yet supported.
`...`	Arguments passed to all instances of `bandwidth`, such as `weights`.
`x`	A list of `UD`s calculated on the same grid.

Details

For weighted AKDE, please note additional `...` arguments passed to `bandwidth` and the `weights=TRUE` argument, specifically.

When feeding in lists of `telemetry` and `ctmm` objects, all UDs will be calculated on the same grid. These UDs can be averaged with the `mean` command, however this is not an optimal way to calculate population ranges.

Value

Returns a `UD` object: a list with the sampled grid line locations `r$x` and `r$y`, the extent of each grid cell `dr`, the probability density and cumulative distribution functions evaluated on the sampled grid locations `PDF` & `CDF`, the optimal bandwidth matrix `H`, and the effective sample size of the data in `DOF.H`.

Note

In the case of coarse grids, the value of `PDF` in a grid cell corresponds to the average probability density over the entire rectangular cell.

Prior to `ctmm` v0.3.2, the default AKDE method was the autocorrelated Gaussian reference function bandwidth. Starting in v0.3.2, the default AKDE method is the autocorrelated Gaussian reference function bandwidth with debiased area.

Prior to `ctmm` v0.3.1, AKDEs included only errors due to autocorrelation uncertainty, which are insignificant in cases such as IID data. Starting in v0.3.1, `akde` calculated an effective sample size `DOF.H` and used this to estimate area uncertainty under a Gaussian reference function approxmation. In v0.3.2, this method was further improved to use `DOF.area` from the Gaussian reference function approximation.

Author(s)

C. H. Fleming and K. Winner.

References

C. H. Fleming, W. F. Fagan, T. Mueller, K. A. Olson, P. Leimgruber, J. M. Calabrese. Rigorous home-range estimation with movement data: A new autocorrelated kernel-density estimator. Ecology, 96:5, 1182-1188 (2015).

C. H. Fleming, J. M. Calabrese. A new kernel-density estimator for accurate home-range and species-range area estimation. Methods in Ecology and Evolution, 8:5, 571-579 (2016).

C. H. Fleming, D. Sheldon, W. F. Fagan, P. Leimgruber, T. Mueller, D. Nandintsetseg, M. J. Noonan, K. A. Olson, E. Setyawan, A. Sianipar, J. M. Calabrese. Correcting for missing and irregular data in home-range estimation. Ecological Applications, 28:4, 1003-1010 (2018).

See Also

`bandwidth`, `raster,UD-method`

Examples

```
# Load package and data
library(ctmm)
data(buffalo)
Cilla <- buffalo$Cilla

GUESS <- ctmm.guess(Cilla,interactive=FALSE)
FIT <- ctmm.fit(Cilla,GUESS)

# Compute akde object
UD <- akde(Cilla,FIT)

# Plot data with AKDE contours
plot(Cilla,UD=UD)
```

as.telemetry *Import, coerce, and summarize MoveBank data*

Description

Functions to import MoveBank csv files, `data.frame`, and `Move` objects, coerce them into `telemetry` objects, and summarize them.

Usage

```
as.telemetry(object,timeformat="",timezone="UTC",projection=NULL,timeout=Inf,na.rm="row",
             mark.rm=FALSE,drop=TRUE,...)

## S3 method for class 'character'
as.telemetry(object,timeformat="",timezone="UTC",projection=NULL,timeout=Inf,na.rm="row",
             mark.rm=FALSE,drop=TRUE,...)

## S3 method for class 'data.frame'
as.telemetry(object,timeformat="",timezone="UTC",projection=NULL,timeout=Inf,na.rm="row",
             mark.rm=FALSE,drop=TRUE,...)

## S3 method for class 'Move'
as.telemetry(object,timeformat="",timezone="UTC",projection=NULL,timeout=Inf,na.rm="row",
             mark.rm=FALSE,drop=TRUE,...)

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

Arguments

`object`	A MoveBank CSV filename, MoveBank `data.frame` object, or `Move` object to coarce, or a `telemetry` object to summarize.
`timeformat`	Format argument for `strptime`.
`timezone`	Timezone argument for `strptime`.
`projection`	Optional PROJ.4 projection argument to be fed to `project`.
`timeout`	GPS location fix timeout value (seconds) for assigning a `"timed-out"` location class.
`na.rm`	If some values are `NA` in the data frame, are the rows (times) deleted or are the columns (data types) deleted.
`mark.rm`	Delete Movebank manually marked outliers. Also see `outlie`.
`drop`	Only return a `telemetry` object for one individual if `TRUE`. Always return a `list` of telemetry objects if `FALSE`.
`...`	Optional arguments to be fed to `fread` or `read.csv`, in the case of compressed files.

Details

If no projection argument is specified, a two-point equidistant projection is calculated that should be good for most range resident and migratory species. Global migrations that are not along one geodesic (locally straight line) will probably suffer distortion.

`as.telemetry` assumes Movebank naming conventions. Sufficient MoveBank columns include `individual.local.identifier` (or `tag.local.identifier`), `timestamp`, `location.long` and `location.lat`, while the optional Movebank columns include (E-OBS) `eobs.horizontal.accuracy.estimate`, (Telonics) `GPS.Horizontal.Error`, `GPS.HDOP`, (ARGOS) `Argos.orientation`, `Argos.semi.minor` and `Argos.semi.major` or `Argos.location.class`, etc.. To have all columns detected and not overwrite eachother's information, **it is best to have only one tracking device model per file imported**. Multiple deployments on a single individual can be merged afterwards.

The `GPS.HDOP` value requires the device's UERE value to be translated into telemetry errors. The UERE represents the RMS error given ideal ($HDOP = 1$) satellite conditions. Therefore, if your device is purported to have an accuracy of 10 meters, then the UERE is likely around 10 meters. Consult your device manual and manufacturer for a specific value or see `uere` for UERE calibration.

Value

`as.telemetry` returns a single `telemetry` object or list of `telemetry` objects if multiple animals are identified.

Author(s)

C. H. Fleming, X. Dong, B. Kranstauber, G. Péron, and K. Safi.

See Also

`plot.telemetry`, `SpatialPoints.telemetry`, `uere`.

`bandwidth` *Calculate the optimal bandwidth matrix of movement data*

Description

This function calculates the optimal bandwidth matrix (kernel covariance) for a two-dimensional animal tracking dataset, given an autocorrelated movement model (Fleming et al, 2015). This optimal bandwidth can fully take into account all autocorrelation in the data, assuming it is captured by the movement model.

Usage

```
bandwidth(data,CTMM,VMM=NULL,weights=FALSE,fast=TRUE,dt=NULL,precision=1/2,PC="Markov",
  verbose=FALSE,trace=FALSE)
```

Arguments

`data`	2D timeseries telemetry data represented as a `telemetry` object.
`CTMM`	A `ctmm` movement model as from the output of `ctmm.fit`.
`VMM`	An optional vertical `ctmm` object for 3D bandwidth calculation.
`weights`	By default, the weights are taken to be uniform, whereas `weights=TRUE` will optimize the weights.
`fast`	Use FFT algorithms for weight optimization.
`dt`	Optional lag bin width for the FFT algorithm.
`precision`	Fraction of maximum possible digits of precision to target in weight optimization. `precision=1/2` results in about 7 decimal digits of precision if the preconditioner is stable.
`PC`	Preconditioner to use: can be "Markov", "circulant", "IID", or "direct".
`verbose`	Optionally return the optimal `weights`, effective sample size `DOF.H`, and other information along with the bandwidth matrix `H`.
`trace`	Produce tracing information on the progress of weight optimization.

Details

The `weights=TRUE` argument can be used to correct temporal sampling bias caused by autocorrelation. `weights=TRUE` will optimize `n=length(data$t)` weights via constrained & preconditioned conjugate gradient algorithms. These algorithms have a few options that should be considered if the data are very irregular.

`fast=TRUE` grids the data with grid width `dt` and applies FFT algorithms, for a computational cost as low as $O(n \log n)$ with only $O(n)$ function evaluations. If no `dt` is specified, the minimum sampling interval `min(diff(data$t))` is used. **If the data are irregular (permitting gaps), then `dt` may need to be several times smaller** to avoid slow down. In this case, try setting `trace=TRUE` and decreasing `dt` until the interations speed up and the number of feasibility assessments becomes less than $O(n)$. On the other hand, if the data contain some very tiny time intervals, say 1 second among hourly sampled data, then the default `dt` setting will create an excessively high-resolution discretization of time, which will also cause slowdown. In this case `CTMM` should contain an error model and `dt` can likely be increased to a larger fraction of the median sampling interval.

`fast=FALSE` uses exact times and has a computational cost as low as $O(n^2)$, including $O(n^2)$ function evaluations. With `PC="direct"` this method will produce a result that is exact to within machine precision, but with a computational cost of $O(n^3)$.

Value

Returns a bandwidth `matrix` object, which is to be the optimal covariance matrix of the individual kernels of the kernel density estimate.

Note

To obtain a bandwidth scalar representing the variance of each kernel, a `ctmm` object with `isotropic=TRUE` is required. In this case, `bandwidth` will return bandwidth matrix with identical variances along its diagonal. Note that this will provide an inaccurate estimate for very eccentric distributions.

Author(s)

C. H. Fleming.

References

T. F. Chan. An Optimal Circulant Preconditioner for Toeplitz Systems. SIAM Journal on Scientific and Statistical Computing, 9:4, 766-771 (1988).

D. Marcotte. Fast variogram computation with FFT. *Computers and Geosciences* 22:10, 1175-1186 (1996).

C. H. Fleming, W. F. Fagan, T. Mueller, K. A. Olson, P. Leimgruber, J. M. Calabrese. Rigorous home-range estimation with movement data: A new autocorrelated kernel-density estimator. Ecology, 96:5, 1182-1188 (2015).

C. H. Fleming, D. Sheldon, W. F. Fagan, P. Leimgruber, T. Mueller, D. Nandintsetseg, M. J. Noonan, K. A. Olson, E. Setyawan, A. Sianipar, J. M. Calabrese. Correcting for missing and irregular data in home-range estimation. Ecological Applications, 28:4, 1003-1010 (2018).

See Also

`akde`, `ctmm.fit`

`buffalo` *African buffalo GPS dataset from Kruger National Park, South Africa.*

Description

GPS data on six African buffalo. When using this dataset, please cite the original article by Getz et al (2007) and the Movebank data package (Cross et al, 2016).

Usage

```
data("buffalo")
```

Format

A list of 6 `telemetry` objects.

Note

In `ctmm` v0.3.2 the erroneous location fix `606` was removed from `buffalo[[4]]` "Pepper".

References

W. M. Getz, S. Fortmann-Roe, P. C. Cross, A. J. Lyons, S. J. Ryan, C. C. Wilmers. LoCoH: Nonparameteric kernel methods for constructing home ranges and utilization distributions. PLoS ONE 2:2, e207 (2007).

P. C. Cross, J. A. Bowers, C. T. Hay, J. Wolhuter, P. Buss, M. Hofmeyr, J. T. du Toit, W. M. Getz. Data from: Nonparameteric kernel methods for constructing home ranges and utilization distributions. Movebank Data Repository. DOI:10.5441/001/1.j900f88t (2016).

See Also

as.telemetry, plot.telemetry, coati, gazelle, turtle, wolf.

Examples

```
# Load package and data
library(ctmm)
data("buffalo")

# Extract movement data for a single animal
Cilla <- buffalo$Cilla

# Plot all sampled locations
plot(Cilla)
```

---

coati — *Coatis on Barro Colorado Island, Panama.*

Description

GPS data on 2 coati. When using this dataset, please cite the original article by Powell et al (in preparation) and the Movebank data package (Kays and Hirsch, 2015).

Usage

```
data("coati")
```

Format

A list of 2 `telemetry` objects.

References

R. A. Powell, S. Ellwood, R. Kays. Stink or swim: techniques to meet the challenges for the study and conservation of small critters that hide, swim or climb and may otherwise make themselves unpleasant. In L. Harrington and D. W. Macdonald; Biology and Conservation of Mustelids and Procyonids (in preparation).

R. Kays, B. T. Hirsch Data from: Stink or swim: techniques to meet the challenges for the study and conservation of small critters that hide, swim or climb and may otherwise make themselves unpleasant. Movebank Data Repository. DOI:10.5441/001/1.41076dq1 (2015).

See Also

as.telemetry, plot.telemetry, buffalo, gazelle, turtle, wolf.

Examples

```
# Load package and data
library(ctmm)
data("coati")

# Plot all sampled locations
plot(coati,col=rainbow(2))
```

---

color *Color telemetry objects by time*

Description

These functions facilitate the coloring of tracks by annotating tracking data with time/location specific information and computing color arguments for `plot`.

Usage

```
annotate(object,by="all",cores=1,...)

color(object,by="time",col.fn=NULL,alpha=1,dt=NULL,cores=1,...)
```

Arguments

`object`	A `telemetry` object or list of objects. `color` can also take `ctmm` and `UD` objects.
`by`	What to `annotate` or `color` times by. Options include `"individual"`, `"time"`, `"sun"`, `"moon"`, `"season"`, and `"tropic"` (see Details below). `ctmm` and `UD` objects can only be colored by `"individual"`.
`col.fn`	Optional coloring function that can take a [0,1] interval and alpha channel argument.
`alpha`	Base alpha channel value.
`dt`	Sampling interval specification for making oversampled times more transparent. If `NULL`, the median will be used. Disabled if zero.
`cores`	Number of annotations or overlap calculations to peform in parallel. `cores=0` will use all cores, while `cores<0` will reserve `abs(cores)`.
`...`	Additional arguments.

Details

Annotated `telemetry` objects are required for `color` `by` arguments `"sun"`, `"moon"`, `"season"`, or `"tropic"`.

`by="time"` colors tracking data with a gradient that increases in time. `by="sun"` colors according to the sine of the sun's altitude, which is proportional to solar flux during daylight hours. `by="moon"` colors according to the illuminated fraction of the moon. `by="season"` colors according to the length of the day, and therefore corresponds to the local season. `by="tropic"` currently colors according to the calender day, but will eventually be upgraded to tropical-year cycle.

`by="individual"` assigns colors to minimize the maximum combined spatial and color overlap. Finding the best color assignment is an NP-hard problem that is here approximated in $O(N^3)$ time with a custom greedy algorithm.

Other named columns in the `telemetry` object can also be used with `color`, by specifying the column name with `by`.

Value

`annotate` returns an annotated telemetry object with extra columns to facilitate coloring. `color` returns a valid `col` argument for `{plot.telemetry}`.

Author(s)

C. H. Fleming.

See Also

`plot.telemetry`

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# assign distinct colors to buffalo
COL <- color(buffalo,by='individual')
# Notice the separation into RGB and CMY for maximum contrast
plot(buffalo,col=COL)

# annotate buffalo with sunlight data and compute colors
buffalo <- annotate(buffalo,cores=2) # CRAN policy limits to 2 cores
COL <- color(buffalo,by='sun')

# use North-preserving projection and plot
projection(buffalo) <- median(buffalo)
plot(buffalo,col=COL)
```

`ctmm` *Specify, fit, and select continuous-time movement models*

Description

These functions allow one to propose hypothethical movement models (with initial estimates), fit those models to the data, and select among those models via an information criteria. The fitting functions wrap around `optim` and `ctmm.loglike` to fit continuous-time movement models to 2D animal tracking data as described in Fleming et al (2014) and Fleming et al (2015), and Fleming et al (2017).

Usage

```
ctmm(tau=NULL,omega=FALSE,isotropic=FALSE,range=TRUE,circle=FALSE,error=FALSE,
     axes=c("x","y"),...)

ctmm.loglike(data,CTMM,REML=FALSE,profile=TRUE,zero=0,verbose=FALSE)

ctmm.fit(data,CTMM=ctmm(),method="ML",COV=TRUE,control=list(),trace=FALSE)

ctmm.select(data,CTMM,verbose=FALSE,level=1,IC="AICc",MSPE="position",trace=FALSE,cores=1,
            ...)
```

Arguments

`tau`	Array of autocorrelation timescales (explained below).
`omega`	Frequency ($2\pi/period$) of oscillatory range crossings.
`isotropic`	A Boolean denoting whether or not the animal's covariance is circular or elliptical.
`range`	A Boolean denoting whether or not the movement model has a finite range.
`circle`	The period it takes the animal to stochastically circle its mean location.
`error`	A Boolean denoting whether or not to use annotated telemetry error estimates or an estimate of the error's standard deviation if the data are not annotated with error estimates or when $HDOP = 1$.
`axes`	Spatial dimensions of the movement model.
`data`	Timeseries data represented as a `telemetry` object.
`CTMM`	A `ctmm` movement-model object containing the initial parameter guesses conforming to the basic structure of the model hypothesis. `ctmm.select` can accept a list of such objects.
`REML`	Use residual maximum likelihood if `TRUE`. Not recommended.
`profile`	Analytically solve for as many covariance parameters as possible.
`zero`	Calculates $log(likelihood) - zero$, instead of just $log(likelihood)$, in a way that maintains numerical precision if the constant `zero` is close to the log likelihood. Used internally by `ctmm.fit`.
`verbose`	Return additional information. See "Value" below.
`method`	Fitting method to use: `"ML"`, `"HREML"`, `"pREML"`, `"pHREML"`, or `"REML"`. See "Description" below.
`COV`	Estimate the autocorrelation parameter covariance matrix.

`control`	An optional argument list with standardized numerical arguments.
`trace`	Report progress updates. Can be among `0:2` with increasing detail.
`level`	Attempt to simplify a model if a feature's non-existence falls within this level of confidence interval.
`IC`	Information criteria used for selection. Can be `"AICc"`, `"AIC"`, `"BIC"` or none (`NA`). AICc is approximate.
`MSPE`	Reject non-stationary features that increase the mean square predictive error of `"position"`, `"velocity"`, or not (`NA`).
`cores`	Maximum number of models to fit in parallel. cores=0 will use all cores, while cores<0 will reserve abs(cores).
`...`	Further arguments passed to `ctmm.fit`.

Details

Model fitting and selection first requires a prototype model with guesstimated parameters (i.e., Brownian motion with a particular diffusion rate). The initial `ctmm` parameter guess can be generated by the output of `ctmm.guess`, `variogram.fit` or manually specified with the function `ctmm(...)`, where the argument `tau` is explained below and additonal model options described in `vignette("ctmm")`.

By default, `tau` (τ) is an ordered array of autocorrelation timescales. If `length(tau)==0`, then an IID bi-variate Gaussian model is fit to the data. If `length(tau)==1`, then an Ornstein-Uhlenbeck (OU) model (Brownian motion restricted to a finite home range) is fit the data, where `tau` is the position autocorrelation timescale. `tau=Inf` then yields Brownian motion (BM). If `length(tau)==2`, then the OUF model (continuous-velocity motion restricted to a finite home range) is fit to the data, where `tau[1]` is again the position autocorrelation timescale and `tau[2]` is the velocity autocorrelation timescale. `tau[1]=Inf` then yields integrated Ornstein-Uhlenbeck (IOU) motion, which is a spatially unrestricted continuous-velocity process.

Two new models were introduced in ctmm version 0.5.2 for the case of `tau[1]==tau[2]`, which can happen with short tracks of data. When `tau[1]==tau[2]` and `omega==0`, the model is categorized as OUf—a special case of OUF—and the two `tau` parameters are treated as identical. On the other hand, when `tau[1]==tau[2]` and `omega>0`, an oscillatory model is implemented, which we refer to as OUΩ.

The potential fitting methods—maximum likelihood (`ML`), residual maximum likelihood (`REML`), perturbative REML (`pREML`), hybrid REML (`HREML`), and perturbative hybrid REML (`pHREML`)—are described in Fleming et al (2019). In general, `pHREML` is the best method, though when parameter estimates lie near boundaries it can fail, in which case `ctmm.fit` will fall back to `HREML`, as reported by the `method` slot of the resulting fit object.

The `control` list can take the folowing arguments, with defaults shown:

`method="Nelder-Mead"` `optim` method for multiple parameters.

`precision=1/2` Fraction of machine numerical precision to target in the maximized likelihood value. MLEs will necessarily have half this precision. On most computers, `precision=1` is approximately 16 decimal digits of precision for the likelihood and 8 for the MLEs.

`maxit=.Machine$integer.max` Maximum number of iterations allowed for optimization.

Model selection in `ctmm.select` proceeds by first fitting the initial model guess, and then attempting to simplify the autocorrelation model and complexify the deterministic (mean) model until the information criteria fails to improve. The intent of working in these directions is to avoid fitting trends to autocorrelation. Note that simpler models in a nested hierarchy will only be attempted if they appear credible, which can be adjusted with the `level` argument. `level=1` will, therefore, always attempt a simpler model.

Value

The function `ctmm` returns a prototype `ctmm` movement-model object. By default, `ctmm.loglike` returns the log-likelihood of the model `CTMM`. `ctmm.fit` (and `ctmm.loglike` with `verbose=TRUE`) returns the maximum likelihood `ctmm` movement-model object with all of the components of `CTMM` plus the components listed below. `ctmm.select` returns the best model by default or the sorted list of attempted models if `verbose=TRUE`.

`AICc` The approximate corrected Akaike information criterion for multivariate distributions with variable numbers of unknown mean and (structured) covariance parameters (Burnham & Anderson, Eq. 7.91). This formula is only exact for IID data.

`loglike` The log-likelihood.

`sigma` The maximum likelihood variance/covariance estimate (possibly debiased). For the endlessly diffusing BM and IOU processes, this is instead the diffusion rate estimate.

`mu` The maximum likelihood stationary mean vector estimate.

`COV.mu` The covariance matrix of the `mu` estimate, assuming that the covariance estimate is correct.

`DOF.mu` The effective number of degrees of freedom in the estimate of `mu`, assuming that the autocorrelation model is correct. This can be much smaller than `length(data$t)` if the data are autocorrelated.

`COV` Covariance of the autocovariance parameter estimate vector `c(sigma,tau,circle)`, as derived (asymptotically) from the `hessian` of the log-likelihood function, and where `sigma` is parameterized in terms of its standard `area`, `eccentricity`, and `angle` of orientation. Typically, `sigma`'s `area` parameter is extremely correlated to `tau[1]`, and sequential components of `tau` are slightly correlated.

Note

The default `optim` method here is `"Nelder-Mead"` with the largest allowable number of maximum iterations. Annecdotally, the faster `"BFGS"` and `"L-BFGS-B"` methods often perform poorly on these types of problems. However, Nelder Mead is not ideal and you may want to attempt a second fit, using the first fit as its initial guess.

The AICs/BICs of endlessly diffusing models like BM and IOU cannot be easily compared to the AICs/BICs of range resident models like bivariate Gaussian, OU, and OUF, as their joint likelihood functions are infinitely different. Endlessly diffusing models have to be conditioned off of an initial state, which we derive in `ctmm` by taking the large range limit of a range-restricted process. I.e., BM is the limit OU(`Inf`) and IOU(`tau`) is the limit OUF(`Inf`,`tau`). Using comparable likelihood functions gives up statistical efficiency and the objective prior. Moreover, comparing conditional likelihoods—with the objective prior taken from the joint likelihood—does not appear to select the true model with a likelihood ratio test. Therefore, there does not appear to be a simple approach for selecting between range resident and endlessly diffusing movement models.

Prior to v0.3.6, the univariate AICc formula was (mis)used, with the full parameter count treated as degrees of freedom in the mean. As of v.0.3.6, the mean and autocovariance parameters are treated separately in the approximate multivariate AICc formula (Burnham & Anderson, Eq. 7.91). Still, this improved formula is only exact for IID data.

Prior to v0.3.2, `ctmm.select` would consider every possible model. This is no longer feasible with current versions of `ctmm`, as the number of possible models has grown too large.

Author(s)

C. H. Fleming and G. Péron.

References

K. P. Burnham, D. R. Anderson. Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach. Springer, 2nd edition (2003).

C. H. Fleming, J. M. Calabrese, T. Mueller, K.A. Olson, P. Leimgruber, W. F. Fagan. From fine-scale foraging to home ranges: A semi-variance approach to identifying movement modes across spatiotemporal scales. The American Naturalist, 183:5, E154-E167 (2014).

C. H. Fleming, Y. Subasi, J. M. Calabrese. A maximum-entropy description of animal movement. Physical Review E, 91, 032107 (2015).

C. H. Fleming, D. Sheldon, E. Gurarie, W. F. Fagan, S. LaPoint, J. M. Calabrese. Kálmán filters for continuous-time movement models. Ecological Informatics, 40, 8-21 (2017).

See Also

`ctmm.boot`, `ctmm.guess`, `optim`, `summary.ctmm`, `variogram.fit`.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)
Cilla <- buffalo$Cilla

GUESS <- ctmm.guess(Cilla,interactive=FALSE)
FIT <- ctmm.fit(Cilla,GUESS)

# some human-readable information
summary(FIT)
```

ctmm-FAQ *ctmm FAQ*

Description

Frequently asked questions for the `ctmm` package.

Details

General recommendations

1. Never edit your CSV in Microsoft Excel. The dates will be reformatted incorrectly.
2. Do not save workspaces between sessions. They can become corrupted.
3. Upgrade `R` to the latest version and update all of your packages.
4. Work through the vignettes `vignette("variogram")` and `vignette("akde")`.
5. Stable beta releases between the CRAN release are published here on request.
6. The development build can be installed via `devtools::install_github("ctmm-initiative/ctmm")`.
7. The `ctmm` user's group is a good place to find and ask for help.
8. Bug reports and feature requests can be raised at the Github project page.

Help installing packages on Linux

These are the packages I needed in Ubuntu:

`sudo apt install fftw3 fftw3-dev libgdal1-dev libproj-dev libgeos-dev libgsl-dev libv8-dev r-base-cor`

`ctmm.select` only returns one model with `verbose=TRUE`

The newer version of `ctmm.select` is much more intelligent, necessitated by the exploding number of new model permutations. Any simpler models were judged to be so unlikely that they were not attempted. You can set `level=1` to force alternative model fitting, but the additional models should lose badly in model selection (please contact me if I am wrong).

`as.telemetry` reports abnormal sampling intervals and speeds

Make sure that you have the correct `timezone` and `timeformat` arguments specified. Also, see `outlie`.

rdb database corruption, "could not find function", "cannot coerce class", and other weird errors

`R` might not have installed or loaded the package correctly—e.g., some files may have failed to overwrite previous versions—or the workspace/session might be corrupted. Uninstall `ctmm`, restart `R` without saving the workspace/session, and install `ctmm` again.

Infinite recursion and stack overflow errors

`ctmm` has no recursive functions, so I am not exactly sure what causes this error, but it only occurs with certain versions of `R` on certain computer architectures. There are several solutions that have worked for people, including restarting `R` in a fresh session and updating their software. Alternatively:

1. Reboot your computer.

2. Increase the allowed number of nested expressions within R via `options(expressions=10000)` or some other large number.
3. Try a different computer.

`plot` **complains about the datatype or has weird errors**

Namespace collision sometimes occurs from `raster` or `sp` against `move` and `ctmm`. Restart R and only load the `ctmm` package.

North is no longer up after importing data

The default projection in `ctmm` does not preserve the direction of North, but better preserves distances for elongated distributions. See the `projection` argument in `as.telemetry` and the example in `projection`.

`variogram.fit` **has no save button**

Maximize the plot window and/or increase your screen resolution.

`manipulate::isAvailable` **is not found**

You probably have an outdated copy of the `manipulate` package installed. Update R to the latest version and then update all of your packages. This seems to happen frequently with the MacOS release of R.

`manipulate` **panel does not popup in** `variogram.fit` **or** `zoom`

Click the gear icon in the upper-left corner of the plot window.

Author(s)

C. H. Fleming

`ctmm.boot` *Parametric bootstrap continuous-time movement models*

Description

This function allows the point estimates and confidence intervals of an initial estimated movement model to be improved by parametric boostrap, as described in Fleming et al (2019).

Usage

```
ctmm.boot(data,CTMM,method=CTMM$method,iterate=FALSE,robust=FALSE,error=0.01,cores=1,
          trace=TRUE,...)
```

Arguments

`data`	Timeseries data represented as a `telemetry` object.
`CTMM`	A `ctmm` movement-model object from the output of `ctmm.fit` containing the initial parameter estimates.
`method`	Fitting method to use: `"ML"`, `"HREML"`, `"pREML"`, `"pHREML"`, or `"REML"`. See `ctmm.fit` for descriptions.

`iterate`	Iteratively solve for the parameters such that the average estimate (of `method`) is that of the data, whereas with `iterate=FALSE` only the first-order correction is calculated from the initial estimate.
`robust`	Uses robust estimates of the average and covariation for debiasing. Useful when parameters are near boundaries.
`error`	Relative standard error target for bootstrap ensemble estimates and nonlinear iterations.
`cores`	Number of simulations to run in parallel. `cores=NULL` will use all cores, while `cores<0` will reserve `abs(cores)`.
`trace`	Report progress updates. Can be among `0:2` with increasing detail.
`...`	Further arguments passed to `ctmm.fit`.

Value

A model fit object with relatively unbiased estimates of location covariance, and autocorrelation timescales (and more accurate CIs than `ctmm.fit`).

Author(s)

C. H. Fleming.

See Also

`ctmm.fit`.

`emulate` *Draw a random model-fit from the sampling distribution*

Description

This function generates random model-fit statistics from the sampling distribution of a given `ctmm` movement model and sampling schedule. If `fast=FALSE`, the results are exact, though slow to evaluate. Else if `fast=TRUE`, the central-limit theorem is invoked.

Usage

```
emulate(object,...)

## S3 method for class 'ctmm'
emulate(object,data=NULL,fast=FALSE,...)

## S3 method for class 'telemetry'
emulate(object,CTMM,fast=FALSE,...)
```

Arguments

`object`	`telemetry` data or `ctmm` model object.
`CTMM`	A `ctmm` movement-model object.
`data`	Optional `telemetry` object for exact results.
`fast`	Whether or not to invoke the central-limit theorem.
`...`	Arguments passed to `ctmm.fit`.

Details

Given `fast=FALSE`, which requires the `data` argument specified, new data are simulated from the `CTMM` movement model with the same sampling schedule and error structure as `data`. A new model, of the same structure as `CTMM`, is then fit to the simulated data and returned.

Given `fast=TRUE`, a model-fit object is sampled from the central-limit distribution, using the covariance estimates within `CTMM`. Strictly positive parametes, such as area, are log-transformed prior to the normal approximation. Note that this faster method does not adjust for bias.

Value

A `ctmm` movement model with the same structure as `CTMM`.

Author(s)

C. H. Fleming.

See Also

`ctmm.fit`, `simulate.ctmm`

`export` *Export ctmm data formats*

Description

Functions to export `ctmm` data formats into common `sp`, `raster`, and ESRI formats.

Usage

```
## S4 method for signature 'UD'
raster(x,DF="CDF",...)

SpatialPoints.telemetry(object,...)

SpatialPointsDataFrame.telemetry(object,...)

SpatialPolygonsDataFrame.UD(object,level.UD=0.95,level=0.95,...)
```

```
## S4 method for signature 'UD,character'
writeRaster(x,filename,format,DF="CDF",...)

writeShapefile(object,folder,file=NULL,...)

## S3 method for class 'telemetry'
writeShapefile(object,folder,file=NULL,...)

## S3 method for class 'UD'
writeShapefile(object,folder,file=NULL,level.UD=0.95,level=0.95,...)
```

Arguments

x	UD object.
DF	Rasterize the probability density function "PDF", probability mass function "PMF", or cumulative distribution function "CDF".
object	telemetry or UD object.
level.UD	Confidence level of the UD area. I.e., the 50% core home range would be given by level.UD=0.50.
level	Confidence level for the magnitude of the above area. I.e., the 95% CI of the core home range area.
filename	Character name of file for raster file.
format	Character format, if not inferred from filename extension (see writeRaster.)
folder	Character name of folder for shapefile.
file	Character name of files for shapefile.
...	Optional arguments passed to writeRaster, writeOGR, etc..

Details

writeRaster writes a raster file to disk, with pixel values corresponding to the density function.

writeShapefile writes a shapefile to disk, with UD polygons corresponding to the low, ML, and high home-range area estimates.

Value

raster.UD returns a raster of the maximum likelihood (ML) distribution function DF. DF=PDF gives the average probability density per cell, DF=PMF gives the total probability per cell, and DF=CDF gives the cumulative probability.

SpatialPoints.telemetry returns a single spatialPoints object for the x-y locations, with individual identity information lost. SpatialPointsDataFrame.telemetry returns a SpatialPointsDataFrame with the individual identities recorded in the data frame.

SpatialPolygonsDataFrame.UD returns a SpatialPolygonsDataFrame of the low, ML, and high home-range area estimates, in the appropriate order for plotting.

Author(s)

C. H. Fleming and K. Safi.

See Also

akde, as.telemetry, occurrence.

---

extent *Extent*

Description

Functions to calculate the (x, y) plotting extent (or bounding box) of various ctmm objects or list of such objects, for use when plotting multiple ctmm objects.

Usage

```
## S4 method for signature 'telemetry'
extent(x,level=1,...)

## S4 method for signature 'ctmm'
extent(x,level=0.95,level.UD=0.95,...)

## S4 method for signature 'UD'
extent(x,level=0.95,level.UD=0.95,...)

## S4 method for signature 'variogram'
extent(x,level=0.95,threshold=2,...)

## S4 method for signature 'list'
extent(x,...)
```

Arguments

`x`	A `telemetry`, `ctmm`, or `UD` object.
`level`	For `telemetry` objects, this is the fraction of locations bounded, according to two-sided quantiles. For `ctmm` and `UD` objects, this is confidence level for the magnitude of the utilization area circumscribed by `level.UD`.
`level.UD`	Confidence level of the UD area. I.e., the 50% core home range would be given by `level.UD=0.50`.
`threshold`	Limit `ylim` to `threshold` times the maximum semi-variance, even if the `level` confidence intervals exceed this amount.
`...`	Optional arguments for future extensions.

Details

Returns a `data.frame` with columns `x` and `y` with rows `min` and `max`. See `vignette('akde')` for an example of `extent` used to plot multiple UDs on the same scale.

Author(s)

C. H. Fleming

See Also

`plot.telemetry`, `plot.variogram`.

---

gazelle — *Mongolian gazelle GPS dataset from the Mongolia's Eastern Steppe.*

Description

`x-y` projected GPS data on 36 Mongolian gazelle.

Usage

```
data("gazelle")
```

Format

A list of 36 `telemetry` objects.

References

C. H. Fleming, J. M. Calabrese, T. Mueller, K.A. Olson, P. Leimgruber, and W. F. Fagan. Data from: From fine-scale foraging to home ranges: A semi-variance approach to identifying movement modes across spatiotemporal scales. Dryad Digital Repository, DOI:10.5061/dryad.45157 (2014).

See Also

`as.telemetry`, `plot.telemetry`, `buffalo`, `coati`, `turtle`, `wolf`.

Examples

```
# Load package and data
library(ctmm)
data("gazelle")

# Plot a gazelle's locations
plot(gazelle[[18]])
```

homerange *Calculate a range distribution estimate*

Description

This function estimates the range distribution from `telemetry` data and a continuous-time movement model.

Usage

```
homerange(data,CTMM,method="AKDE",...)
```

Arguments

`data`	2D timeseries telemetry data represented as a `telemetry` object.
`CTMM`	A `ctmm` movement model from the output of `ctmm.fit`.
`method`	Which range distribution method to use. For now only "AKDE" is supported.
`...`	Arguments passed to the method call.

Details

Please consult `akde` for further details.

Value

Returns a `UD` object.

Author(s)

C. H. Fleming.

See Also

`akde`, `raster,UD-method`

`mean.variogram` *Compute a number-weighted average of variogram objects*

Description

This function takes a list of variogram objects and calculates its number-weighted average variogram.

Usage

```
## S3 method for class 'variogram'
mean(x,...)
```

Arguments

`x`	A `variogram` object or list of such objects to be averaged.
`...`	Additional variograms if specified individually.

Value

Returns a `variogram` object which is a dataframe containing the lag, the semi-variance estimate at that lag, and the approximate degrees of freedom associated with the semi-variance estimate.

Note

Variogram averaging should only be used when there is a degree of similarity across individual variograms.

Author(s)

J. M. Calabrese and C. H. Fleming

References

C. H. Fleming, J. M. Calabrese, T. Mueller, K.A. Olson, P. Leimgruber, W. F. Fagan. From fine-scale foraging to home ranges: A semi-variance approach to identifying movement modes across spatiotemporal scales. The American Naturalist, 183:5, E154-E167 (2014).

See Also

`plot.variogram`, `variogram`.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# Calculate a list of variograms for all similar individuals in the dataset
# the 4th buffalo has a different sampling rate
SVFS <- lapply( buffalo[-4] , variogram )
# alternatively, we could variogram all at coarsest scale with variogram option dt

# Calculate the average variogram
SVF <- mean(SVFS)

# Plot the mean variogram
plot(SVF)
```

`occurrence` *Calculate a Kriged occurrence disribution estimate*

Description

This function calculates an occurrence distribution from `telemetry` data and a continuous-time movement model.

Usage

```
occurrence(data,CTMM,H=0,res.time=10,res.space=10,grid=NULL,cor.min=0.05,dt.max=NULL)
```

Arguments

`data`	2D timeseries telemetry data represented as a `telemetry` object.
`CTMM`	A `ctmm` movement model from the output of `ctmm.fit`.
`H`	Optional additional banwidth matrix for future use.
`res.time`	Number of temporal grid points per median timestep.
`res.space`	Number of grid points along each axis, relative to the average diffusion (per median timestep) from a stationary point.
`grid`	Optional grid specification with columns labeled `x` and `y`. Not yet supported.
`cor.min`	Velocity correlation threshold for skipping gaps.
`dt.max`	Maximum absolute gap size (in seconds) for Kriging interpolation. If left `NULL`, the median of `diff(data$t)` will be used.

Details

The arguments `cor.min` or `dt.max` are used to prevent the interpolation of large gaps, which would bias the estimate to more resemble the movement model than the data. Because `cor.min` can produce an empty range with fractal movement models, the larger of the two rules is employed.

Value

Returns a UD object containing the sampled grid line locations x and y, the probability density and cumulative distribution functions evaluated on the sampled grid locations PDF & CDF, the optional bandwidth matrix H, and the area of each grid cell dA.

Note

Large gaps have a tendency to slow down computation and blow up the estimate. This can be avoided with the cor.min or dt.max arguments.

In the case of coarse grids, the value of PDF in a grid cell actually corresponds to the average probability density over the entire rectangular cell.

Prior to ctmm v0.5.6, cor.min referred to the location correlation, with a default of 50%. In ctmm v0.5.6 and above, cor.min refers to the velocity correlation, with a default of 5%.

Author(s)

C. H. Fleming.

References

C. H. Fleming, W. F. Fagan, T. Mueller, K. A. Olson, P. Leimgruber, J. M. Calabrese. Estimating where and how animals travel: An optimal framework for path reconstruction from autocorrelated tracking data. Ecology, 97:3, 576-582 (2016).

C. H. Fleming, D. Sheldon, E. Gurarie, W. F. Fagan, S. LaPoint, J. M. Calabrese. Kálmán filters for continuous-time movement models. Ecological Informatics, 40, 8-21 (2017).

See Also

akde, raster,UD-method

Examples

```
# Load package and data
library(ctmm)
data(buffalo)
Cilla <- buffalo$Cilla

GUESS <- ctmm.guess(Cilla,interactive=FALSE)
FIT <- ctmm.fit(Cilla,GUESS)

# Compute occurence distribution
UD <- occurrence(Cilla,FIT)

# Plot occurrence UD
plot(UD,col.level=NA)
```

outlie *Methods to facilitate outlier detection.*

Description

Produces a data.frame of speed and distance estimates to analyze, as well as a plot highlighting potential speed and distance outliers in telemetry data.

Usage

```
outlie(data,UERE=10,standardize=FALSE,plot=TRUE,...)
```

Arguments

data	telemetry object.
UERE	Device-dependent telemetry error in meters. Only necessary for uncalibrated data.
standardize	Standardize speed and distance output by their median absolute deviation (MAD).
plot	Output a plot highlighting high speeds (blue) and distant locations (red).
...	Arguments passed to plot.

Details

Distances are calculated from the median longitude & latitude, while speeds are calculated over the timesteps. Both estimates account for telemetry error and condition on as few data points as possible. The speed estimates furthermore account for timestamp truncation and assign each timestep's speed to the most likely offending time, based on its other adjacent speed estimate.

If plot=TRUE, intervals of high speed are highlighted with blue segments, while distant locations are highlighted with red points.

Value

Returns a data.frame of distances and speeds. Can also produce a plot as a side effect.

Note

The speed estimates here are tailored for outlier detection and have poor statistical efficiency. The predict and speed methods are appropriate for estimating speed (after outliers have been removed and a movement model has been selected).

Author(s)

C. H. Fleming.

See Also

as.telemetry.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# look for outliers in Cilla
OUT <- outlie(buffalo[[1]])

# look at the distribution of speed esitmates
hist(OUT$speed)
```

---

overlap *Calculate the overlap between two stationary distributions*

Description

This function calculates a useful measure of similarity between distributions known as the *Bhattacharyya coefficient* in statistics and simply the *fidelity* or *overlap* in quantum and statistical mechanics. It is roughly speaking the ratio of the intersection area to the average individual area, but it is a direct comparison between the density functions and does not require an arbitrary quantile to be specified. When applied to `ctmm` objects, this function returns the overlap of the two Gaussian distributions. When applied to aligned `UD` objects with corresponding movement models, this function returns the overlap of their (autocorrelated) kernel density estimates.

Usage

```
overlap(object,level=0.95,debias=TRUE,...)
```

Arguments

`object`	A `list` of `ctmm` fit or aligned `UD` objects to compare.
`level`	The confidence level desired for the output.
`debias`	Approximate debiasing of the overlap.
`...`	Additional arguments relevant for `akde`, such as `res` and `weights`.

Value

A table of confidence intervals on the overlap estimate. A value of `1` implies that the two distributions are identical, while a value of `0` implies that the two distributions share no area in common. Corresponding `ctmm` objects are necessary to provide confidence intervals and debiasing for the point esitmates.

Note

In ctmm v0.5.2, direct support for telemetry objects was dropped and the CTMM argument was depreciated for UD objects, simplifying usage.

Uncertainties in the model fits are propagated into the overlap estimate under the approximation that the Bhattacharyya distance is a chi-square random variable. Debiasing makes further approximations noted in Winner & Noonan et al (2018).

Author(s)

C. H. Fleming and K. Winner

References

K. Winner, M. J. Noonan, C. H. Fleming, K. Olson, T. Mueller, D. Sheldon, J. M. Calabrese. Statistical inference for home range overlap. Methods in Ecology and Evolution, DOI:10.1111/2041-210X.13027 (2018).

See Also

akde, ctmm.fit

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# fit models for first two buffalo
GUESS <- lapply(buffalo[1:2], function(b) ctmm.guess(b,interactive=FALSE) )
FITS <- lapply(1:2, function(i) ctmm.fit(buffalo[[i]],GUESS[[i]]) )
names(FITS) <- names(buffalo[1:2])

# Gaussian overlap between these two buffalo
overlap(FITS)

# AKDE overlap between these two buffalo
# create aligned UDs
UDS <- akde(buffalo[1:2],FITS)
# evaluate overlap
overlap(UDS)
```

---

periodogram — *Calculate the Lomb-Scargle periodogram of animal-tracking data*

Description

This function calculates isotropic Lomb-Scargle periodogram (LSP, Scargle, 1982) from a telemetry object. One of two algorithms is used. The slow $O(n^2)$ algorithm vectorizes the exact relations of Scargle (1982), while the fast $O(n \log n)$ algorithm uses the FFT method described in Péron & Fleming et al (2016). The latter method is exact if the data are evenly scheduled, permitting gaps, and otherwise it can be made arbitrarily precise via the `res.time` option.

Usage

```
periodogram(data,CTMM=NULL,dt=NULL,res.freq=1,res.time=1,fast=NULL,axes=c("x","y"))

## S3 method for class 'periodogram'
plot(x,max=FALSE,diagnostic=FALSE,col="black",transparency=0.25,grid=TRUE,...)
```

Arguments

`data`	`telemetry` data object or list of such objects.
`CTMM`	An optional `ctmm` model object for specifying the mean.
`dt`	Sampling interval for frequency cutoff.
`res.freq`	Multiplier to inflate the frequency resolution.
`res.time`	Integer multiplier to inflate the temporal resolution. Useful when `fast`>0 and the sampling rate is variable.
`fast`	Use the exact algorithm if FALSE, the FFT algorithm if TRUE, and further inflate the frequency resolution to a power of two sample size if `fast=2`.
`axes`	Array of axes to calculate an average (isotropic) variogram for.
`x`	Output object of `periodogram`.
`max`	Plot only the local maxima of the periodogram. Use only with `res>1`.
`diagnostic`	Plot the sampling schedule's periodogram to check for spurious periodicities.
`col`	Color of periodogram.
`transparency`	Adds transparency to clustered data if greater than zero. Should be less than one.
`grid`	Whether or not to plot gridlines at common periodicities.
`...`	Optional arguments fed to `plot`.

Details

If no `dt` is specified, the median sampling interval is used. This is typically a good assumption for most data, even when there are gaps and this choice corresponds to the discrete Fourier transform (DFT) periodogram for evenly-sampled data.

At default resolution the frequency grid interval is given by `1/(2*(range(data$t)+dt))` and the frequency cutoff is given by `1/(2*dt)`, both in accordance with the DFT periodogram. Increasing `res.freq` beyond `res.freq=1` will make for a smooth periodogram, but sequential frequencies will be highly correlated. The `max=TRUE` option to `plot.periodogram` may be useful for `res.freq>1`. Increasing `res.time` beyond `res.time=1` is helpful if there is variability in the sampling rate and `fast>0`.

If a `CTMM` argument is provided, the ML mean will be detrended from the data prior to calculating the periodogram. Otherwise, the sample mean will be detrended.

If a list of `telemetry` objects are fed into `periodogram`, then a mean `periodogram` object will be returned with the default `dt` and base frequency resolution selected on a worst case basis according to the method described by Péron & Fleming et al (2016).

Value

Returns a periodogram object (class `periodogram`) which is a dataframe containing the frequency, `f` and the Lomb-Scargle periodogram at that frequency, `LSP`.

Note

The LSP is totally inappropriate if you in any way alter the sampling rate within the dataset. Stick with variograms in that case. There is a `diagnostic` option in `plot.periodogram` that can check for spurious periodicities that result from an autocorrelated sampling schedule. This plot will not contain any periodicities if the LSP is appropriate.

`res.time>1` relies on Lagrange interpolation of the sinusoids (not the data), which can suffer from Runge's phenomena. `periodogram` tests for an invalid result and can fail with an error message. For whatever reason, this more frequently seems to happen when `res.time=3`.

Author(s)

C. H. Fleming and G. Péron

References

J. D. Scargle. Studies in astronomical time-series analysis. II. Statistical aspects of spectral analysis of unevenly-sampled data. The Astrophysical Journal, 263, 835-853 (1952).

G. Péron, C. H. Fleming, R. C. de Paula, J. M. Calabrese. Uncovering periodic patterns of space use in animal tracking data with periodograms, including a new algorithm for the Lomb-Scargle periodogram and improved randomization tests. Movement Ecology, 4:19, DOI:10.1186/s40462-016-0084-7 (2016).

Examples

```
#Load package and data
library(ctmm)
data(wolf)

#Extract movement data for a single animal
Tay <- wolf$Tay

#Calculate periodogram (fast==2 for a speedy example)
#There is some variability in the sampling frequency, so we increase res.time
LSP <- periodogram(Tay,fast=2,res.time=2)

#Plot the periodogram
plot(LSP,max=TRUE)
```

plot.telemetry *Plotting methods for telemetry objects.*

Description

Produces simple plots of telemetry objects, possibly overlayed with a Gaussian ctmm movement model or a UD utilization distribution.

Usage

```
plot(x,y,...)

## S3 method for class 'telemetry'
plot(x,CTMM=NULL,UD=NULL,level.UD=0.95,level=0.95,DF="CDF",error=TRUE,velocity=FALSE,
     units=TRUE,col="red",col.level="black",col.DF="blue",col.grid="white",
    transparency.error=0.25,pch=1,type='p',labels=NULL,fraction=1,add=FALSE,xlim=NULL,
     ylim=NULL,ext=NULL,cex=NULL,lwd=1,lwd.level=1,...)

## S4 method for signature 'list'
zoom(x,...)

## S4 method for signature 'telemetry'
zoom(x,fraction=1,...)

## S4 method for signature 'UD'
zoom(x,fraction=1,...)
```

Arguments

x	telemetry or UD object.
y	Unused option.
CTMM	Optional Gaussian ctmm movement model from the output of ctmm.fit or list of such objects.
UD	Optional UD object such as from the output of akde or list of such objects.
level.UD	Confidence level of Gaussian ctmm model or UD estimate contours to be displayed. I.e., level.UD=0.50 can yield the 50% core home range within the rendered contours.
level	Confidence levels placed on the contour estimates themselves. I.e., the above 50% core home-range area can be estimated with 95% confidence via level=0.95.
DF	Plot the maximum likelihood probability density function "PDF" or cumulative distribution function "CDF".
error	Plot error circles/ellipses if present in the data. error=2 will fill in the circles and error=3 will plot densities instead.
velocity	Plot velocity vectors if present in the data.

`units`	Convert axes to natural units.
`col`	Color option for telemetry data. Can be an array or list of arrays.
`col.level`	Color option for home-range contours. Can be an array.
`col.DF`	Color option for the density function. Can be an array.
`col.grid`	Color option for the maximum likelihood `akde` bandwidth grid.
`transparency.error`	Transparency scaling for erroneous locations when `error=1:2`. `trans=0` disables transparancy. Should be no greater than `1`.
`pch`	Plotting symbol. Can be an array or list of arrays.
`type`	How plot points are connected. Can be an array.
`labels`	Labels for UD contours. Can be an array or list of arrays.
`fraction`	Quantile fraction of the data, Gaussian `ctmm`, or `UD` range to plot, whichever is larger.
`add`	Setting to `TRUE` will disable the unit conversions and base layer plot, so that `plot.telemetry` can be overlayed atop other outputs more easily.
`xlim`	The `x` limits `c(x1, x2)` of the plot (in SI units).
`ylim`	The `y` limits `c(y1, y2)` of the plot (in SI units).
`ext`	Plot extent alternative to `xlim` and `ylim` (see `extent`).
`cex`	Relative size of plotting symbols. Only used when errors are missing.
`lwd`	Line widths of `telemetry` points.
`lwd.level`	Line widths of `UD` contours.
`...`	Additional options passed to `plot`.

Details

Confidence intervals placed on the `ctmm` Gaussian home-range contour estimates only represent uncertainty in the area's magnitude and not uncertainty in the mean location, eccentricity, or orientation angle. For `akde` `UD` estimates, the provided contours also only represent uncertainty in the magnitude of the area. With `akde` estimates, it is also important to note the scale of the bandwidth and, by default, grid cells are plotted with `akde` contours such that their length and width matches that of a bandwidth kernels' standard deviation in each direction. Therefore, this grid provides a visual approximation of the kernel-density estimate's “resolution”.

Value

Returns a plot of x vs. y, and, if specified, Gaussian `ctmm` distribution or `UD`. `akde` `UD` plots also come with a standard resolution grid. `zoom` includes a zoom slider to manipulate `fraction`.

Note

If `xlim` or `ylim` are provided, then the smaller or absent range will be expanded to ensure `asp=1`.

Author(s)

C. H. Fleming.

See Also

akde, ctmm.fit, plot, SpatialPoints.telemetry.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# Plot the data
plot(buffalo,col=rainbow(length(buffalo)))
```

---

plot.variogram *Plotting methods for variogram objects.*

Description

Produces simple plots of `varigram` objects (semi-variance vs. time lag) and model semi-variance functions, with approximate confidence intervals around the semi-variance estimates.

Usage

```
## S3 method for class 'variogram'
plot(x,CTMM=NULL,level=0.95,units=TRUE,fraction=0.5,col="black",col.CTMM="red",xlim=NULL,
     ylim=NULL,ext=NULL,...)

## S4 method for signature 'variogram'
zoom(x,fraction=0.5,...)
```

Arguments

`x`	A `variogram` object calculated using `variogram`.
`CTMM`	A `ctmm` movement model object in the same format as the output of `ctmm.fit` or `variogram.fit`.
`level`	Confidence level of confidence bands (95% default CIs). Can be an array.
`units`	Convert axes to natural units.
`fraction`	The proportion of the variogram object, `variogram`, that will be plotted. By convention, half is shown. The tail end is generally garbage.
`col`	Color for the empirical variogram. Can be an array.
`col.CTMM`	Color for the model. Can be an array.
`xlim`	Range of lags to plot (in SI units).
`ylim`	Range of semi-variance to plot (in SI units).
`ext`	Plot extent alternative to `xlim` and `ylim` (see `extent`).
`...`	Additional `plot` function parameters.

Value

Returns a plot of semi-variance vs. time lag, with the empirical variogram in blue and the `ctmm` semi-variance function in red if specified. `zoom` includes a log-scale zoom slider to manipulate `fraction`.

Note

The errors of the empirical variogram are correlated. Smooth trends are not necessarily significant.

Author(s)

J. M. Calabrese and C. H. Fleming

References

C. H. Fleming, J. M. Calabrese, T. Mueller, K.A. Olson, P. Leimgruber, W. F. Fagan. From fine-scale foraging to home ranges: A semi-variance approach to identifying movement modes across spatiotemporal scales. The American Naturalist, 183:5, E154-E167 (2014).

See Also

`correlogram`, `ctmm.fit`, `plot`, `variogram`, `variogram.fit`.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# Extract movement data for a single animal
Cilla <- buffalo$Cilla

# Calculate variogram
SVF <- variogram(Cilla)

# Plot the variogram
plot(SVF)
```

`projection` *Projection*

Description

Functions to manipulate the coordinate reference system (CRS) of ctmm objects

Usage

```
## S4 method for signature 'telemetry'
projection(x,asText=TRUE)

## S4 method for signature 'ctmm'
projection(x,asText=TRUE)

## S4 method for signature 'UD'
projection(x,asText=TRUE)

## S4 method for signature 'list'
projection(x,asText=TRUE)

## S4 method for signature 'NULL'
projection(x,asText=TRUE)

## S4 replacement method for signature 'telemetry'
projection(x) <- value

## S4 replacement method for signature 'list'
projection(x) <- value

## S3 method for class 'telemetry'
median(x,na.rm=FALSE,...)
```

Arguments

`x`	A `telemetry`, `ctmm`, or `UD` object.
`asText`	If `TRUE`, the projection is returned as text. Otherwise a `CRS` object is returned.
`value`	Projection to apply. Can also be a data.frame of longitude-latitude foci.
`na.rm`	Not used.
`...`	Arguments passed to `Gmedian`.

Details

`projection(x)` returns the projection information from ctmm object `x`, while `projection(x) <- value` applies the projection `value` to object `x`. `median(x)` returns the ellipsoidal geometric median of a telemetry object.

Author(s)

C. H. Fleming

See Also

`as.telemetry`.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# Apply a 1-point projection that preserves North==up
projection(buffalo) <- median(buffalo)
plot(buffalo)

# Apply a 2-point projection better for elongated disributions
projection(buffalo) <- median(buffalo,k=2)
plot(buffalo)
# This is the default projection for ctmm
```

`residuals.ctmm` *Calculate model fit residuals and assess their autocorrelation*

Description

These functions calculate the residuals of a CTMM or UERE calibration model, which should be standardized and IID if the model correctly specified. A correlogram method is also provided to assess autocorrelation. This function is analogous to `acf`, but can handle missing data and multiple dimensions.

Usage

```
## S3 method for class 'ctmm'
residuals(object,data,...)

## S3 method for class 'telemetry'
residuals(object,CTMM=NULL,...)

correlogram(data,dt=NULL,fast=TRUE,res=1,axes=c("x","y"))
```

Arguments

`object`	`ctmm` model object or `telemetry` data object for calculating residuals.
`data`	`telemetry` data object or `data.frame` with time column `t` and data columns `axes`.
`CTMM`	`ctmm` model object. If `NULL`, the data is treated as (calibrated) calibration data.
`...`	Unused arguments.
`dt`	Lag bin width. An ordered array will yield a progressive coarsening of the lags. Defaults to the median sampling interval.
`fast`	Use the lag-weighted algorithm if `FALSE` or the FFT algorithm if `TRUE`. The slow algorithm outputs a progress bar.
`res`	Increase the discretization resolution for irregularly sampled data with `res>1`. Decreases bias at the cost of smoothness.
`axes`	Array of axes to calculate an average (isotropic) correlogram for.

Details

Given a telemetry dataset and ctmm model, residuals calculates the standardized residuals of the Kalman filter, which can be tested for independence. The residuals object can then be plotted with plot or fed into the correlogram method to test independence. Output of the correlogram can then be plotted as well, though zoom is much more useful.

When calculating correlograms, minimizing bias is more important than producing a overall smooth estimate. If fast=TRUE, then res needs to be large enough to resolve variability in the sampling interval (missing data is permitted). E.g., if the sampling interval is set to 15 minutes, but can be off by a minute or two, then res=15 is a good choice.

Value

residuals return a residual object (class telemetry, but flagged as residual) and correlogram returns a correlogram object (class variogram, but flagged as an ACF).

Note

If the sampling schedule is irregular, permitting gaps, then the correlogram may not look good even if the model is correctly specified. In this case the correlogram of the residuals should be compared to the correlogram of simulated residuals, using "data" simulated from the fit model and with the same sampling schedule.

Author(s)

C. H. Fleming

References

C. H. Fleming, D. Sheldon, E. Gurarie, W. F. Fagan, S. LaPoint, J. M. Calabrese. Kálmán filters for continuous-time movement models. Ecological Informatics, 40, 8-21 (2017).

See Also

plot.variogram, variogram.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)
Cilla <- buffalo$Cilla

# fit a model
GUESS <- ctmm.guess(Cilla,interactive=FALSE)
FIT <- ctmm.fit(Cilla,GUESS)

# calculate residuals
RES <- residuals(Cilla,FIT)
```

```
# scatter plot of residuals with 50% and 95% quantiles
plot(RES,col.DF=NA,level.UD=c(.50,.95))

# calculate correlogram of residuals
# increase the res argument to account for sampling variability
ACF <- correlogram(RES,res=10)

# plot 4 day's worth of lags
plot(ACF[ACF$lag<=4 %#% 'day',],fraction=1)
```

`select` *Spatial selection methods for telemetry objects.*

Description

Methods to segment or subset telemety objects based on polygon lasso, rectangular marquee, and time slider selectors.

Usage

```
lasso(object,...)

marquee(object,...)

cleave(object,fraction=0.5,name="CLEFT",...)
```

Arguments

`object`	`telemetry` object or list of such objects.
`fraction`	Initial split index, as fraction of total (will change to time).
`name`	Name of list to store cleft telemetry objects to.
`...`	Additional arguments passed to `plot`.

Details

`lasso` and `marquee` allow the user to subset telemetry data into two groups (interior and exterior), based on a hand-drawn polygon lasso or rectangular marquee. `cleave` allows the user to split the data into two halves at a particular time selected via slider.

Value

`lasso` and `marquee` return a named list telemetry objects, twice the length of the input `object`, where the first half are the interior subsets and the second half are the exterior subsets. `cleave` stores a similar list of telemetry objects to `name` on button press.

Author(s)

C. H. Fleming.

See Also

plot.telemetry

Examples

```
# This example is interactive
if(interactive())
{
  # Load package and data
  library(ctmm)
  data(wolf)

  # Extract wolf Luna
  Luna <- wolf$Luna

  # Select resident data
  Luna.sub <- lasso(Luna)

  # You can now work with the resident and dispersive data separately
  names(Luna.sub)
}
```

---

simulate.ctmm *Predict or simulate from a continuous-time movement model*

Description

Given a ctmm movement model (and optional telemetry data to condition upon) these functions predict or simulate animal locations over a prescribed set of times.

Usage

```
predict(object,...)

## S3 method for class 'ctmm'
predict(object,data=NULL,t=NULL,dt=NULL,res=1,complete=FALSE,...)

## S3 method for class 'telemetry'
predict(object,CTMM=NULL,t=NULL,dt=NULL,res=1,complete=FALSE,...)

simulate(object,nsim=1,seed=NULL,...)

## S3 method for class 'ctmm'
simulate(object,nsim=1,seed=NULL,data=NULL,t=NULL,dt=NULL,res=1,complete=FALSE,
         precompute=FALSE,...)

## S3 method for class 'telemetry'
simulate(object,nsim=1,seed=NULL,CTMM=NULL,t=NULL,dt=NULL,res=1,complete=FALSE,
         precompute=FALSE,...)
```

Arguments

`object`	A `ctmm` movement-model or `telemetry` object, which requires an additional `CTMM` argument.
`data`	Optional `telemetry` object on which the prediction or simulation will be conditioned.
`t`	Optional array of numeric time values over which the process will be predicted or simulated.
`dt`	Timestep to space the prediction or simulation over if `data` is specified.
`res`	Average number of locations to predict or simulate per `data` time.
`complete`	Additionally calculate timestamps and geographic coordinates.
`CTMM`	A `ctmm` movement-model in the same format as the output of `ctmm.fit` or `variogram.fit`.
`nsim`	Not yet supported.
`seed`	Optional random seed to fix.
`precompute`	Precalculate matrices of the Kalman filter (see details).
`...`	Unused options.

Details

The prediction or simulation necessarily requires a `ctmm` model object. If a `telemetry` `data` object is supplied, the output will be conditional on the `data` (i.e., simulations that run through the data). If no `data` is provided then the output will be purely Gaussian, and times `t` must be provided. Details of the movement model parameters can be found in `ctmm.fit`.

The `t` argument fixes the output times to a specific array of times. The `dt` and `res` arguments are relative to the sampling schedule present in the optional `telemetry` object. The same span of time will be used, while `dt` will fix the sampling rate absolutely and `res` will fix the sampling rate relative to that of the data.

The `precompute` option can speed up calculations of multiple simulations of the same model, data, and *irregular* sampling schedule. First run `simulate` with `precompute=TRUE` to calculate and store all of the necessary matrices of the Kalman filter. A simulated `telemetry` object will be produced, as usual, and the precomputed objects are stored in the environment. Subsequent simulations with `precompute=-1` will then apply these precomputed matrices for a computational cost savings. If the sampling schedule is irregular, then this can result in faster simulations.

Value

A simulated animal-tracking `telemetry` object with components `t`, `x`, and `y`, or a predicted `telemetry` object that also includes `x`-`y` covariances for the location point estimates `x` and `y`.

Note

Predictions are autocorrelated and should not be treated as data.

Author(s)

C. H. Fleming.

References

C. H. Fleming, J. M. Calabrese, T. Mueller, K.A. Olson, P. Leimgruber, W. F. Fagan. From fine-scale foraging to home ranges: A semi-variance approach to identifying movement modes across spatiotemporal scales. The American Naturalist, 183:5, E154-E167 (2014).

C. H. Fleming, D. Sheldon, E. Gurarie, W. F. Fagan, S. LaPoint, J. M. Calabrese. Kálmán filters for continuous-time movement models. Ecological Informatics, 40, 8-21 (2017).

See Also

ctmm.fit

Examples

```
#Load package
library(ctmm)

#prepare simulation parameters
t <- 1:1000
MODEL <- ctmm(tau=c(100,10),sigma=10,mu=c(0,0))

#simulate data
SIM <- simulate(MODEL,t=t)

#plot data with Gaussian model
plot(SIM,CTMM=MODEL)
```

---

speed *Estimate the average speed of a tracked animal*

Description

Given a `ctmm` movement model and `telemetry` data, speed simulates multiple realizations of the individual's trajectory to estimate the time-averaged speed, which is proportional to distance traveled, while speeds estimates instantaneous speeds at a specified array of times `t`. Both tortuosity (non straight-line motion between the data) and telemetry error can be accounted for. Given only a `ctmm` movement model and no data, speed calculates the average speed of the Gaussian movement process.

Usage

```
speed(object,...)

## S3 method for class 'ctmm'
speed(object,data=NULL,level=0.95,robust=FALSE,units=TRUE,prior=TRUE,fast=TRUE,
      cor.min=0.5,dt.max=NULL,error=0.01,cores=1,...)

## S3 method for class 'telemetry'
```

```
speed(object,CTMM,level=0.95,robust=FALSE,units=TRUE,prior=TRUE,fast=TRUE,cor.min=0.5,
      dt.max=NULL,error=0.01,cores=1,...)

speeds(object,...)

## S3 method for class 'ctmm'
speeds(object,data=NULL,t=NULL,cycle=Inf,level=0.95,robust=FALSE,prior=FALSE,fast=TRUE,
       error=0.01,cores=1,...)

## S3 method for class 'telemetry'
speeds(object,CTMM,t=NULL,cycle=Inf,level=0.95,robust=FALSE,prior=FALSE,fast=TRUE,
       error=0.01,cores=1,...)
```

Arguments

`object`	A `ctmm` movement-model or `telemetry` object, which requires an additional `CTMM` argument.
`data`	Optional `telemetry` object on which the simulations will be conditioned.
`CTMM`	Movement model object.
`t`	Array of times to estimate instantaneous speeds at.
`cycle`	Average over time `t` indices modulo `cycle`. E.g., for `t` sequenced by hours, `cycle=24` gives daily the cycle of speeds. (Not yet supported.)
`level`	Confidence level to report on the estimated average speed.
`robust`	Use robust statistics for the ensemble average and its confidence intervals (see Details).
`units`	Convert result to natural units.
`prior`	Account for model uncertainty.
`fast`	Whether or not to invoke the central-limit theorem (see `emulate`) when `prior=TRUE`.
`cor.min`	Velocity correlation threshold for skipping gaps.
`dt.max`	Absolute gap sizes to skip (in seconds), alternative to `cor.min`.
`error`	Target (relative) standard error.
`cores`	Number of simulations to run in parallel. `cores=0` will use all cores, while `cores<0` will reserve `abs(cores)`.
`...`	Arguments passed to `emulate`.

Details

The `cor.min` or `dt.max` arguments are used to constrain the estimate to be derived from simulations near the data, and therefore ensure that the estimate is more reflective of the data than the model.

If data quality is poor and velocity can barely be resolved, then the sampling distribution may occassionally include impersistent motion and the ensemble mean will be infinite. In these cases `robust=TRUE` can be used to report the ensemble median rather than the ensemble mean. The time average of `speed`, in either case, is still the mean and the resulting quantity is still proportional to distance traveled. Furthermore, note that medians should be compared to medians and means to means, so the `robust` option should be the same for all compared individuals.

Value

Returns the estimated mean speed of the sampled trajectory with CIs by default. If `level=NULL`, then the ensemble of mean speeds is returned instead.

Note

The average speeds estimated here are mean speeds. The speeds reported by `summary.ctmm` are root-mean-square (RMS) speeds. These quantities are sometimes proportional, but not equivalent.

Author(s)

C. H. Fleming.

See Also

`emulate`, `simulate`

Examples

```
# Load package and data
library(ctmm)
data(buffalo)
Gabs <- buffalo$Gabs

GUESS <- ctmm.guess(Gabs,interactive=FALSE)
FIT <- ctmm.fit(Gabs,GUESS)

# stationary Gaussian estimate
speed(FIT)

# conditional estimate
speed(FIT,Gabs)
```

`summary.ctmm` *Summarize a continuous-time movement model*

Description

This function returns a list of biologically interesting parameters in human readable format, as derived from a continuous-time movement model.

Usage

```
## S3 method for class 'ctmm'
summary(object,level=0.95,level.UD=0.95,units=TRUE,IC="AICc",MSPE="position",...)
```

Arguments

`object`	A `ctmm` movement-model object from the output of `ctmm.fit`.
`level`	Confidence level for parameter estimates.
`level.UD`	Coverage level for the Gaussian home-range area.
`units`	Convert result to natural units.
`IC`	Information criteria for sorting lists of `ctmm` objects. Can be `"AICc"`, `"AIC"`, `"BIC"` or none (`NA`). AICc is approximate.
`MSPE`	Sort models with the same autocovariance structure by the mean square predictive error of `"position"`, `"velocity"`, or not (`NA`).
`...`	Unused options.

Value

If summary is called with a single `ctmm` object output from `ctmm.fit`, then a list is returned with the effective sample sizes of various parameter estimates (`DOF`) and a parameter estimate table `CI`, with low, maximum likelihood, and high estimates for the following possible parameters:

`tau` The autocorrelation timescales. `tau position` is also the home-range crossing timescale.

`area` The Gaussian home-range area, where the point estimate has a significance level of `level.UD`. I.e., the core home range is where the animal is located 50% of the time with `level.UD=0.50`. This point estimate itself is subject to uncertainty, and is given confidence intervals derived from `level`.

This Gaussian estimate differs from the kernel density estimate of `summary.UD`. The Gaussian estimate has more statistical efficiency, but is less related to space use for non-Gaussian processes.

`speed` The Gaussian root-mean-square (RMS) velocity, which is a convenient measure of average speed but not the conventional measure of average speed (see `speed`).

If summary is called on a list of `ctmm` objects output from `ctmm.select`, then a table is returned with the model names and IC differences for comparison across autocovariance structures. Given non-stationary models and `MSPE>0`, the mean square prediction error (MSPE) is also returned for comparison across trend structures (with autocovariance structure fixed). For the model names, "IID" denotes the uncorrelated bi-variate Gaussian model, "OU" denotes the continuous-position Ornstein-Uhlenbeck model, and "OUF" denotes the continuous-velocity Ornstein-Uhlenbeck-F model.

Note

Confidence intervals on the autocorrelation timescales assume they are sufficiently greater than zero and less than infinity.

In `ctmm` v0.5.1 the MSPE was averaged over all possible times instead of over all sampled times.

In `ctmm` v0.3.4 the speed estimate was fixed to be the RMS velocity and not $1/\sqrt{2}$ times the RMS velocity.

Author(s)

C. H. Fleming.

See Also

ctmm.fit, ctmm.select.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# Extract movement data for a single animal
Cilla <- buffalo$Cilla

# fit model
GUESS <- ctmm.guess(Cilla,interactive=FALSE)
FIT <- ctmm.fit(Cilla,GUESS)

# Tell us something interpretable
summary(FIT)
```

---

summary.UD *Summarize a range distribution*

Description

This function returns a list of biologically interesting parameters in human readable format, as derived from an autocorrelated kernel density estimate.

Usage

```
## S3 method for class 'UD'
summary(object,level=0.95,level.UD=0.95,units=TRUE,...)
```

Arguments

object	An akde autocorrelated kernel-density estimate from the output of akde.
level	Confidence level for the above area estimate. E.g., the 95% confidence interval of the 50% core area.
level.UD	Coverage level for the home-range area. E.g., the 50% core area.
units	Convert result to natural units.
...	Unused options.

Value

A `matrix` with low, maximum likelihood, and high estimates for the following parameters:

`area` The home-range area with fraction of inclusion `level.UD`. E.g., the 50% core home range is estimated with `level.UD=0.50`, and 95% confidence intervals are placed on that area estimate with `level=0.95`.

This kernel density estimate differs from the Gaussian estimate of `summary.ctmm`. The Gaussian estimate has more statistical efficiency, but is less related to space use for non-Gaussian processes.

Note

Prior to `ctmm` v0.3.1, AKDEs included only errors due to autocorrelation uncertainty, which are insignificant in cases such as IID data. Starting in v0.3.1, `akde` calculated an effective sample size `DOF.H` and used this to estimate area uncertainty under a chi-square approxmation. Starting in v0.3.2, this method was improved to use `DOF.area` in the Gaussian reference function approximation.

Author(s)

C. H. Fleming.

References

C. H. Fleming, J. M. Calabrese. A new kernel-density estimator for accurate home-range and species-range area estimation. Methods in Ecology and Evolution, 8:5, 571-579 (2016).

See Also

`akde`.

Examples

```
# Load package and data
library(ctmm)
data(buffalo)

# Extract movement data for a single animal
Cilla <- buffalo$Cilla

# Fit a movement model
GUESS <- ctmm.guess(Cilla,interactive=FALSE)
FIT <- ctmm.fit(Cilla,GUESS)

# Estimate and summarize the AKDE
UD <- akde(Cilla,FIT)
summary(UD)
```

turtle — *Wood turtle GPS and calibration dataset from Working Land and Seascapes.*

Description

x-y projected GPS data from 2 calibration runs and 2 wood turtles. Please contact Tom Akre (akret@si.edu) if you want to publish with these data.

Usage

```
data("turtle")
```

Format

A list of 4 telemetry objects.

See Also

as.telemetry, plot.telemetry, uere, buffalo, coati, gazelle, wolf.

Examples

```
# Load package and data
library(ctmm)
data("turtle")

# Plot a turtle's locations
plot(turtle[[3]])
```

uere — *Estimate UERE from calibration data*

Description

Functions for estimating and assigning the User Equivalent Range Error (UERE) of a GPS device from calibration data.

Usage

```
uere(data)

uere(data) <- value

uere.fit(data,precision=1/2)

## S3 method for class 'UERE'
summary(object,level=0.95,...)
```

Arguments

`data`	`telemetry` object or list of telemetry objects, preferably with DOP columns.
`value`	UERE value(s) to assign to telemetry data (see details).
`precision`	Fraction of maximum possible digits of precision to target in categorical error fitting. `precision=1/2` results in about 7 decimal digits of precision.
`object`	`UERE` object to summarize or list of `UERE` objects to compare.
`level`	Confidence level for UERE estimate confidence intervals.
`...`	Further arguments are ignored.

Details

Often times GPS animal tracking devices return HDOP values but do not specifiy the device's UERE necessary to transform the HDOP values into absolute errors. `uere.fit()` allows users to estimate the UERE from calibration data, where the device was left fixed over a period of time. The calibration UERE can then be applied to tracking data with the `uere()<-` assignment method. Otherwise, when `error=TRUE` in `ctmm`, `ctmm.fit` will estimate the UERE simultaneously with the movement model, which is less reliable than using calibration data.

`summary()` applied to single `UERE` object will return UERE parameter estimates and confidence intervals, while `summary()` applied to a list of `UERE` objects will return a model-selection table, with AICc and reduced Z squared (goodness of fit) values.

Value

The UERE estimate.

Note

The GPS device should be fixed during calibraiton.

Author(s)

C. H. Fleming

See Also

`as.telemetry`, `residuals.telemetry`.

Examples

```
# Load package and data
library(ctmm)
data(turtle)

# the first two datasets are calibration data
names(turtle)

# estimate UERE from calibration data
UERE <- uere.fit(turtle[1:2])
```

```
# inspect UERE estimate
summary(UERE)

# assign UERE to entire dataset
uere(turtle) <- UERE

# calculate residuals of calibration data
RES <- lapply(turtle[1:2],residuals)

# scatter plot of residuals with 50% and 95% quantiles
plot(RES,col.DF=NA,level.UD=c(0.50,0.95))

# check calibration data for autocorrelation using fast=FALSE because samples are small
ACFS <- lapply(RES,function(R){correlogram(R,fast=FALSE,dt=10 %#% 'min')})

# pooling ACFs
ACF <- mean(ACFS)

plot(ACF)
```

`Unit conversion` *Convert dimensionful quantities to and from SI units*

Description

This function takes a number in some specified units and converts that number to SI units, or from SI units to the specified units. Internally, all `ctmm` objects are specified in SI units, and so this is a utility function to facilitate working with `ctmm` objects.

Usage

```
x %#% y
```

Arguments

`x`	A numeric quantity specified in `y` character labeled units, or a character unit label to convert a numeric quantity `y` that is specified in SI units.
`y`	A unit character label for the quantity `x` to be converted to SI units, or a numeric quantity in SI units to be converted into unit label `x`.

Details

If `x` is a number and `y` is a character unit label, then `x` is converted from units `y` to SI units. If `x` is a character unit label and `y` is a number, then `y` is converted from SI units to units `x`.

Value

Returns a numeric in SI units or units specified by character label `x`.

Note

Dimensions of weight are not currently supported as they are not used by `ctmm`.

Author(s)

C. H. Fleming.

Examples

```
# one yard -> meters
1 %#% "yard"

# one meter -> yards
"yard" %#% 1

# 1 month -> days
"day" %#% 1 %#% "month"

# 6 miles per hour -> meters per second
"hour" %#% 6 %#% "mile"
```

`variogram` *Calculate an empirical variogram from movement data*

Description

This function calculates the empirical variogram of multi-dimensional tracking data for visualizing stationary (time-averaged) autocorrelation structure. One of two algorithms is used. The slow $O(n^2)$ algorithm is based upon Fleming & Calabrese et al (2014), but with interval-weights instead of lag-weights. Additional modifications have also been included to accommodate drift in the sampling rate. The fast $O(n \log n)$ algorithm is based upon the FFT method of Marcotte (1996), with some tweaks to better handle irregularly sampled data. Both methods reduce to the unbiased "method of moments" estimator in the case of evenly *scheduled* data, even with missing observations, but they produce slightly different outputs for irregularly sampled data.

Usage

```
variogram(data,dt=NULL,fast=TRUE,res=1,CI="Markov",axes=c("x","y"))
```

Arguments

`data`	`telemetry` data object of the 2D timeseries data.
`dt`	Lag bin width. An ordered array will yield a progressive coarsening of the lags. Defaults to the median sampling interval.
`fast`	Use the interval-weighted algorithm if `FALSE` or the FFT algorithm if `TRUE`. The slow algorithm outputs a progress bar.

res	Increase the discretization resolution for irregularly sampled data with res>1. Decreases bias at the cost of smoothness.
CI	Defaults to only consider non-overlapping lags independent. All unique lags are considered independent with "IID".
axes	Array of axes to calculate an average (isotropic) variogram for.

Details

If no dt is specified, the median sampling interval is used. This is typically a good assumption for most data, even when there are gaps. A dt coarser than the sampling interval may bias the variogram (particuarly if fast=TRUE) and so this should be reserved for poor data quality.

For irregularly sampled data, it may be useful to provide an array of time-lag bin widths to progressively coarsen the variogram. I.e., if you made the very bad choice of changing your sampling interval on the fly from dt1 to dt2, where dt1 $<$ dt2, the an appropriate choice would be dt=c(dt1,dt2). On the other hand, if your sampling is itself a noisy process, then you might want to introduce larger and larger dt components as the visual appearance of the variogram breaks down with increasing lags. Alternatively, you might try the fast=FALSE option or aggregating multiple individuals with mean.variogram.

With irregularly sampled data, different size lags must be aggregated together, and with current fast methods there is a tradeoff between bias and smoothness. The default settings produce a relatively smooth estimate, while increasing res (or setting fast=FALSE) will produce a less biased estimate, which is very useful for correlogram.

In standard variogram regression treatments, all lags are considered as independent (CI="IID") for the purposes of confidence-interval estimation, even if they overlap in time. However, in high resolution datasets this will produce vastly underestimated confidence intervals. Therefore, the default CI="Markov" behavior is to consider only the maximum number of non-overlapping lags in calculating confidence intervals.

Value

Returns a variogram object (class variogram) which is a dataframe containing the time-lag, lag, the semi-variance estimate at that lag, SVF, and the approximate number of degrees of freedom associated with that semi-variance, DOF, with which its confidence intervals can be estimated.

Note

Prior to ctmm v0.3.6, fast=FALSE used the lag-weighted esitmator of Fleming et al (2014). Lag weights have been abandoned in favor of interval weights, which are less sensitive to sampling irregularity. The same weighting formulas are used, but with dt instead of the current lag.

Author(s)

C. H. Fleming and J. M. Calabrese.

References

D. Marcotte. Fast variogram computation with FFT. *Computers and Geosciences* 22(10), 1175-1186 (1996).

C. H. Fleming, J. M. Calabrese, T. Mueller, K.A. Olson, P. Leimgruber, W. F. Fagan. From fine-scale foraging to home ranges: A semi-variance approach to identifying movement modes across spatiotemporal scales. The American Naturalist, 183:5, E154-E167 (2014).

See Also

`vignette("variogram")`, `correlogram`, `mean.variogram`, `plot.variogram`, `variogram.fit`.

Examples

```
#Load package and data
library(ctmm)
data(buffalo)

#Extract movement data for a single animal
Cilla <- buffalo$Cilla

#Calculate variogram
SVF <- variogram(Cilla)

#Plot the variogram with 50% and 95% CIs
plot(SVF,level=c(0.5,0.95))
```

`variogram.fit` *Visually fit a movement model to a variogram*

Description

This function plots a `variogram` object overlayed with a continuous-time movement model guesstimated from the variogram's shape. Sliders are given to adjust the parameter guesstimates and the result can be saved to a global variable. The intention of this function is to facilitate good starting guesses for `ctmm.fit`, starting with a prototype hypothesis argument `CTMM`, which can contain features such as `isotropic`, `range`, `circle`, etc..

Usage

```
ctmm.guess(data,CTMM=ctmm(),variogram=NULL,name="GUESS",interactive=TRUE)

variogram.fit(variogram,CTMM=ctmm(),name="GUESS",fraction=0.5,interactive=TRUE,...)
```

Arguments

`data`	A `telemetry` object.
`CTMM`	Optional model prototype or initial guesstimate of the model parameters, in `ctmm` object format.
`name`	Name of the global variable to store the guesstimate in.
`interactive`	Boolean denoting whether to render the initial guess with interactive sliders or store the result silently.

`variogram`	A `variogram` object from the output of `variogram`.
`fraction`	Initial fraction of the variogram to render.
`...`	Optional parameters passed to `plot.variogram`.

Details

By default, `sigma` is the asymptote of the variogram and `tau` is an array of autocorrelation timescales. The position timescale is roughly the time lag it takes of the variogram to reach 63% of its asymptote. The velocity autocorrelation timescale visually corresponds to width of the concave bowl shape at the beginning of the variogram. If `CTMM=ctmm(range=FALSE)`, `sigma` is the asymptotic slope of the variogram and only the velocity timescale is finite.

By default, parameter values are estimated from the shape of the variogram. If this fails, the `CTMM` option can provide alternative initial guesstimates.

Note

If the `manipulate` package is unavailable, then `interactive` is set to `FALSE`.

Author(s)

C. H. Fleming.

See Also

`ctmm.fit`, `plot.variogram`, `variogram`.

Examples

```
#Load package and data
library(ctmm)
data(buffalo)

#Extract movement data for a single animal
Cilla <- buffalo$Cilla

# generate a visual fit of the variogram (requires RStudio or a guess object is returned)
ctmm.guess(Cilla)
```

---

wolf — *Maned wolf GPS dataset from The Maned Wolf Conservation Program.*

Description

x-y projected GPS data on 8 Maned wolves. Please contact Rogerio Cunha de Paula (rogercunha@gmail.com) if you want to publish with these data.

Usage

```
data("wolf")
```

Format

A list of 8 `telemetry` objects.

See Also

`as.telemetry`, `plot.telemetry`, `buffalo`, `coati`, `gazelle`, `turtle`.

Examples

```
# Load package and data
library(ctmm)
data("wolf")

# Plot a wolf's locations
plot(wolf[[8]])
```

Index