Package ‘ctmm’

November 4, 2022

Encoding UTF-8
Version 1.1.0
Date 2022-11-03
Title Continuous-Time Movement Modeling
URL https://github.com/ctmm-initiative/ctmm,
https://groups.google.com/g/ctmm-user
Depends R (>= 3.5.0)
Imports Bessel, data.table, digest, expm, fasttime, Gmedian, graphics,
grDevices, gsl, manipulate, MASS, methods, numDeriv, pbivnorm,
pracma, raster, rgdal, shape, sp, statmod, stats, utils
Suggests animation, bit64, dplyr, fftw, knitr, move, parallel,
parsedate, quadprog, rmarkdown, sf, suncalc
Description Functions for identifying, fitting, and applying continuous-space, continuous-
time stochastic-process movement models to animal tracking data.
The package is described in Calabrese et al (2016) <doi:10.1111/2041-
210X.12559>, with models and methods based on those introduced and detailed in
Fleming & Calabrese et al (2014) <doi:10.1086/675504>,
Fleming et al (2016) <doi:10.1890/15-1607>,
License GPL-3
VignetteBuilder knitr
NeedsCompilation no
RoxygenNote 6.1.1
Author Christen H. Fleming [aut, cre],
Justin M. Calabrese [aut],
Xianghui Dong [ctb],
Kevin Winner [ctb],
Björn Reineking [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],
Autumn-Lynn Harrison [dtc],
Ronaldo G. Morato [dtc]
Maintainer Christen H. Fleming <flemingc@si.edu>
Repository CRAN
Date/Publication 2022-11-04 15:50:02 UTC

R topics documented:

ctmm-package .................................................. 3
akde .......................................................... 6
as.telemetry .................................................. 8
bandwidth ..................................................... 10
buffalo ........................................................ 13
cluster ........................................................ 14
coati .......................................................... 15
color .......................................................... 16
ctmm .......................................................... 18
ctmm-FAQ .................................................... 22
ctmm.boot ................................................... 24
difference ..................................................... 26
distance ........................................................ 28
dt.plot ........................................................ 29
emulate ......................................................... 30
encounter ..................................................... 31
export .......................................................... 32
extent .......................................................... 34
format .......................................................... 36
ctmm-package

Description


Details

Package: ctmm
Type: Package
Version: 1.1.0
Date: 2022-11-03
License: GPL-3

- CTMM Initiative
- Movement of Life
- 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


akde  

*Calculate an autocorrelated kernel density estimate*

**Description**

These functions calculate individual and population-level autocorrelated kernel density home-range estimates from telemetry data and a corresponding continuous-time movement models.

**Usage**

akde(data,CTMM,VMM=NULL,R=list(),SP=NULL,SP.in=TRUE,variable="utilization",debias=TRUE,weights=FALSE,smooth=TRUE,error=0.001,res=10,grid=NULL,...)

pkde(data,UD,kernel="individual",weights=FALSE,ref="Gaussian",...)

**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.

- **R**
  A named list of raster covariates if CTMM contains an RSF model.

- **SP**
  SpatialPolygonsDataFrame object for enforcing hard boundaries.

- **SP.in**
  Locations are assumed to be inside the SP polygons if SP.in=TRUE and outside of SP if SP.in=FALSE.

- **variable**
  Not yet supported.

- **debias**
  Debias the distribution for area estimation (AKDEc).

- **smooth**
  "Smooth" out errors from the data.

- **weights**
  Optimally weight the data to account for sampling bias (See bandwidth for akde details).

- **error**
  Target probability error.

- **res**
  Number of grid points along each axis, relative to the bandwidth.

- **grid**
  Optional grid specification via raster, UD, or list of arguments (See ‘Details’ below).

- **...**
  Arguments passed to bandwidth and akde.

- **UD**
  A list of individual UD objects corresponding to data.

- **kernel**
  Bandwidths are proportional to the individual covariances if kernel="individual" or to the population covariance if kernel="population".

- **ref**
  Include non-Gaussian overlap corrections if ref="AKDE" and weights=TRUE.
Details

For weighted AKDE, please note additional ... arguments passed to `bandwidth`, which can have a large impact on computation time in certain cases.

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.UD` command.

If a UD or raster object is supplied in the grid argument, then the estimate will be calculated on the same grid. Alternatively, a list of grid arguments can be supplied, with any of the following components:

- `r` A list with vectors `x` and `y` that define the grid-cell midpoints.
- `dr` A vector setting the `x` and `y` cell widths in meters. Equivalent to `res` for raster objects.
- `extent` The `x-y` extent of the grid cells, formatted as from the output of `extent`.
- `align.to.origin` Logical value indicating that cell midpoint locations are aligned to be an integer number of `dr` steps from the projection origin.

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.

The PDF estimate is not re-normalized to 1, and may fall short of this by the target numerical error. If inspecting quantiles that are very far from the data, the quantiles may hit the grid boundary or become erratic, making it necessary to reduce the numerical error target. However, default arguments should be able to render any quantiles of reasonable accuracy.

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 approximation. 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


See Also

bandwidth, mean.UD, raster, UD-method, revisitation

Examples

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

# calculate fit guess object
GUESS <- ctmm.guess(DATA,interactive=FALSE)
# in general, you should be running ctmm.select here instead of ctmm.fit
FIT <- ctmm.fit(DATA,GUESS)

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

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

as.telemetry

Import, coerce, summarize, and combine MoveBank data

Description

Functions to import MoveBank csv files, data.frame, and Move objects, coerce them into telemetry objects, summarize them, and combine data from multiple tracking devices.

Usage

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

- **object**: A MoveBank CSV filename, MoveBank data.frame object, or Move object to coerce, or a telemetry object to summarize.
- **timeformat**: Format argument for `strptime`, corresponding to the input data. Alternatively `timeformat="auto"` will attempt to infer the timestamp format with `parse_date`.
- **timezone**: Timezone argument for `strptime`, corresponding to the input data.
- **projection**: Optional PROJ.4 projection argument to be fed to `spTransform`, for the output telemetry object.
- **datum**: Optional PROJ.4 projection argument to specify the input longitude-latitude datum. The default is WGS84.
- **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`.
- **keep**: Retain additional columns after coercion. `keep=TRUE` retains all columns, while individual columns to retain can be specified by name.
- **drop**: Only return a telemetry object for one individual if TRUE. Always return a list of telemetry objects if FALSE.
... telemetry objects or a list of such objects, for `tbind()`. Optional arguments to be fed to `fread` or `read.csv`, in the case of compressed files, for `as.telemetry()`. 

x telemetry object.

n Number of rows to return, if positive, or number of rows to omit, if negative.

Details

For data that have not been corralled through Mobebank, timestamps either need to be provided in a POSIX format (see the output of `Sys.time()`) or supplied with a timeformat argument for interpretation (see `strptime`). Alternatively, you can try your luck with `timeformat="auto"`, and `parse_date` will attempt to infer the format.

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, using `tbind()`.

Value

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

`as.telemetry` will always report the smallest sampling interval, as a message, and the number repeating timestamps, as a warning. Tiny sampling intervals (and repeating timestamps) can sometimes result from misformated timestamps or an incorrect `timeformat` argument. However, even if genuine, they can necessitate data cleaning (outlie) or location-error modeling (`vignette('error')`).

Author(s)


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.
**bandwidth**

**Usage**

```r
bandwidth(data, CTMM, VMM=NULL, weights=FALSE, fast=NULL, dt=NULL, PC="Markov", error=0.01, precision=1/2, verbose=FALSE, trace=FALSE, dt.plot=TRUE,...)
```

**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. `fast=NULL` will attempt to intelligently decide between the fast and exact algorithms based on computational complexity.
- `dt`: Optional lag bin width for the FFT algorithm.
- `PC`: Preconditioner to use: can be "Markov", "circulant", "IID", or "direct".
- `error`: Maximum grid error for FFT algorithm, if `dt` is not specified.
- `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.
- `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.
- `dt.plot`: Execute a diagnostic `dt.plot` with a red line at `dt`, if `weights=TRUE`.
- `...`: Additional arguments not currently used.

**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` is an approximation that discretizes the data with timestep `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, then a choice of `dt` will be automated with a message. **If the data contain some very tiny time intervals**, say 1 second among hourly sampled data, then the default `dt` setting can create an excessively high-resolution discretization of time, which will cause slowdown. In this case `CTMM` should contain a location-error model and `dt` should be increased to a larger fraction of the most-frequent sampling intervals. **If the data are irregular (permitting gaps), then `dt` may need to be several times smaller** than the median to avoid slow down. In this case, try setting `trace=TRUE` and decreasing `dt` below the median until the iterations speed up and the number of feasibility assessments becomes less than $O(n)$.

- `fast=FALSE` uses exact time spacing 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 $O(n^3)$. fast=FALSE, PC='direct' is often the fastest method with small datasets, where $n \leq O(1,000)$, but scales terribly with larger datasets.

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 forcing isotropic=TRUE will provide an inaccurate estimate for very eccentric distributions.

In v1.0.1 the default fast, dt, PC arguments depend on the sample size, with fast=FALSE, PC="Direct" for small sample sizes, fast=FALSE, PC="Markov" for moderate sample sizes, and fast=TRUE, PC="Markov" for large sample sizes, where dt is taken to be the integer fraction of the median sampling interval closest to the minimum sampling interval.

In v0.6.2 the default dt was increased from the minimum time difference to a small quantile no less than error times the median.

Author(s)

C. H. Fleming.

References


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


See Also

as.telemetry, plot.telemetry, coati, gazelle, jaguar, pelican, 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)
cluster (Clustering of movement-model parameters)

Description

These functions cluster and classify individual movement models and related estimates, including AKDE home-range areas, while taking into account estimation uncertainty.

Usage

cluster(x, level=0.95, level.UD=0.95, debias=TRUE, IC="BIC", units=TRUE, plot=TRUE, sort=FALSE, ...)

Arguments

- **x**: A list of `ctmm` movement-model objects, UD objects, or UD summary output, constituting a sampled population, or a list of such lists, each constituting a sampled sub-population.
- **level**: Confidence level for parameter estimates.
- **level.UD**: Coverage level for home-range estimates. E.g., 50% core home range.
- **debias**: Apply Bessel’s inverse-Gaussian correction and various other bias corrections.
- **IC**: Information criterion to determine whether or not population variation can be estimated. Can be "AICc", AIC, or "BIC".
- **units**: Convert result to natural units.
- **plot**: Generate a meta-analysis forest plot with two means.
- **sort**: Sort individuals by their point estimates in forest plot.
- **...**: Further arguments passed to plot.

Details

So-far only the clustering of home-range areas is implemented. More details will be provided in an upcoming manuscript.

Value

A list with elements `P` and `CI`, where `P` is an array of individual membership probabilities for sub-population 1, and `CI` is a table with rows corresponding to the sub-population means, coefficients of variation, and membership probabilities, and the ratio of sub-population means.

Note

The AICc formula is approximated via the Gaussian relation.

Author(s)

C. H. Fleming.
coati

See Also

akde, ctmm.fit, meta.

Examples

# load package and data
library(ctmm)
data(buffalo)

# fit movement models
FITS <- AKDES <- list()
for(i in 1:length(buffalo))
{
  GUESS <- ctmm.guess(buffalo[i],interactive=FALSE)
  # use ctmm.select unless you are certain that the selected model is OUF
  FITS[[i]] <- ctmm.fit(buffalo[i],GUESS)
}

# calculate AKDES on a consistent grid
AKDES <- akde(buffalo,FITS)

# color to be spatially distinct
COL <- color(AKDES,by='/quotesingle.Var
individual'/quotesingle.Var)

# plot AKDEs
plot(AKDES,col.DF=COL,col.level=COL,col.grid=NA,level=NA)

# cluster-analysis of buffalo
cluster(AKDES,sort=TRUE)

cinati

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, jaguar, pelican, 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.
**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 calendar 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**

```r
# 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 hypothetical movement models (with initial estimates), fit those models to the data, and select among those models via an information criterion. 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="pHREML",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
\((2\pi)\) divided by) 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 for the optimizer.
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 criterion used for selection. Can be "AICc", "AIC", "BIC", "LOOCV", "HSCV", 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 additional 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 \texttt{ctmm.fit} will fall back to \texttt{HREML}, as reported by the method slot of the resulting fit object.

The \texttt{control} list can take the following arguments, with defaults shown:

- \texttt{method="pNewton"} The partial-Newton method of \texttt{optimizer} is default. See \texttt{optim} for alternative methods in multiple dimensions.
- \texttt{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, \texttt{precision=1} is approximately 16 decimal digits of precision for the likelihood and 8 for the MLEs.
- \texttt{maxit=.Machine$integer.max} Maximum number of iterations allowed for optimization.

Model selection in \texttt{ctmm.select} proceeds in two phases. If there are a large number of parameters that must be fit numerically (such as when error is modeled), then the target model (argument \texttt{CTMM}) is worked toward by first fitting simpler, compatible models. The second phase proceeds by attempting to simplify the autocorrelation model and complexify the deterministic (trend) model until the information criterion fails to improve. The intent of working in these directions is to improve numerical convergence and 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 \texttt{level} argument. \texttt{level=1} will, therefore, always attempt a simpler model.

The leave-one-out cross validation IC, IC=\texttt{"LOOCV"}, is (-2 times) the sum of log-likelihoods of the validation data, after fitting to and conditioning on the training data. This information criterion is intended for small amounts of data where AIC/BIC are not valid, and where the questions of interest are targeted at the finest scales of the data, such as speed or occurrence. Unlike other model-selection criteria, the computational complexity of LOOCV is $O(n^2)$, which is very slow for sample sizes on the order of 10-100 thousand locations. Furthermore, as autocorrelation in the validation data is ignored, this information criterion is not valid for making inferences at scales coarser than the sampling interval, such as home range.

The half-sample cross validation IC, IC=\texttt{"HSCV"}, is (-2 times) the sum of log-likelihoods of the validation data, after fitting to and conditioning on the training data consisting of the first (and second) halves of the data when split temporally. This information criterion is intended for when few range crossings are observed and AIC/BIC may not be valid.

**Value**

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

- \texttt{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.
- \texttt{loglike} The log-likelihood.
- \texttt{sigma} The maximum likelihood variance/covariance estimate (possibly debiased). For the end-lessly diffusing BM and IOU processes, this is instead the diffusion rate estimate.
- \texttt{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 \(c(\sigma, \tau, \text{circle})\), as derived (asymptotically) from the hessian of the log-likelihood function, and where \(\sigma\) is parameterized in terms of its largest variance \(\text{major}\), the ratio of the smallest to largest variance \(\text{minor}\), and angle of orientation. Typically, \(\sigma\)'s \(\text{major}\) parameter is extremely correlated to \(\tau[1]\), and sequential components of \(\tau\) are slightly correlated.

**Warnings**

The warning "MLE is near a boundary or optim() failed" indicates that you should be using `ctmm.select` rather than `ctmm.fit`, because some features are not well supported by the data.

The warning "pREML failure: indefinite ML Hessian" is normal if some autocorrelation parameters cannot be well resolved.

**Note**

The default optimization method in `ctmm` v0.5.7 and above is `optimizer`'s "pNewton". Anecdotally, on these problems, `optimizer`'s `pNewton` method generally outperforms `optim`'s "Nelder–Mead", which generally outperforms `optim`'s "BFGS" and "L-BFGS-B" methods. With default arguments, "pNewton" is about half as fast as "Nelder–Mead", but is resolving about twice as much numerical precision by default.

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(\(\text{Inf}\)) and IOU(\(\tau\)) is the limit OUF(\(\text{Inf}, \tau\)). 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. Different criteria must be used to select 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 v0.3.6, 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)**


**References**


See Also
cxmm.boot, ctmn.guess, optim, summary.ctmm, variogram.fit.

Examples

# Load package and data
library(cxmm)
data(buffalo)
DATA <- buffalo$Cilla

GUESS <- ctmn.guess(DATA,interactive=FALSE)
# in general, you want to run ctmn.select instead
FIT <- ctmn.fit(DATA,GUESS)

# some human-readable information
summary(FIT)

ctmm-FAQ
ctmm FAQ

Description

Frequently asked questions for the ctmm package.

Details

General recommendations

1. Work through the vignettes vignette("variogram") and vignette("akde"). Also, see the help file for the method of interest, and its example.

2. Do not save workspaces between sessions. They will become corrupted over time. In RStudio, go to Tools: Global Options: Workspace, uncheck Restore and set Save to Never.
3. If RStudio is crashing frequently in Windows (or your display driver is crashing), try setting the rendering engine to Software under Tools : Global Options : General : Advanced : Rendering Engine.

4. Never edit or save your CSV in Microsoft Excel. The dates will be reformatted incorrectly and inconsistently.

5. If using Windows, make sure to have the suggested version of “Rtools” installed. If using MacOS, make sure to have “Xcode” installed. If using Ubuntu, make sure to have “build-essential” installed. Otherwise, you can sometimes run into problems when trying to update packages.

6. Upgrade R to the latest version and update all of your packages.

7. The development build can be installed via devtools::install_github("ctmm-initiative/ctmm").

8. Stable beta releases between the CRAN release are published here on request.

9. The ctmm user’s group is a good place to find and ask for help.

10. 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:

```bash
sudo apt install ffmpeg fftw3 libfftw3-dev libgeos-dev libgmp-dev libgsl-dev libmpfr-dev libproj-dev libnode-dev libudunits2-dev r-base-core

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 between raster, sp, move, and ctmm. Either restart R and only load the ctmm package, or run ctmm::plot instead of plot.

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`. The `compass` function is also useful for pointing north.

**Projection complains about the datatype and fails**

Namespace collision can occur between `raster` and `ctmm`. Either restart R and only load the `ctmm` package, or run `ctmm::projection` instead of `projection`.

`ctmm.guess` has no save button

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

`manipulate` panel does not popup in `ctmm.guess` or `zoom`

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

**Gear icon missing in `ctmm.guess` or `zoom`**

Recent versions of `manipulate` and/or RStudio seem to have some issues. Sometimes the gear icon does not render unless you re-run the function 2-5 times.

`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.

**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 bootstrap, as described in Fleming et al (2019).

**Usage**

```
ctmm.boot(data, CTMM, method = CTMM$method, AICc = FALSE, 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.
ctmm.boot

AICc  Run dual set of simulations to approximate AICc values via Kullback–Leibler divergence. Otherwise, only the AIC is updated.
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). If AICc=TRUE, then, in addition to an updated AICc slot, the model fit object will also contain a VAR.AICc slot quantifying the numerical variance in the AICc estimate. This variance can be decreased by decreasing argument error.

Author(s)
C. H. Fleming.

References

See Also
cxmm.fit.

Examples

# Load package and data
library(ctmm)
data(gazelle)
DATA <- gazelle[[3]]

GUESS <- ctmm.guess(DATA,interactive=FALSE)
FIT <- ctmm.select(DATA,GUESS)

# some human-readable information
summary(FIT)
# in general, you will want to set iterate=TRUE,trace=TRUE
BOOT <- ctmm.boot(DATA,FIT,iterate=FALSE,trace=FALSE)

# compare to the previous estimate
summary(BOOT)

---

difference

*Estimate the proximity of two individuals*

description

Given a pair of telemetry objects and ctmm movement models, predict their location differences at shared times and estimate their mean-square distance.

Usage

difference(data,CTMM,t=NULL,...)
distances(data,CTMM,t=NULL,level=0.95,...)
proximity(data,CTMM,GUESS=ctmm(error=TRUE),debias=TRUE,level=0.95,...)

Arguments

data A list of two telemetry objects.
CTMM A list of two ctmm movement-model objects.
t An optional vector of times over which to predict the location differences.
level Confidence level for the distance/proximity estimate.
GUESS An optional ctmm object to specify the candidate model parameters of the location differences.
debias Include inverse-χ² bias corrections.
... Options passed to ctmm.select.

details

The difference function predicts the location difference vectors, \((x_A - x_B, y_A - y_B)\), for a pair of individuals, \(\{A, B\}\), at overlapping times. The distances function further estimates the instantaneous distances between individuals. The proximity function fits an autocorrelation model to the output of difference, and then compares the mean-square distance between the individuals to what you would expect if the two individuals were moving independently.
difference outputs a telemetry object of the location differences with prediction covariances. distances outputs a data.frame of distance estimates with confidence intervals. proximity outputs a ratio estimate with confidence intervals, where values <1 indicate that the two individuals are closer on average than expected for independent movement, 1 is consistent with independent movement, and values >1 indicate that the individuals are farther from each other on average than expected for independent movement. Therefore, if the CIs contain 1, then the distance is insignificant with a p-value threshold of 1-level (two-sided) or half that for a one-sided test.

Author(s)
C. H. Fleming.

See Also
cctmm.select,predict.ctmm

Examples

#Load package
library(ctmm)

# load buffalo data
data(buffalo)

# select two buffalo that overlap in space and time
DATA <- buffalo[c(1,3)]
# plot the two buffalo
plot(DATA,col=c('red','blue'))

FITS <- list()
for(i in 1:2)
{
  GUESS <- ctmm.guess(DATA[[i]],interactive=FALSE)
  # in general, you want to use ctmm.select
  FITS[[i]] <- ctmm.fit(DATA[[i]],GUESS)
}

# calculate difference vectors
DIFF <- difference(DATA,FITS)
# plot the difference vectors with prediction-error ellipses
plot(DIFF)

# calculate the proximity statistic
# disabling location error for speed
proximity(DATA,FITS,GUESS=ctmm(error=FALSE))
distance

Calculate the square distance between two stationary distributions

Description

This function calculates various square distances measures between distributions, including the, Bhattacharyya distance, Mahalanobis distance, and Euclidean distance.

Usage

distance(object, method="Mahalanobis", level=0.95, debias=TRUE, ...)

Arguments

- **object**: A list of ctmm fit objects to compare.
- **method**: Square distance measure to return: "Bhattacharyya", "Mahalanobis", or "Euclidean".
- **level**: The confidence level desired for the output.
- **debias**: Approximate debiasing of the square distance.
- **...**: Not currently used.

Value

A table of confidence intervals on the square distance estimate. A value of 0 implies that the two distributions have the same mean location, while larger values imply that the two distributions are farther apart. The square Euclidean distance has units of square meters. The square Mahalanobis and Bhattacharyya distances are unitless.

Note

The Bhattacharyya distance (BD) is naturally of a squared form and is not further squared.

Author(s)

C. H. Fleming

See Also

cmmm.fit, overlap

Examples

```r
# 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) )
# using ctmm.fit here for speed, but you should almost always use ctmm.select
FITS <- lapply(1:2, function(i) ctmm.fit(buffalo[[i]],GUESS[[i]]) )
names(FITS) <- names(buffalo[1:2])

# Mahalanobis distance between these two buffalo
distance(FITS)

dt.plot

Functions for diagnosing sampling schedules

Description

Produces a log-scale plot of the sorted sampling intervals for inspection.

Usage

dt.plot(data,...)

Arguments

data A telemetry object.

... Additional options passed to plot.

Details

Horizontal lines are included at common sampling intervals (e.g., 1-hour) and dimmed horizontal lines are included at common subdivisions (e.g., 30-minutes).

Author(s)

C. H. Fleming.

See Also

as.telemetry.

Examples

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

# Plot the sampling intervals
dt.plot(gazelle)
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 co-variance estimates within CTMM. Strictly positive parameters, 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**

cHmm.fit, simulate.ctmm
Encounter

Encounter statistics

Description

Functions to calculate relative encounter rates and the conditional location distribution of where encounters take place (conditional on said encounters taking place), as described in Noonan et al (2021).

Usage

rates(object, debias=TRUE, level=0.95, normalize=TRUE, self=TRUE, ...)

encounter(object, include=NULL, exclude=NULL, debias=FALSE, ...)

Arguments

object A list of aligned UD objects.
debias Approximate bias corrections (encounter corrections in development).
level Confidence level for relative encounter rates.
normalize Normalize relative encounter rates by the average uncorrelated self-encounter rate.
self Fix the self-interaction rate appropriately.
include A matrix of interactions to include in the calculation (see Details below).
exclude A matrix of interactions to exclude in the calculation (see Details below).
... Additional arguments for future use.

Details

If normalize=FALSE, the relative encounter rates have units of $1/m^2$ and tend to be very small numbers for very large home-range areas. If normalize=TRUE, the relative encounter rates are normalized by the average uncorrelated self-encounter rate, which is an arbitrary value that provides a convenient scaling.

The include argument is a matrix that indicates which interactions are considered in the calculation. By default, include = 1 - diag(length(object)), which implies that all interactions are considered aside from self-interactions. Alternatively, exclude = 1 - include can be specified, and is by-default exclude = diag(length(object)), which implies that only self-encounters are excluded.

Value

rates produces an array of relative encounter rates with CIs, while encounter produces a single UD object.
Author(s)

C. H. Fleming

References


See Also

akde, overlap

Examples

```r
# 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) )
# in general, you should use ctmm.select here
FITS <- lapply(1:2, function(i) ctmm.fit(buffalo[[i]], GUESS[[i]]) )
names(FITS) <- names(buffalo[1:2])

# create aligned UDs
UDS <- akde(buffalo[1:2], FITS)

# calculate CDE
CDE <- encounter(UDS)

# plot data and encounter distribution
plot(buffalo[1:2], col=c('red','blue'), UD=CDE, col.DF='purple', col.level='purple', col.grid=NA)
```

---

export

Export ctmm data formats

Description

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

Usage

```r
as.sf(x, error=FALSE,...)
```

## method for class 'telemetry'
SpatialPoints.telemetry(object,...)
## method for class 'telemetry'
SpatialPointsDataFrame.telemetry(object,...)

## method for class 'telemetry'
SpatialPolygonsDataFrame.telemetry(object, level.UD=0.95,...)

## method for class 'UD'
SpatialPolygonsDataFrame.UD(object, convex=FALSE, level.UD=0.95, level=0.95,...)

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

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

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

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

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

### Arguments

- **x**  
  telemetry or UD object.

- **error**  
  Export telemetry location error circles/ellipses as polygons if TRUE.

- **object**  
  telemetry or UD object.

- **convex**  
  Export convex coverage areas if TRUE. By default, the highest density regions (HDRs) are exported.

- **level.UD**  
  Coverage 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.

- **DF**  
  Rasterize the probability density function "PDF", probability mass function "PMF", or cumulative distribution function "CDF".

- **folder**  
  Character name of folder for shapefile.

- **file**  
  Character name of files for shapefile.

- **filename**  
  Character name of file for raster file.

- **format**  
  Character format, if not inferred from filename extension (see writeRaster).

- **...**  
  Optional arguments passed to writeRaster, writeOGR, etc..
Details

as.sf exports ctmm objects to the sf format. Arguments to ctmm Spatial* export functions can also be used, such as level.UD and level.

Spatial* functions export ctmm objects to sp formats.

raster exports UD object point-estimates distribution functions (DF) to raster objects. DF="PDF" gives the average probability density per cell, DF="PMF" gives the total probability per cell, and DF="CDF" gives the cumulative probability.

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

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

Value

as.sf returns an sf object for the input points or polygons, with individual identity and other information retained.

SpatialPoints.telemetry returns a single spatialPoints object for the x-y locations, without individual identity and other information retained.

SpatialPointsDataFrame.telemetry returns a SpatialPointsDataFrame with the individual identities and other data recorded in the data frame retained.

SpatialPolygonsDataFrame.telemetry returns a SpatialPolygonsDataFrame that encodes the location estimate’s error circles/ellipses.

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

raster returns a raster of the point-estimate distribution function DF, given a UD object.

Author(s)


See Also

akde, as.telemetry, occurrence.

extent

<table>
<thead>
<tr>
<th>extent</th>
<th>Extent</th>
</tr>
</thead>
</table>

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**

```r
## 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, complete=FALSE, ...)

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

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

## S4 method for signature 'data.frame'
extent(x, level=1, ...)

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

**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**
  - Coverage level of the UD area. I.e., the 50% core home range would be given by level.UD=0.50.

- **complete**
  - Also calculate longitude-latitude extent of UD objects.

- **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 UD.s on the same scale.

**Author(s)**

C. H. Fleming

**See Also**

- plot.telemetry, plot.variogram.
Description

Functions for concisely representing dimensionful quantities and uncertain quantities.

Usage

dimfig(data,dimension,thresh=1,...)

sigfig(est,VAR=NULL,SD=NULL,level=0.95,digits=2,...)

Arguments

data A numerical vector of dimensionful quantities represented in SI units.
dimension One of "length", "area", "time", "frequency", "speed", "diffusion", or "mass".
thresh Threshold quantity for switching between units. E.g., 100 cm is represented as 1 m only if thresh>=1.
est Can be either confidence-interval estimates with rows (lower-limit,point-estimate,upper-limit) or point estimates (with VAR or SD also specified).
VAR Variance in the sampling distribution of x.
SD Standard deviation in the sampling distribution of x.
level Confidence level for designating the numerical precision of the significant digits.
digits Number of significant digits to retain.
... Not currently used.

Details

dimfig chooses the set of units that provides the most concise representation for data, and sigfig concisely represents statistical estimates with a fixed number of significant digits.

Value

dimfig returns a list with slots for the converted data and the name of the most concise units.
sigfig returns a character string that is formatted with the specified number of significant digits.

Author(s)

C. H. Fleming.

See Also

%#%
Examples

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

GUESS <- ctmm.guess(DATA, interactive=FALSE)
# in general, you want to run ctmm.select instead
FIT <- ctmm.fit(DATA, GUESS)

# raw summary (SI units)
summary(FIT, units=FALSE)

# default summary (concise units)
summary(FIT, units=TRUE)

# text-formatted summary
sigfig( summary(FIT)$CI )
```

---

**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


See Also

as.telemetry, plot.telemetry, buffalo, coati, jaguar, pelican, turtle, wolf.
Examples

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

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

---

**homerange**  
*Calculate a range distribution estimate*

**Description**

Estimates the range distributions and suitability from telemetry data and a continuous-time movement model.

**Usage**

```r
homerange(data, CTMM, method="AKDE", ...)
suitability(CTMM, R=list(), grid=NULL, ...)
agde(CTMM, R=list(), variable="utilization", error=0.001, res=100, grid=NULL, ...)
```

**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. Can be "AKDE" or "AGDE".

- **...**  
  Arguments passed to the method call or `bandwidth`.

- **R**  
  A named list of raster covariates if `CTMM` contains an RSF model.

- **grid**  
  Grid specification via `raster`, `UD`, or list of arguments (See `akde` for details).

- **variable**  
  Not yet supported.

- **error**  
  Target probability error.

- **res**  
  Number of grid points along each axis, relative to the location covariance.

**Details**

- `homerange` is a wrapper function that calls either `akde` or `agde`. Please consult `akde` for further details on method="AKDE".

- `suitability` calculates a suitability raster from an `rsf.fit` object.

- `agde` calculates autocorrelated Gaussian and RSF home-range areas.
Value

homerange and agde return a UD object. suitability returns a raster object.

Author(s)

C. H. Fleming.

See Also

akde, raster, UD-method

---

jaguar  
Jaguar data from the Jaguar movement database.

Description

x-y projected GPS data on 4 jaguar. Please cite Morato et al (2018) when publishing with these data.

Usage

data("jaguar")

Format

A list of 4 telemetry objects.

References


See Also

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

Examples

# Load package and data  
library(ctmm)  
data("jaguar")

# Plot all jaguar locations  
plot(jaguar, col=rainbow(length(jaguar)))
Log transformation of parameter estimates and their uncertainties

Description

Methods for log transforming individual parameter estimates and their uncertainty estimates for use in meta-analytic regression, and then back-transforming mean-log parameter estimates back to mean parameter estimates.

Usage

Log(x, debias=TRUE, ...)

Exp(est, VAR.est=0, VAR=0, VAR.VAR=0, variable="area", debias=TRUE, level=0.95, units=TRUE, ...)

Arguments

- **x**: A list of UD objects, UD summary objects, or speed objects.
- **debias**: Apply log χ² and log χ bias corrections if TRUE.
- **...**: Further arguments passed.
- **est**: Point estimate of the mean log-parameter.
- **VAR.est**: Uncertainty in the mean log-parameter estimate (square standard error).
- **VAR**: Variance in the log-parameters.
- **VAR.VAR**: Uncertainty in the log-parameter variance estimate (square standard error).
- **variable**: Variable being back-transformed. Can be "area" or "speed".
- **level**: Confidence level for parameter estimates.
- **units**: Convert result to natural units.

Value

Log returns a list with two slots, log and VAR.log, corresponding to the point estimates and variance estimates of the logged variables.

Exp returns a confidence intervals for the back-transformed mean parameter estimate.

Author(s)

C. H. Fleming.

See Also

meta, mean.
mean.ctmm

Examples

# load package and data
library(ctmm)
data(buffalo)

# fit movement models
FITS <- AKDES <- list()
for(i in 1:length(buffalo))
{
  GUESS <- ctmm.guess(buffalo[[i]],interactive=FALSE)
  # use ctmm.select unless you are certain that the selected model is OUF
  FITS[[i]] <- ctmm.fit(buffalo[[i]],GUESS)
}

# calculate AKDES on a consistent grid
AKDES <- akde(buffalo,FITS)

# extract 95% areas
AREAS <- lapply(AKDES,summary)

# log transform for further meta-analysis
LOG <- Log(AREAS)

LOG

mean.ctmm

Average movement models and autocorrelated kernel density estimates

Description

These functions calculate population averages of continuous-time movement models and utilization distributions.

Usage

## S3 method for class 'ctmm'
mean(x,weights=NULL,sample=TRUE,debias=TRUE,IC="AICc",...)

## S3 method for class 'UD'
mean(x,weights=NULL,sample=TRUE,...)

Arguments

x

A list of ctmm objects calculated in the same projection or UD objects calculated on the compatible grids.

weights

A vector of numeric weights with the same length as x, specifying the relative frequency of each distribution in x.
sample: x represents a sample of a larger population if TRUE, or the entire statistical population if FALSE.

debias: Include log $-\chi^2$ and REML bias corrections.

IC: Model selection criterion for the anisotropy of the distribution of mean locations and covariance matrices.

... Additional arguments for future use.

Details

When applied to a list of ctmm objects, mean calculates an average movement model with population variability estimates. The population model is taken to be multivariate normal and log-normal. The population mean location represents an arithmetic mean, while the population mean home-range areas, RMS speeds, and diffusion rates represent geometric means. Location-error estimates are not correctly averaged yet.

When applied to a list of UD objects, mean calculates a weighted average of autocorrelated kernel density home-range estimates from akde. The point estimates are correct, but the confidence-interval calculation is not yet complete.

By default, uniform weights are used (weights=rep(1,length(x))). This can be sensible for averaging over individuals. For averaging over periods of time, users should consider weighting by the proportion of time spent in each distribution. For example, if an animal spends 4 months in its winter range, $x[[1]]$, and 7 months in its summer range, $x[[2]]$, then the annual range (sans migration corridor) would be calculated with weights=c(4,7).

All UDs need to be calculated on the same grid (see overlap for an example).

Value

When applied to a list of ctmm objects, mean returns a ctmm object with additional population variability parameter estimates.

When applied to a list of UD objects, mean 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.

Author(s)

C. H. Fleming

See Also

akde, ctmm.select
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

```r
## 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


See Also

plot.variogram, variogram.
Examples

```r
# 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)
```

---

**meta**

_Meta-analysis of movement-model parameters_

**Description**

These functions estimate population-level mean parameters from individual movement models and related estimates, including AKDE home-range areas, while taking into account estimation uncertainty.

**Usage**

```r
meta(x, variable="area", level=0.95, level.UD=0.95, method="MLE", IC="AICc", boot=FALSE, error=0.01, debias=TRUE, verbose=FALSE, units=TRUE, plot=TRUE, sort=FALSE, mean=TRUE, col="black", ...)
```

**Arguments**

- `x` A named list of `ctmm` movement-model objects, UD objects, or UD summary output, constituting a sampled population, or a named list of such lists, with each constituting a sampled population.
- `variable` Variable of interest. Can be "area", "diffusion", "speed", "tau position", or "tau velocity".
- `level` Confidence level for parameter estimates.
- `level.UD` Coverage level for home-range estimates. E.g., 50% core home range.
- `method` Statistical estimator used—either maximum likelihood estimation based ("MLE") or approximate ‘best linear unbiased estimator’ ("BLUE")—for comparison purposes.
- `IC` Information criterion to determine whether or not population variation can be estimated. Can be "AICc", "AIC", or "BIC".
boot Perform a parametric bootstrap for confidence intervals and first-order bias correction if debias=TRUE.

error Relative error tolerance for parametric bootstrap.

debias Apply Bessel’s inverse-Gaussian correction and various other bias corrections if method="MLE", REML if method="BLUE", and an additional first-order correction if boot=TRUE.

verbose Return a list of both population and meta-population analyses if TRUE and x is a list of population lists.

units Convert result to natural units.

plot Generate a meta-analysis forest plot.

sort Sort individuals by their point estimates in forest plot.

mean Include population mean estimate in forest plot.

col Color(s) for individual labels and error bars.

... Further arguments passed to plot.

Details

meta employs a custom $\chi^2 - IG$ hierarchical model to calculate debiased population mean estimates of positive scale parameters, including home-range areas, diffusion rates, mean speeds, and autocorrelation timescales. Population coefficient of variation (CoV) estimates are also provided. More details are provided in Fleming et al (2022).

Value

If x constitutes a sampled population, then meta returns a table with rows corresponding to the population mean and coefficient of variation.

If x constitutes a list of sampled populations, then meta returns confidence intervals on the population mean variable ratios.

Note

The AICc formula is approximated via the Gaussian relation.

Confidence intervals depicted in the forest plot are $\chi^2$ and may differ from the output of summary() in the case of mean speed and timescale parameters with small effective sample sizes.

As mean ratio estimates are debiased, reciprocal estimates can differ slightly.

Author(s)

C. H. Fleming.

References

See Also

akde, cluster, ctmm.fit.

Examples

# load package and data
library(ctmm)
data(buffalo)

# fit movement models
FITS <- AKDES <- list()
for(i in 1:length(buffalo))
{
  GUESS <- ctmm.guess(buffalo[[i]],interactive=FALSE)
  # use ctmm.select unless you are certain that the selected model is OUF
  FITS[[i]] <- ctmm.fit(buffalo[[i]],GUESS)
}

# calculate AKDES on a consistent grid
AKDES <- akde(buffalo,FITS)

# color to be spatially distinct
COL <- color(AKDES,by="Var individual")

# plot AKDEs
plot(AKDES,col.DF=COL,col.level=COL,col.grid=NA,level=NA)

# meta-analysis of buffalo home-range areas
meta(AKDES,col=c(COL,'black'),sort=TRUE)

npr

Calculate a non-parametric regression surface

Description

This function estimates the mean value of an annotated covariate as a function of location, using non-parametric regression.

Usage

npr(data,UD,variable="speed",normalize=FALSE,debias=TRUE,error=0.001,...)

Arguments

data 2D timeseries telemetry data represented as a telemetry object or list of objects.

UD A UD object from the output of akde.
variable Variable for mean estimation. Can be a column of data.
normalize Consider variable as providing a weighted probability distribution.
debias Correct for oversmoothing if normalize=TRUE.
error Target probability error.
... Arguments passed to akde.

Value
Returns a UD object.

Author(s)
C. H. Fleming.

See Also
akde, occurrence

Examples

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

# calculate fit guess object
GUESS <- ctmm.guess(DATA,interactive=FALSE)
# in general, you should be running ctmm.select here instead of ctmm.fit
FIT <- ctmm.fit(DATA,GUESS)

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

# compute revisitation distribution
RD <- revisitation(DATA,UD)

# Plot data with revisitation distribution
plot(DATA,RD)

 occurrence

Calculate a Kriged occurrence distribution estimate

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

occurrence(data, CTMM, R=list(), SP=NULL, SP.in=TRUE, H=0, variable="utilization", res.time=10, res.space=10, grid=NULL, cor.min=0.05, dt.max=NULL, buffer=TRUE,...)

Arguments

data A telemetry object or list of telemetry objects.

CTMM A ctmm movement model, as from the output of ctmm.select, or a list of ctmm objects.

R A named list of raster covariates if CTMM contains an RSF model.

SP SpatialPolygonsDataFrame object for enforcing hard boundaries.

SP.in Locations are assumed to be inside the SP polygons if SP.in=TRUE and outside of SP if SP.in=FALSE.

H Optional additional bandwidth matrix for future use.

variable Either "utilization" or "revisitation". Only utilization is accurately estimated.

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.

gird Optional grid specification via raster, UD, or list of arguments (See akde for details).

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.

buffer Buffer the observation period, according to the minimum gap specified by cor.min and dt.max, to include more probable locations if possible.

... Not 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 for interpolation.

If buffer=TRUE, then the data are also extrapolated according to the minimum of the two rules (cor.min and dt.max) which is limited to cases where persistence of motion is modeled.

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


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 occurrence distribution
UD <- occurrence(Cilla,FIT)

# Plot occurrence UD
plot(UD,col.level=NA)
Minimize a function

**Description**

This function serves as a wrapper around `optimize`, `optim`, and `ctmm`'s partial-Newton optimization routine, with standardized arguments and return values. It finds the optimal parameters that minimize a function, whether it be a cost, loss, risk, or negative log-likelihood function.

**Usage**

```r
tmp <- optimizer(par, fn, ..., method = "pNewton", lower = -Inf, upper = Inf, period = FALSE, reset = identity, control = list())
```

**Arguments**

- **par**: Initial parameter guess.
- **fn**: Function to be minimized with first argument `par` and optional argument `zero` (see 'Details' below).
- **...**: Optional arguments fed to `fn`.
- **method**: Optimization algorithm (see 'Details' below).
- **lower**: Lower bound for parameters.
- **upper**: Upper bound for parameters.
- **period**: Period of circular parameters if not FALSE.
- **reset**: Optional function to re-center parameters, if symmetry permits, to prevent numerical underflow.
- **control**: Argument list for the optimization routine (see 'Details' below).

**Details**

Only `method = 'pNewton'` will work in both one dimension and multiple dimensions. Any other `method` argument will be ignored in one dimension, in favor of `optimize` with a backup evaluation of `nlm` (under a log-link) for cases where `optimize` is known to fail. In multiple dimensions, methods other than `pNewton` include those detailed in `optim`.

`method = 'pNewton'` is `ctmm`'s partial-Newton optimizer, which is a quasi-Newton method that is more accurate than BFGS-based methods when the gradient of `fn` must be calculated numerically. In short, while BFGS-based methods provide a single rank-1 update to the Hessian matrix per iteration, the partial-Newton algorithm provides `length(par) + 1` rank-1 updates to the Hessian matrix per iteration, at the same computational cost. Furthermore, `length(par)` of those updates have better numerical precision than the BFGS update, meaning that they can be used at smaller step sizes to obtain better numerical precision. The `pNewton` optimizer also supports several features not found in other R optimizers: the `zero` argument, the `period` argument, and parallelization.

The `zero` argument is an optional argument in `fn` supported by `method = 'pNewton'`. Briefly, if you rewrite a negative log-likelihood of the form $f_n = \sum_{i=1}^{n} f n_i$ as $f_n = \sum_{i=1}^{n} (f n_i - zero/n) + zero$, the function.

```r
tmp <- optimizer(par, fn, ..., method = "pNewton", lower = -Inf, upper = Inf, period = FALSE, reset = identity, control = list())
```
where zero is the current estimate of the minimum value of fn, then the sum becomes approximately "zeroed" and so the variance in numerical errors caused by the difference in magnitude between fn and fn_i is mitigated. In practice, without the zero argument, log-likelihood functions grow in magnitude with increasing data and then require increasing numerical precision to resolve the same differences in log-likelihood. But absolute differences in log-likelihoods (on the order of 1) are always important, even though most optimization routines more naturally consider relative differences as being important.

The period argument informs method='pNewton' if parameters is circular, such as with angles, and what their periods are.

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

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

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

parscale=pmin(abs(par),abs(par-lower),abs(upper-par)) The natural scale of the parameters such that variations in par on the order of parscale produce variations in fn on the order of one.

trace=FALSE Return step-by-step progress on optimization.

cores=1 Perform cores evaluations of fn in parallel, if running in UNIX. cores<=0 will use all available cores, save abs(cores). This feature is only supported by method='pNewton' and is only useful if fn is slow to evaluate, length(par)>1, and the total number of parallel evaluations required does not trigger fork-bomb detection by the OS.

Value

Returns a list with components par for the optimal parameters, value for the minimum value of fn, and possibly other components depending on the optimization routine employed.

Note

method='pNewton' is very stringent about achieving its precision target and assumes that fn has small enough numerical errors (permitting the use of argument zero) to achieve that precision target. If the numerical errors in fn are too large, then the optimizer can fail to converge. ctmm.fit standardizes its input data before optimization, and back-transforms afterwards, as one method to minimize numerical errors in fn.

Author(s)

C. H. Fleming.

See Also

optim, optimize.nlm
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, plot=TRUE, by='d', ...)  
## S3 method for class 'outlie'
plot(x, level=0.95, units=TRUE, axes=c('d', 'v'), xlim=NULL, ylim=NULL, ...)

Arguments

data telemetry object.
plot Output a plot highlighting high speeds (blue) and distant locations (red).
by Color and size side-effect plot points by 'd', 'v', 'dz', 'vz', for distance from center, minimum speed, vertical distance from center, and minimum vertical speed.
... Arguments passed to plot.
x outlie object to plot.
level Confidence level for error bars.
units Convert axes to natural units.
axes $x$-$y$ axes to plot. Can be any of 'd', 'v', 'dz', 'vz', for time, distance from center, minimum speed, vertical distance from center, and minimum vertical speed.
xlim $x$-axis plotting range in SI units.
ylim $y$-axis plotting range in SI units.

Details

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

When plotting the outlie object itself, ‘core deviation’ denotes distances from the median longitude & latitude, while ‘minimum speed’ denotes the minimum speed required to explain the location estimate’s displacement as straight-line motion. 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.

The output outlie object contains the above noted speed and distance estimates in a data.frame, with rows corresponding to those of the input telemetry object.
Value

Returns an outlie object, which is a data.frame of distance and speed information. 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).

In ctmm v0.6.1 the UERE argument was deprecated. For uncalibrated data, the initial estimates used by outlie are now generated on import and stated by summary(uere(data)). These values not be reasonable for arbitrary datasets.

Author(s)

C. H. Fleming.

References


See Also

as.telemetry.

Examples

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

# look for outliers in a turtle
OUT <- outlie(turtle[[3]])

# look at the distribution of estimates
plot(OUT)
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, method="Bhattacharyya", level=0.95, debias=TRUE,...)

Arguments
- object: A list of ctmm fit or aligned UD objects to compare.
- method: Can be "Bhattacharyya" or "Encounter" (see Details below).
- level: The confidence level desired for the output.
- debias: Approximate debiasing of the overlap.
- ...: Not currently used.

Details
The default method="Bhattacharyya" estimates the standard overlap measure $\int \int \sqrt{p(x, y) q(x, y)} \, dx \, dy$ between the distributions $p(x, y)$ and $q(x, y)$, while method="encounter" estimates the non-standard measure $\frac{\int \int p(x, y) q(x, y) \, dx \, dy}{\sqrt{\int \int p(x, y)^2 \, dx \, dy \int \int q(x, y)^2 \, dx \, dy}}$, which has a numerator proportional to the uncorrelated encounter rate. Both measures lie between 0 and 1, where 0 indicates no shared support and 1 indicates identical distributions.

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.

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


See Also

akde, ctmm.fit, distance, encounter

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) )
# using ctmm.fit here for speed, but you should almost always use ctmm.select
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)

pelican  Brown Pelican GPS and ARGOS data.

Description

GPS and ARGOS data on a single brown pelican (Pelecanus occidentalis). Please contact Autumn-Lynn Harrison (HarrisonAL@si.edu) if you want to publish with these data.

Funding for Brown Pelican tracking was provided by the Friends of the National Zoo Conservation Research Grant and ConocoPhillips Global Signature Program. Field support provided by D. Brinker.

Usage

data("pelican")

Format

A list of 2 telemetry objects.
periodogram

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.

See Also

as.telemetry, plot.telemetry, buffalo, coati, gazelle, jaguar, turtle, wolf.
periodogram

```
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 \(1/(2*\text{range(data$t$)+dt})\) 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, \(\text{LSP}\).

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


Examples

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

#Extract movement data for a single animal
DATA <- 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(DATA, 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**

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

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

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

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

## S4 method for signature 'UD'
```r
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.
col.bg  Background color
cex  Relative size of plotting symbols. Only used when error=FALSE, because error=TRUE uses the location-error radius instead of cex.
col  Color option for telemetry data. Can be an array or list of arrays.
lwd  Line widths of telemetry points.
pch  Plotting symbol. Can be an array or list of arrays.
type  How plot points are connected. Can be an array.
error  Plot error circles/ellipses if present in the data. error=2 will fill in the circles and error=3 will plot densities instead. error=FALSE will disable this feature.
transparency.error  Transparency scaling for erroneous locations when error=1:2. trans=0 disables transparancy. Should be no greater than 1.
velocity  Plot velocity vectors if present in the data.
DF  Plot the maximum likelihood probability density function "PDF" or cumulative distribution function "CDF".
col.DF  Color option for the density function. Can be an array.
col.grid  Color option for the maximum likelihood akde bandwidth grid.
labels  Labels for UD contours. Can be an array or list of arrays.
convex  Plot convex coverage-area contours if TRUE. By default, the highest density region (HDR) contours are plotted.
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.
level.UD  Coverage 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.
col.level  Color option for home-range contours. Can be an array.
lwd.level  Line widths of UD contours.
SP  SpatialPolygonsDataFrame object for plotting a shapefile base layer.
border.SP  Color option for shapefile polygon boundaries.
col.SP  Color option for shapefile polygon regions.
R  Background raster, such as habitat suitability.
col.R  Color option for background raster.
legend  Plot a color legend for background raster.
fraction

Quantile fraction of the data, Gaussian ctmm, or UD range to plot, whichever is larger.

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).

units

Convert axes to natural units.

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.

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”. Grid lines can be disabled with the argument col.grid=NA.

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 variogram 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 black 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.
Projection

Description

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

Usage

```r
# 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, ...)

compass(loc=NULL, cex=3, ...)

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 or text.

loc Optional two-dimensional coordinates (in meters) at which to draw a north-facing compass needle.

cex Relative size of compass.

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. compass(c(x,y)) plots a north-pointing compass needle at the coordinates (x,y).

Note

Plotting UTF-8 chracters in a PDF, like the compass needle, requires specifying a compatible font family. For example:

```r
library(ctmm)
data(buffalo)
cairo_pdf(file="buffalo.pdf", family="DejaVu Sans")
plot(buffalo[[1]])
compass()
dev.off()
```

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)
compass()

# Apply a 2-point projection safer for elongated distributions
projection(buffalo) <- median(buffalo,k=2)
# This is the default projection for ctmm
plot(buffalo)
compass()

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. Finally, mag calculates residual magnitudes, which is useful for comparing against
potential covariates.

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"), trace=TRUE)

mag(x, ...)

## S3 method for class 'telemetry'
mag(x, axes=c("x","y"), ...)

## S3 method for class 'telemetry'
mag(x, axes=c("x","y"), ...)

## S3 method for class 'telemetry'
mag(x, axes=c("x","y"), ...)

## S3 method for class 'telemetry'
mag(x, axes=c("x","y"), ...)

residuals.ctmm
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.
- **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 for which to calculate residual correlogram or magnitudes.
- **trace**: Display a progress bar if fast=FALSE.
- **x**: telemetry object from the output of residuals.

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


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%, 95%, and 99.9% quantiles
plot(RES,col.DF=NA,level.UD=c(.50,.95,0.999))

# 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)

---

revisitation

Calculate an revisitation distribution estimate

Description

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

Usage

revisitation(data,UD,debias=TRUE,error=0.001,...)
revisitation

Arguments

- **data**: 2D timeseries telemetry data represented as a telemetry object or list of objects.
- **UD**: A UD object from the output of *akde*.
- **debias**: Correct for oversmoothing.
- **error**: Target probability error.
- **...**: Arguments passed to *akde*.

Value

Returns a UD object.

Author(s)

C. H. Fleming.

See Also

- *akde*, *occurrence*

Examples

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

# calculate fit guess object
GUESS <- ctmm.guess(DATA, interactive=FALSE)
# in general, you should be running ctmm.select here instead of ctmm.fit
FIT <- ctmm.fit(DATA, GUESS)

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

# compute revisitation distribution
RD <- revisitation(DATA, UD)

# Plot data with revisitation distribution
plot(DATA, RD)
```
rsf.fit  
Fit integrated resource selection functions (iRSFs) with autocorrelation-adjusted weighted likelihood [IN DEVELOPMENT]

Description
This function fits integrated resource selection functions with autocorrelation-adjusted weights on the RSF likelihood function, importance sampling, and iterative numerical convergence. This function is in development and is subject to change.

Usage
rsf.fit(data, UD, beta=NULL, R=list(), formula=NULL, integrated=TRUE, reference="auto", level.UD=0.99, isotropic=TRUE, debias=TRUE, smooth=TRUE, standardize=TRUE, integrator="MonteCarlo", error=0.01, max.mem="1 Gb", interpolate=TRUE, trace=TRUE,...)

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A telemetry object.</td>
</tr>
<tr>
<td>UD</td>
<td>A UD object generated by akde from the same telemetry object as data. If weights were optimized in akde, then they will be adopted by rsf.fit.</td>
</tr>
<tr>
<td>beta</td>
<td>Initial guesstimates for regression coefficients (not currently important).</td>
</tr>
<tr>
<td>R</td>
<td>A named list of rasters or time-varying raster stacks [NOT TESTED] to fit Poisson regression coefficients to (under a log link).</td>
</tr>
<tr>
<td>formula</td>
<td>Formula object for log(λ) referencing the elements of R and columns of data (see Details below). If not specified, a linear term will be included for every element of R.</td>
</tr>
<tr>
<td>integrated</td>
<td>Fit an integrated RSF model with simultaneously estimated spatial constraints. integrated=FALSE is for comparison purposes only.</td>
</tr>
<tr>
<td>reference</td>
<td>When expanding categorical predictors into indicator variables, reference=&quot;auto&quot; will choose the most common predictor to be the reference category. Otherwise, the reference category can be specified by this argument.</td>
</tr>
<tr>
<td>level.UD</td>
<td>Coverage probability of UD to sample uniformly from if integrated=FALSE. Can also be a pre-defined spatial polygon object.</td>
</tr>
<tr>
<td>isotropic</td>
<td>Force spatial constraint parameters to be symmetric (only isotropic=TRUE is currently supported).</td>
</tr>
<tr>
<td>debias</td>
<td>Apply a post-hoc bias correction to the spatial constraint parameters, and apply bias corrections to the numerical log-likelihood estimates.</td>
</tr>
<tr>
<td>smooth</td>
<td>Apply location-error smoothing to the tracking data before regression.</td>
</tr>
<tr>
<td>standardize</td>
<td>For numerical stability, predictors are internally standardized, if rescale=TRUE and no formula is specified. (The final outputs are not standardized.) Otherwise, users are responsible for standardizing their predictors.</td>
</tr>
<tr>
<td>integrator</td>
<td>Numerical integrator used for likelihood evaluation. Can be &quot;MonteCarlo&quot; or &quot;Riemann&quot; (IN TESTING).</td>
</tr>
</tbody>
</table>
**error** Relative numerical error threshold for the parameter estimates and log-likelihood.

**max.mem** Maximum amount of memory to allocate for availability sampling.

**interpolate** Whether or not to interpolate raster values during extraction.

**trace** Report progress on convergence (see Details).

... Arguments passed to optimizer.

**Details**

For autocorrelated tracking data, the relative weights of the log-likelihood used here are taken from the output of `akde`, which are optimized for non-parametric density estimation (if `weights=TRUE`), and so are approximate here. The absolute weight of the data is taken to be the effective sample size of the integrated spatial parameters, when estimated separately.

Integrated resource selection functions simultaneously estimate the spatially constraining (availability) parameters with the resource selection parameters, rather than first estimating the availability parameters (usually via MCP) and then holding those parameters fixed—as known values—when estimating the resource selection parameters. The “integrated” analysis reduces estimation bias, exposes correlations in the resource and availability estimate uncertainties, and propagates the availability estimate uncertainties into the final outputs.

Instead of specifying a number of “available” points to sample and having an unknown amount of numerical error to contend with, `rsf.fit` specifies an estimation target `error` and the number of “available” points is increased until this target is met. Moreover, the output log-likelihood is that of the continuous Poisson point process, which does not depend on the number of “available” points that were sampled, though the numerical variance estimate is recorded in the `VAR.loglike` slot of the fit object.

When `trace=TRUE`, a number of convergence estimates are reported, including the standard deviation of the numerical error of the log-likelihood, \( \text{SD}[\log(\ell)] \), the most recent log-likelihood update, \( d\log(\ell) \), and the most recent (relative) parameter estimate updates \( d\hat{\beta}/\text{SD}[\hat{\beta}] \).

The `formula` object determines \( \log(\lambda) \) and can reference static rasters in R, time-dependent raster stacks in R [NOT TESTED], and time-dependent effect modifiers in the columns of `data`, such as provided by `annotate`. Any offset terms are applied under a log transformation (or multiplicatively to \( \lambda \)), and can be used to enforce hard boundaries, where `offset(raster)=TRUE` denotes accessible points and `offset(raster)=FALSE` denotes inaccessible points [NOT TESTED]. Intercept terms are ignored, as they generally do not make sense for individual Poisson point process models. This includes terms only involving the columns of `data`, as they lack spatial dependence.

Categorical raster variables are expanded into indicator variables, according to the `reference` category argument. Upon import via `raster`, categorical variables may need to be assigned with `as.factor`, or else they may be interpreted as numerical variables.

**Note**

It is much faster to calculate all predictors ahead of time and specifying them in the R list than to reference them in the `formula` argument, which will calculate them as needed, saving memory.

AIC and BIC values for `integrated=FALSE` models do not include any penalty for the estimated location and shape of the available area, and so their AIC and BIC values are expected to be **worse** than reported.
**Author(s)**

C. H. Fleming and B. Reineking

**See Also**

*ctmm.fit, optimizer, summary.ctmm.*

---

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

**Description**

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

**Usage**

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

**Arguments**

- `object` telemetry object or list of such objects.
- `fraction` Initial split, as fraction of total time period.
- `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.
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,VMM=NULL,t=NULL,dt=NULL,res=1,complete=FALSE,...)

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

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

See Also

plot.telemetry

Examples

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

  # Extract wolf Luna
  DATA <- wolf$Luna

  # Select resident data
  SUB <- lasso(DATA)

  # You can now work with the resident and dispersive data separately
  names(SUB)
}
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.
- **CTMM**: A ctmm movement model in the same format as the output of ctmm.fit or variogram.fit.
- **VMM**: An optional vertical ctmm movement model for 3D predictions and simulations.
- **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.
- **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.
speed

**Author(s)**

C. H. Fleming.

**References**


**See Also**

cxmm.fit

**Examples**

```r
#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 mean speed of the Gaussian movement process. All methods are described in Noonan & Fleming et al (2019).
Usage

speed(object,...)

## S3 method for class 'ctmm'
speed(object,data=NULL,t=NULL,level=0.95,robust=FALSE,units=TRUE,prior=TRUE,fast=TRUE,
cor.min=0.5,dt.max=NULL,error=0.01,cores=1,trace=TRUE,...)

## S3 method for class 'telemetry'
speed(object,CTMM,t=NULL,level=0.95,robust=FALSE,units=TRUE,prior=TRUE,fast=TRUE,
cor.min=0.5,dt.max=NULL,error=0.01,cores=1,trace=TRUE,...)

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,trace=TRUE,...)

## 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,trace=TRUE,...)

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, or range of times to estimate
mean speed over.
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 parameter uncertainty.
fast Whether or not to invoke the central-limit theorem when propagating parameter
uncertainty (see emulate).
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).
trace Display a progress bar.
... 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 occasionally include impersistent motion and its mean will be infinite. In these cases robust=TRUE can be used to report the sampling distribution's median rather than its mean. The time average of speed, in either case, is still the mean average of times 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 mean speed estimated by speed is applicable only during the sampling periods. If an individual is diurnal/nocturnal and only tracked during the day/night, then the output of speed will only be the mean speed during the day/night. For instance, if an individual is tracked the 12 hours per day during which it is active, and speed reports a mean speed of 10 kilometers per day during those periods, then the average distance traveled per day is only 5 kilometers (from 10 kilometers / day * 12 hours). An average of 10 kilometers would only result if the individual were similarly active for 24 hours a day.

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.

References


See Also

emulate, simulate

Examples

# Load package and data
library(ctmm)
data(buffalo)
DATA <- buffalo$Gabs
GUESS <- ctmm.guess(DATA,interactive=FALSE)
# in general, you should use ctmm.select instead
FIT <- ctmm.fit(DATA,GUESS)

# stationary Gaussian estimate
speed(FIT)

# conditional estimate
# you will likely want trace=TRUE
speed(FIT,DATA,trace=FALSE)

---

### 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**

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

**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", "LOOCV", "HSCV", 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, point, 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. 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, "OUF" denotes the continuous-velocity Ornstein-Uhlenbeck-F model, "OUf" denotes the OUF model where the two autocorrelation timescales cannot be statistically distinguished.

Note
Confidence intervals on the autocorrelation timescales assume they are sufficiently greater than zero and less than infinity.
IC="LOOCV" can only be attempted if also specified during ctmm.select, as this argument requires additional calculations.
Prior to ctmm v0.6.2, timescale confidence intervals were constructed from normal and inverse-normal sampling distributions, whereas v0.6.2 onward uses gamma and inverse-gamma sampling distributions.
In ctmm v0.5.1 onward the MSPE is 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
DATA <- buffalo$Cilla
# fit model
GUESS <- ctmm.guess(DATA,interactive=FALSE)
FIT <- ctmm.fit(DATA,GUESS)

# Tell us something interpretable
summary(FIT)

summary.UD  

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

Arguments

- **object**  
  An akde autocorrelated kernel-density estimate from the output of akde.

- **convex**  
  Report convex coverage areas if TRUE. By default, the highest density regions (HDRs) are reported.

- **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 list is returned with the effective sample sizes of various parameter estimates (DOF) and a parameter estimate table CI, with low, point, and high estimates for the following possible 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.

*Summarize a range distribution*
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 \( \text{DOF}_H \) and used this to estimate area uncertainty under a chi-square approximation. Starting in v0.3.2, this method was improved to use \( \text{DOF}_\text{area} \) in the Gaussian reference function approximation.

Author(s)
C. H. Fleming.

References

See Also
akde.

Examples

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

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

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

# Estimate and summarize the AKDE
UD <- akde(DATA, 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, jaguar, pelican, wolf.

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

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

uere Estimate RMS UERE from calibration data

Description
Functions for estimating and assigning the root-mean-square 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 RMS 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 specify the device’s RMS UERE necessary to transform the HDOP values into absolute errors. `uere.fit()` allows users to estimate the RMS UERE from calibration data, where the device was left fixed over a period of time. The calibration RMS UERE can then be applied to tracking data with the `uere()<-` assignment method. Otherwise, when `error=TRUE` in `ctmm, ctmm.fit` will estimate the RMS UERE simultaneously with the movement model, which is less reliable than using calibration data.

`summary()` applied to single `UERE` object will return RMS UERE parameter estimates and confidence intervals in meters, 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 RMS UERE estimate.

Note

The GPS device should be fixed during calibration.

Author(s)

C. H. Fleming

References


See Also

`as.telemetry, residuals.telemetry`.

Examples

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

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

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

# assign RMS UERE to entire dataset
uere(turtle) <- UERE
```
# calculate residuals of calibration data
RES <- lapply(turtle[1:2], residuals)

# scatter plot of residuals with 50%, 95%, and 99.9% coverage areas
plot(RES, col.DF=NA, level.UD=c(0.50, 0.95, 0.999))

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

# 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 \(x\).

The default non-SI units include the mean solar 'day', mean synodic 'month' and mean tropical 'year'. These defaults can be changed to conventional calendar units via `options(time.units='calendar')`.

**Value**

Returns a numeric in SI units or units specified by character label \(x\).

**Author(s)**

C. H. Fleming.
**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 and an iterative algorithm to adjust for calibrated errors. 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**

```r
variogram(data, dt=NULL, fast=TRUE, res=1, CI="Markov", error=FALSE, axes=c("x","y"), precision=1/8, trace=TRUE)
```

**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.

**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"

# the same conversion in one step
6 %% "mph"
```
**res**
Increase the discretization resolution for irregularly sampled data with res>1. Decreases bias at the cost of smoothness.

**CI**
Argument for confidence-interval estimation. Can be “IID” to consider all unique lags as independent, “Markov” to consider only non-overlapping lags as independent, or “Gauss” for an exact calculation (see Details below).

**error**
Adjust for the effect of calibrated errors.

**axes**
Array of axes to calculate an average (isotropic) variogram for.

**precision**
Fraction of machine precision to target when adjusting for telemetry error (fast=FALSE with calibrated errors). precision=1/8 returns about 2 decimal digits of precision.

**trace**
Display a progress bar if fast=FALSE.

### 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 conventional 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, though this is a crude approximation and is overly conservative at large lags. CI=”Gauss” implements exact confidence intervals under the assumption of a stationary Gaussian process, but this algorithm is $O(n^2 \log n)$ even when fast=TRUE.

If fast=FALSE and the tracking data are calibrated (see `uere`), then with error=TRUE the variogram of the movement process (sans the telemetry-error process) is estimated using an iterative maximum-likelihood estimator that downweights more erroneous location estimates (Fleming et al, 2020). The variogram is targeted to have precision fraction of machine precision. If the data are very irregular and location errors are very homoskedastic, then this algorithm can be slow to converge at time lags where there are few data pairs. If fast=TRUE and error=TRUE, then the estimated contribution to the variogram from location error is subtracted on a per lag basis, which is less ideal for heteroskedastic errors.
variogram

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 estimator 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)


References


See Also

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

Examples

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

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

#Calculate variogram
SVF <- variogram(DATA)

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

Visual fitting of 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**

```r
cxmm.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.

`variogram.fit` is called by `ctmm.guess`, and there is usually no reason to call `variogram.fit` directly.

**Note**

If the `manipulate` package is unavailable, then `interactive` is set to `FALSE`. 
Video record animated telemetry objects.

Description

Produces an MP4 video file by animating telemetry objects.

Usage

```r
video(x, ext=extent(x), fps=60, dt=NULL, ghost=0, timestamp=FALSE, file="ctmm.mp4", res=720, col="red", pch=1, cex=NULL, lwd=1, par.list=list(), ...)```

Arguments

- `x` telemetry object or list of telemetry objects.
- `ext` Plot extent for all frames.
- `fps` Frames per viewed second.
- `dt` Tracked time per frame (not per viewed second). By default, the median timestep will be used.
- `ghost` Timescale over which image retention (ghosting) decays.
- `timestamp` Display timestamps on title.
- `file` File name for MP4 file to save. The full path can also be specified. Otherwise the working directory will be used.
- `res` Pixel resolution for square videos or pixel `c(width, height)` for rectangular videos.
- `col` Color option for telemetry data. Can be an array or list of arrays.
plot

pch Plotting symbol. Can be an array or list of arrays.
cex Relative size of plotting symbols. Only used when errors are missing.
lwd Line widths of telemetry points.
par.list List of additional arguments passed to par within animate that do not work outside of animate, like mar.
... Additional options passed to plot.telemetry.

Details

This function does not interpolate locations to make smooth animations. For that, please use predict or simulate outputs instead of a raw tracking data.

Value

Saves an MP4 file named file to the working directory.

Note

Further animation and ffmpeg options can be set via ani.options.

Author(s)

C. H. Fleming.

See Also

plot, plot.telemetry, ani.options

Examples

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

# temporary file to store videos for CRAN compliance
FILE <- tempfile("ctmm",fileext=".mp4")
# you will likely want to save your video elsewhere
# the working directory is the default location

# create guess object
GUESS <- ctmm.guess(coati[[2]],interactive=FALSE)
# in general, use ctmm.select instead of ctmm.fit
FIT <- ctmm.fit(coati[[2]],GUESS)

# consider a few hours of consecutive sampling, at 1 minute per frame
t <- seq(coati[[2]]$t[19],coati[[2]]$t[27],by=60)

# tau[velocity] is a natural scale to demonstrate persistance of motion
ghost <- FIT$tau[2]
# predicted locations each minute
PRED <- predict(coati[[2]], FIT, t = t)

# most likely path
video(PRED, error = FALSE, pch = 16, ghost = ghost, file = FILE)

# prediction (distribution)
video(PRED, error = 3, file = FILE)

# conditional simulations
SIMS <- lapply(1:6, function(i) { simulate(coati[[2]], FIT, t = t) })

# random paths
video(SIMS, pch = 16, ghost = ghost, file = FILE)

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 (roger-cunha@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, pelican, turtle.

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

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

* datasets
  - buffalo, 13
  - coati, 15
  - gazelle, 37
  - jaguar, 39
  - pelican, 55
  - turtle, 79
  - wolf, 89

  %%% (Unit conversion), 82
  %%, 36

  agde (homerange), 38
  akde, 6, 12, 15, 32, 34, 38, 39, 42, 46–49, 55, 60, 67–69, 79
  ani.options, 88
  annotate, 69
  annotate (color), 16
  as.factor, 69
  as.sf (export), 32
  as.telemetry, 8, 13, 16, 24, 29, 34, 37, 39, 53, 56, 64, 80, 81, 89

  bandwidth, 6–8, 10, 38
  buffalo, 13, 16, 37, 39, 56, 80, 89

  cleave (select), 70
  cluster, 14, 46
  coati, 13, 15, 37, 39, 56, 80, 89
  color, 16
  compass, 24
  compass (projection), 62
  correlogram, 62, 84, 85
  correlogram (residuals.ctmm), 64
  CRS, 63
  ctmm, 18, 81
  ctmm-FAQ, 4, 22
  ctmm-faq (ctmm-FAQ), 22
  ctmm-package, 3
  ctmm.boot, 22, 24
  ctmm.fit, 12, 15, 24, 25, 28, 30, 46, 51, 55, 60, 62, 70, 72, 73, 76, 77, 81, 87
  ctmm.guess, 22, 24
  ctmm.guess (variogram.fit), 86
  ctmm.select, 26, 27, 42, 77
  difference, 26
  dimfig (format), 36
  distance, 28, 55
  distances (difference), 26
  dt.plot, 11, 29
  emulate, 30, 74, 75
  encounter, 31, 55
  Exp (Log), 40
  export, 32
  extent, 7, 34, 60, 61
  extent.ctmm-method (extent), 34
  extent.data.frame-method (extent), 34
  extent.list-method (extent), 34
  extent.matrix-method (extent), 34
  extent.telemetry-method (extent), 34
  extent,UD-method (extent), 34
  extent,variogram-method (extent), 34
  format, 36
  fread, 10
  gazelle, 13, 16, 37, 39, 56, 80, 89
  Gmedian, 63
  head (as.telemetry), 8
  homerange, 38
  jaguar, 13, 16, 37, 39, 56, 80
  lasso (select), 70
  Log, 40
  mag (residuals.ctmm), 64
  marquee (select), 70
INDEX

mean, 40
mean.ctmm, 41
mean.UD, 7, 8
mean.UD (mean.ctmm), 41
mean.variogram, 43, 84, 85
median (projection), 62
meta, 15, 40, 44

glm, 50, 51
npr, 46

occurrence, 34, 47, 47, 67
optim, 20–22, 50, 51
optimize, 50, 51
optimizer, 20, 21, 50, 70
outlie, 9, 10, 23, 52
outlier (outlie), 52
overlap, 28, 32, 42, 53

par, 88
parse_date, 9, 10
pelican, 13, 16, 37, 39, 55, 80, 89
periodogram, 56
pkde (akde), 6
plot, 57, 60, 62, 88
plot (plot.telemetry), 58
plot.outlie (outlie), 52
plot.periodogram (periodogram), 56
plot.telemetry, 10, 13, 16, 17, 35, 37, 39, 56, 58, 71, 80, 88, 89
plot.variogram, 35, 43, 51, 66, 65, 85, 87
predict, 53, 88
predict (simulate.ctmm), 71
predict.ctmm, 27
projection, 24, 62
projection.ctmm-method (projection), 62
projection, list-method (projection), 62
projection, NULL-method (projection), 62
projection, telemetry-method (projection), 62
projection, UD-method (projection), 62
projection<- (list-method (projection), 62
projection<- (telemetry-method (projection), 62
proximity (difference), 26

raster, 39, 69
raster, UD-method (export), 32

rates (encounter), 31
read.csv, 10
res, 7
residuals (residuals.ctmm), 64
residuals.ctmm, 64
residuals.telemetry, 81
revisitation, 8, 66
rsf.fit, 38, 68

select, 70
sigfig (format), 36
simulate, 75, 88
simulate (simulate.ctmm), 71
simulate.ctmm, 30, 71
SpatialPoints.telemetry, 10, 60
SpatialPoints.telemetry (export), 32
SpatialPointsDataFrames.telemetry (export), 32
SpatialPolygonsDataFrames.telemetry (export), 32
SpatialPolygonsDataFrame.telemetry (export), 32
Speeds (speed), 73
spTransform, 9
strptime, 9, 10
suitability, 59
suitability (homerange), 38
summary.ctmm, 22, 70, 75, 76, 78
summary.telemetry (as.telemetry), 8
summary.UD, 77, 78
summary.URE (uere), 80
tail (as.telemetry), 8
tbind (as.telemetry), 8
text, 63
turtle, 13, 16, 37, 39, 56, 79, 89

uere, 10, 80, 80, 84
uere<- (uere), 80
unit, 83

Unit conversion, 82

variogram, 43, 61, 62, 66, 83, 87
variogram.fit, 22, 62, 72, 85, 86
video, 87

wolf, 13, 16, 37, 39, 56, 80, 89
writeOGR, 33
writeRaster, 33
writeRaster, UD, character-method (export), 32
writeShapefile (export), 32

zoom, 24
zoom, list-method (plot.telemetry), 58
zoom, telemetry-method (plot.telemetry), 58
zoom, UD-method (plot.telemetry), 58
zoom, variogram-method (plot.variogram), 61