Package: crawl (via r-universe)

October 22, 2024

Type Package

Title Fit Continuous-Time Correlated Random Walk Models to Animal Movement Data

Version 2.3.0

Date 2022-10-06

Depends R (>= 3.4.0)

Imports mvtnorm, Rcpp (>= 0.11.1), methods, dplyr, sf, sp, tibble, magrittr, lubridate, purrr, rlang

LinkingTo Rcpp, RcppArmadillo

Description Fit continuous-time correlated random walk models with time indexed covariates to animal telemetry data. The model is fit using the Kalman-filter on a state space version of the continuous-time stochastic movement process.

License CC0

Encoding UTF-8

LazyLoad yes

ByteCompile TRUE

NeedsCompilation yes

RoxygenNote 7.2.1

LazyData true

Author Devin S. Johnson [aut, cre], Josh London [aut], Brett T. McClintock [ctb], Kenady Wilson [ctb]

Maintainer Devin S. Johnson <devin.johnson@noaa.gov>

Repository CRAN

Date/Publication 2022-10-09 20:30:02 UTC

Contents

`crawl-package` *Fit Continuous-Time Correlated Random Walk Models to Animal Movement Data*

Description

The [C]orrelated [RA]ndom [W]alk [L]ibrary (I know it is not an R library, but, "crawp" did not sound as good) of R functions was designed for fitting continuous-time correlated random walk (CTCRW) models with time indexed covariates. The model is fit using the Kalman-Filter on a state space version of the continuous-time stochastic movement process.

Package:	crawl
Type:	Package
Version:	2.3.0
Date:	October 6, 2022
License:	CC0
LazyLoad:	yes

Note

This software package is developed and maintained by scientists at the NOAA Fisheries Alaska Fisheries Science Center and should be considered a fundamental research communication. The recommendations and conclusions presented here are those of the authors and this software should not be construed as official communication by NMFS, NOAA, or the U.S. Dept. of Commerce. In addition, reference to trade names does not imply endorsement by the National Marine Fisheries Service, NOAA. While the best efforts have been made to insure the highest quality, tools such as this are under constant development and are subject to change.

Author(s)

Josh London and Devin S. Johnson

Maintainer: Devin S. Johnson <devin.johnson@noaa.gov>

References

Johnson, D., J. London, M. -A. Lea, and J. Durban (2008) Continuous-time correlated random walk model for animal telemetry data. Ecology 89(5) 1208-1215.

`aic.crw` *Calculates AIC for all objects of class crwFit listed as arguments*

Description

AIC, delta AIC, and Akaike weights for all models listed as arguments.

Usage

```
aic.crw(...)
```

Arguments

`...`	a series of crwFit objects

Details

The function can either be executed with a series of 'crwFit' objects (see `crwMLE`) without the '.crwFit' suffix or the function can be called without any arguments and it will search out all 'crwFit' objects in the current workspace and produce the model selection table for all 'crwFit' objects in the workspace. Caution should be used when executing the function in this way. ALL 'crwFit' objects will be included whether or not the same locations are used! For all of the models listed as arguments (or in the workspace), AIC, delta AIC, and Akaike weights will be calculated.

Value

A table, sorted from lowest AIC value to highest.

Author(s)

Devin S. Johnson

`argosDiag2Cov` *Transform Argos diagnostic data to covariance matrix form*

Description

Using this function the user can transform the Argos diagnostic data for location error into a form usable as a covariance matrix to approximate the location error with a bivariate Gaussian distribution. The resulting data.frame should be attached back to the data with `cbind` to use with the `crwMLE` function.

Usage

```
argosDiag2Cov(Major, Minor, Orientation)
```

Arguments

`Major`	A vector containing the major axis information for each observation (na values are ok)
`Minor`	A vector containing the minor axis information for each observation (na values are ok)
`Orientation`	A vector containing the angle orientation of the Major axis from North (na values are ok)

Value

A `data.frame` with the following columns

`ln.sd.x`	The log standard deviation of the location error in the x coordinate
`ln.sd.y`	The log standard deviation of the location error in the x coordinate
`rho`	The correlation of the bivariate location error ellipse

Author(s)

Devin S. Johnson

as.flat *'Flattening' a list-form crwPredict object into a data.frame*

Description

"Flattens" a list form `crwPredict` object into a flat data.frame.

Usage

```
as.flat(predObj)
```

Arguments

`predObj`	A crwPredict object

Value

a `data.frame` version of a crwPredict list with columns for the state standard errors

Author(s)

Devin S. Johnson

See Also

`northernFurSeal` for use example

beardedSeals *Bearded Seal Location Data*

Description

Bearded Seal Location Data

Format

A data frame with 27,548 observations on 3 bearded seals in Alaska:

deployid Unique animal ID

ptt Hardware ID

instr Hardware type

date_time Time of location

type Location type

quality Argos location quality

latitude Observed latitude

longitude Observed longitude

error_radius Argos error radius

error_semimajor_axis Argos error ellipse major axis length

error_semiminor_axis Argos error ellipse minor axis length

error_ellipse_orientation Argos error ellipse degree orientation

Source

Marine Mammal Laboratory, Alaska Fisheries Science Center, National Marine Fisheries Service, NOAA 7600 Sand Point Way NE Seattle, WA 98115

crwMLE *Fit Continuous-Time Correlated Random Walk Models to Animal Telemetry Data*

Description

The function uses the Kalman filter to estimate movement parameters in a state-space version of the continuous-time movement model. Separate models are specified for movement portion and the location error portion. Each model can depend on time indexed covariates. A “haul out” model where movement is allowed to completely stop, as well as, a random drift model can be fit with this function.

Usage

```
crwMLE(data, ...)

## Default S3 method:
crwMLE(
  data,
  mov.model = ~1,
  err.model = NULL,
  activity = NULL,
  drift = FALSE,
  coord = c("x", "y"),
  proj = NULL,
  Time.name = "time",
  time.scale = NULL,
  theta = NULL,
  fixPar = NULL,
  method = "Nelder-Mead",
  control = NULL,
  constr = list(lower = -Inf, upper = Inf),
  prior = NULL,
  need.hess = TRUE,
  initialSANN = list(maxit = 200),
  attempts = 1,
  retrySD = 1,
  skip_check = FALSE,
  ...
)

## S3 method for class 'SpatialPoints'
crwMLE(
  data,
  mov.model = ~1,
  err.model = NULL,
  activity = NULL,
  drift = FALSE,
  Time.name = "time",
  time.scale = NULL,
  theta = NULL,
  fixPar = NULL,
  method = "Nelder-Mead",
  control = NULL,
  constr = list(lower = -Inf, upper = Inf),
  prior = NULL,
  need.hess = TRUE,
  initialSANN = list(maxit = 200),
  attempts = 1,
  retrySD = 1,
  skip_check = FALSE,
  coord = NULL,
  ...
)

## S3 method for class 'sf'
crwMLE(
  data,
  mov.model = ~1,
  err.model = NULL,
  activity = NULL,
  drift = FALSE,
  Time.name = "time",
  time.scale = NULL,
  theta = NULL,
  fixPar = NULL,
  method = "Nelder-Mead",
  control = NULL,
  constr = list(lower = -Inf, upper = Inf),
  prior = NULL,
  need.hess = TRUE,
  initialSANN = list(maxit = 200),
  attempts = 1,
  retrySD = 1,
  skip_check = FALSE,
  ...
)
```

Arguments

`data` a data set of location observations as a data.frame, tibble, SpatialPointsDataFrame ('sp' package), or a data.frame of class 'sf' that contains a geometry column of type `sfc_POINT`

`...` further arguments passed to or from other methods

`mov.model` formula object specifying the time indexed covariates for movement parameters.

`err.model` A 2-element list of formula objects specifying the time indexed covariates for location error parameters.

`activity` formula object giving the covariate for the activity (i.e., stopped or fully moving) portion of the model.

`drift` logical indicating whether or not to include a random drift component. For most data this is usually not necessary. See `northernFurSeal` for an example using a drift model.

`coord` A 2-vector of character values giving the names of the "X" and "Y" coordinates in `data`. Ignored if `data` inherits class 'sf' or 'sp'.

`proj` A valid epsg integer code or proj4string for `data` that does not inherit either 'sf' or 'sp'. A valid 'crs' list is also accepted. Otherwise, ignored.

`Time.name` character indicating name of the location time column. It is strongly preferred that this column be of type POSIXct and in UTC.

`time.scale` character. Scale for conversion of POSIX time to numeric for modeling. Defaults to "hours" and most users will not need to change this.

`theta` starting values for parameter optimization.

`fixPar` Values of parameters which are held fixed to the given value.

`method` Optimization method that is passed to `optim`.

`control` Control list which is passed to `optim`.

`constr` Named list with elements `lower` and `upper` that are vectors the same length as theta giving the box constraints for the parameters

`prior` A function returning the log-density function of the parameter prior distribution. THIS MUST BE A FUNCTION OF ONLY THE FREE PARAMETERS. Any fixed parameters should not be included.

`need.hess` A logical value which decides whether or not to evaluate the Hessian for parameter standard errors

`initialSANN` Control list for `optim` when simulated annealing is used for obtaining start values. See details

`attempts` The number of times likelihood optimization will be attempted in cases where the fit does not converge or is otherwise non-valid

`retrySD`	optional user-provided standard deviation for adjusting starting values when attempts > 1. Default value is 1.
`skip_check`	Skip the likelihood optimization check and return the fitted values. Can be useful for debugging problem fits.

Details

- A full model specification involves 4 components: a movement model, an activity model, 2 location error models, and a drift indication. The movement model (`mov.model`) specifies how the movement parameters should vary over time. This is a function of specified, time-indexed, covariates. The movement parameters (sigma for velocity variation and beta for velocity autocorrelation) are both modeled with a log link as par = exp(eta), where eta is the linear predictor based on the covariates. The `err.model` specification is a list of 2 such models, one for "X (longitude)" and one for "Y (latitude)" (in that order) location error. If only one location error model is given, it is used for both coordinates (parameter values as well). If `drift.model` is set to `TRUE`, then, 2 additional parameters are estimated for the drift process, a drift variance and a beta multiplier.
- `theta` and `fixPar` are vectors with the appropriate number or parameters. `theta` contains only those parameters which are to be estimated, while `fixPar` contains all parameter values with `NA` for parameters which are to be estimated.
- The data set specified by `data` must contain a numeric or POSIXct column which is used as the time index for analysis. The column name is specified by the `Time.name` argument and it is strongly suggested that this column be of POSIXct type and in UTC. If a POSIXct column is used it is internally converted to a numeric vector with units of `time.scale`. `time.scale` defaults to NULL and an appropriate option will be chosen ("seconds","minutes","days","weeks") based on the median time interval. The user can override this by specifying one of those time intervals directly. If a numeric time vector is used, then the `time.scale` is ignored and there is no adjustment to the data. Also, for activity models, the activity covariate must be between 0 and 1 inclusive, with 0 representing complete stop of the animal (no true movement, however, location error can still occur) and 1 represent unhindered movement. The coordinate location should have `NA` where no location is recorded, but there is a change in the movement covariates.
- The CTCRW models can be difficult to provide good initial values for optimization. If `initialSANN` is specified then simulated annealing is used first to obtain starting values for the specified optimization method. If simulated annealing is used first, then the returned `init` list of the crwFit object will be a list with the results of the simulated annealing optimization.
- The `attempts` argument instructs `crwMLE` to attempt a fit multiple times. Each time, the fit is inspected for convergence, whether the covariance matrix could be calculated, negative values in the diag of the covariance matrix, or NA values in the standard errors. If, after n attempts, the fit is still not valid a `simpleError` object is returned. Users should consider increasing the number of attempts OR adjusting the standard deviation value for each attempt by setting `retrySD`. The default value for `retrySD` is 1, but users may need to increase or decrease to find a valid fit. Adjusting other model parameters may also be required.

Value

A list with the following elements:

`par`	Parameter maximum likelihood estimates (including fixed parameters)
`estPar`	MLE without fixed parameters
`se`	Standard error of MLE
`ci`	95% confidence intervals for parameters
`Cmat`	Parameter covariance matrix
`loglik`	Maximized log-likelihood value
`aic`	Model AIC value
`coord`	Coordinate names provided for fitting
`fixPar`	Fixed parameter values provided
`convergence`	Indicator of convergence (0 = converged)
`message`	Messages given by `optim` during parameter optimization
`activity`	Model provided for stopping variable
`drift`	Logical value indicating random drift model
`mov.model`	Model description for movement component
`err.model`	Model description for location error component
`n.par`	number of parameters
`nms`	parameter names
`n.mov`	number of movement parameters
`n.errX`	number or location error parameters for “longitude” error model
`n.errY`	number or location error parameters for “latitude” error model
`stop.mf`	covariate for stop indication in stopping models
`polar.coord`	Logical indicating coordinates are polar latitude and longitude
`init`	Initial values for parameter optimization
`data`	Original data.frame used to fit the model
`lower`	The lower parameter bounds
`upper`	The upper parameter bounds
`need.hess`	Logical value
`runTime`	Time used to fit model

Author(s)

Devin S. Johnson, Josh M. London

crwN2ll *-2 * log-likelihood for CTCRW models*

Description

This function is designed for primary use within the crwMLE model fitting function. But, it can be accessed for advanced R and crawl users. Uses the state-space parameterization and Kalman filter method presented in Johnson et al. (2008).

Usage

```
crwN2ll(
  theta,
  fixPar,
  y,
  noObs,
  delta,
  mov.mf,
  err.mfX,
  err.mfY,
  rho = NULL,
  activity = NULL,
  n.errX,
  n.errY,
  n.mov,
  driftMod,
  prior,
  need.hess,
  constr = list(lower = -Inf, upper = Inf)
)
```

Arguments

`theta`	parameter values.
`fixPar`	values of parameters held fixed (contains `NA` for `theta` values).
`y`	N by 2 matrix of coordinates with the longitude coordinate in the first column.
`noObs`	vector with 1 for unobserved locations, and 0 for observed locations.
`delta`	time difference to next location.
`mov.mf`	Movement covariate data.
`err.mfX`	longitude error covariate data.
`err.mfY`	latitude error covariate data.
`rho`	A vector of known correlation coefficients for the error model, typically used for modern ARGOS data.
`activity`	Stopping covariate (= 0 if animal is not moving).

`n.errX` number or longitude error parameters.

`n.errY` number of latitude error parameters.

`n.mov` number or movement parameters.

`driftMod` Logical. indicates whether a drift model is specified.

`prior` Function of theta that returns the log-density of the prior

`need.hess` Whether or not the Hessian will need to be calculated from this call

`constr` Named list giving the parameter constraints

Details

This function calls compiled C++ code which can be viewed in the `src` directory of the crawl source package.

Value

-2 * log-likelihood value for specified CTCRW model.

Author(s)

Devin S. Johnson

References

Johnson, D., J. London, M. -A. Lea, and J. Durban. 2008. Continuous-time model for animal telemetry data. Ecology 89:1208-1215.

See Also

`crwMLE`

`crwPostIS` *Simulate a value from the posterior distribution of a CTCRW model*

Description

The crwPostIS draws a set of states from the posterior distribution of a fitted CTCRW model. The draw is either conditioned on the fitted parameter values or "full" posterior draw with approximated parameter posterior

Usage

```
crwPostIS(object.sim, fullPost = TRUE, df = Inf, scale = 1, thetaSamp = NULL)
```

Arguments

`object.sim` A crwSimulator object from `crwSimulator`.

`fullPost` logical. Draw parameter values as well to simulate full posterior

`df` degrees of freedom for multivariate t distribution approximation to parameter posterior

`scale` Extra scaling factor for t distribution approximation

`thetaSamp` If multiple parameter samples are available in object.sim, setting `thetaSamp=n` will use the nth sample. Defaults to the last.

Details

The crwPostIS draws a posterior sample of the track state matrices. If fullPost was set to TRUE when the object.sim was build in crwSimulator then a pseudo-posterior draw will be made by first sampling a parameter value from a multivariate t distribution which approximates the marginal posterior distribution of the parameters. The covariance matrix from the fitted model object is used to scale the MVt approximation. In addition, the factor "scale" can be used to further adjust the approximation. Further, the parameter simulations are centered on the fitted values.

To correct for the MVt approximation, the importance sampling weight is also supplied. When calculating averages of track functions for Bayes estimates one should use the importance sampling weights to calculate a weighted average (normalizing first, so the weights sum to 1).

Value

List with the following elements:

`alpha.sim.y` A matrix a simulated latitude state values

`alpha.sim.x` Matrix of simulated longitude state values

`locType` Indicates prediction types with a "p" or observation times with an "o"

`Time` Initial state covariance for latitude

`loglik` log likelihood of simulated parameter

`par` Simulated parameter value

`log.isw` non normalized log importance sampling weight

Author(s)

Devin S. Johnson

See Also

See `demo(northernFurSealDemo)` for example.

crwPredict *Predict animal locations and velocities using a fitted CTCRW model and calculate measurement error fit statistics*

Description

The `crwMEfilter` function uses a fitted model object from `crwMLE` to predict animal locations (with estimated uncertainty) at times in the original data set and supplemented by times in `predTime`. If `speedEst` is set to `TRUE`, then animal log-speed is also estimated. In addition, the measurement error shock detection filter of de Jong and Penzer (1998) is also calculated to provide a measure for outlier detection.

Usage

```
crwPredict(object.crwFit, predTime = NULL, return.type = "minimal", ...)
```

Arguments

`object.crwFit`	A model object from `crwMLE`.
`predTime`	vector of desired prediction times (numeric or POSIXct). Alternatively, a character vector specifying a time interval (see Details).
`return.type`	character. Should be one of `"minimal"`,`"flat"`,`"list"` (see Details).
`...`	Additional arguments for testing new features

Details

The requirements for `data` are the same as those for fitting the model in `crwMLE`.

- ("predTime") `predTime` can be either passed as a separate vector of POSIXct or numeric values for all prediction times expected in the returned object. Note, previous versions of `crwPredict` would return both times specified via `predTime` as well as each original observed time. This is no longer the default (see return.type). If the original data were provided as a POSIXct type, then `crwPredict` can derive a sequence of regularly spaced prediction times from the original data. This is specified by providing a character string that corresponds to the `by` argument of the `seq.POSIXt` function (e.g. '1 hour', '30 mins'). `crwPredict` will round the first observed time up to the nearest unit (e.g. '1 hour' will round up to the nearest hour, '30 mins' will round up to the nearest minute) and start the sequence from there. The last observation time is truncated down to the nearest unit to specify the end time.

Value

There are three possible return types specified with `return.type`:

`minimal`	a data.frame with a minimal set of columns: `date_time`,`mu.x`,`mu.y`,`se.mu.x`,`se.mu.y`
`flat`	a data set is returned with the columns of the original data plus the state estimates, standard errors (se), and speed estimates

`list` List with the following elements:

`originalData` A data.frame with `data` merged with `predTime`.

`alpha.hat` Predicted state

`Var.hat` array where `Var.hat[,,i]` is the prediction covariance matrix for `alpha.hat[,i]`.

Author(s)

Devin S. Johnson

References

de Jong, P. and Penzer, J. (1998) Diagnosing shocks in time series. Journal of the American Statistical Association 93:796-806.

`crwPredictPlot` *Plot CRW predicted object*

Description

Creates 2 types of plots of a crwPredict object: a plot of both coordinate axes with prediction intervals and a plot of just observed locations and predicted locations.

Usage

```
crwPredictPlot(object, plotType = "ll", ...)
```

Arguments

`object` `crwPredict` object.

`plotType` type of plot has to be one of the following: “map” or “ll” (default).

`...` Further arguments passed to plotting commands.

Value

A plot.

Author(s)

Devin S. Johnson and Sebastian Luque

See Also

See `demo(northernFurSealDemo)` for additional examples.

crwSamplePar *Create a weighted importance sample for posterior predictive track simulation.*

Description

The crwSamplePar function uses a fitted model object from crwMLE and a set of prediction times to construct a list from which crwPostIS will draw a sample from either the posterior distribution of the state vectors conditional on fitted parameters or a full posterior draw from an importance sample of the parameters.

Usage

```
crwSamplePar(
  object.sim,
  method = "IS",
  size = 1000,
  df = Inf,
  grid.eps = 1,
  crit = 2.5,
  scale = 1,
  quad.ask = T,
  force.quad
)
```

Arguments

object.sim	A simulation object from crwSimulator.
method	Method for obtaining weights for movement parameter samples
size	Size of the parameter importance sample
df	Degrees of freedom for the t approximation to the parameter posterior
grid.eps	Grid size for method="quadrature"
crit	Criterion for deciding "significance" of quadrature points (difference in log-likelihood)
scale	Scale multiplier for the covariance matrix of the t approximation
quad.ask	Logical, for method='quadrature'. Whether or not the sampler should ask if quadrature sampling should take place. It is used to stop the sampling if the number of likelihood evaluations would be extreme.
force.quad	A logical indicating whether or not to force the execution of the quadrature method for large parameter vectors.

Details

The crwSamplePar function uses the information in a crwSimulator object to create a set of weights for importance sample-resampling of parameters in a full posterior sample of parameters and locations using crwPostIS. This function is usually called from crwPostIS. The average user should have no need to call this function directly.

Value

List with the following elements:

`x`	Longitude coordinate with NA at prediction times
`y`	Similar to above for latitude
`locType`	Indicates prediction types with a "p" or observation times with an "o"
`P1.y`	Initial state covariance for latitude
`P1.x`	Initial state covariance for longitude
`a1.y`	Initial latitude state
`a1.x`	Initial longitude state
`n.errX`	number of longitude error model parameters
`n.errY`	number of latitude error model parameters
`delta`	vector of time differences
`driftMod`	Logical. indicates random drift model
`stopMod`	Logical. Indicated stop model fitted
`stop.mf`	stop model design matrix
`err.mfX`	Longitude error model design matrix
`err.mfY`	Latitude error model design matrix
`mov.mf`	Movement model design matrix
`fixPar`	Fixed values for parameters in model fitting
`Cmat`	Covariance matrix for parameter sampling distribution
`Lmat`	Cholesky decomposition of Cmat
`par`	fitted parameter values
`N`	Total number of locations
`loglik`	log likelihood of the fitted model
`Time`	vector of observation times
`coord`	names of coordinate vectors in original data
`Time.name`	Name of the observation times vector in the original data
`thetaSampList`	A list containing a data frame of parameter vectors and their associated probabilities for a resample

Author(s)

Devin S. Johnson

See Also

See `demo(northernFurSealDemo)` for example.

crwSimulator — *Construct a posterior simulation object for the CTCRW state vectors*

Description

The crwSimulator function uses a fitted model object from crwMLE and a set of prediction times to construct a list from which crwPostIS will draw a sample from either the posterior distribution of the state vectors conditional on fitted parameters or a full posterior draw from an importance sample of the parameters.

Usage

```
crwSimulator(
  object.crwFit,
  predTime = NULL,
  method = "IS",
  parIS = 1000,
  df = Inf,
  grid.eps = 1,
  crit = 2.5,
  scale = 1,
  quad.ask = TRUE,
  force.quad
)
```

Arguments

`object.crwFit` A model object from crwMLE.

`predTime` vector of additional prediction times.

`method` Method for obtaining weights for movement parameter samples

`parIS` Size of the parameter importance sample

`df` Degrees of freedom for the t approximation to the parameter posterior

`grid.eps` Grid size for `method="quadrature"`

`crit` Criterion for deciding "significance" of quadrature points (difference in log-likelihood)

`scale` Scale multiplier for the covariance matrix of the t approximation

`quad.ask` Logical, for method='quadrature'. Whether or not the sampler should ask if quadrature sampling should take place. It is used to stop the sampling if the number of likelihood evaluations would be extreme.

`force.quad` A logical indicating whether or not to force the execution of the quadrature method for large parameter vectors.

Details

The crwSimulator function produces a list and preprocesses the necessary components for repeated track simulation from a fitted CTCRW model from `crwMLE`. The `method` argument can be one of `"IS"` or `"quadrature"`. If method="IS" is chosen standard importance sampling will be used to calculate the appropriate weights via t proposal with df degrees of freedom. If df=Inf (default) then a multivariate normal distribution is used to approximate the parameter posterior. If `method="quadrature"`, then a regular grid over the posterior is used to calculate the weights. The argument `grid.eps` controls the quadrature grid. The arguments are approximately the upper and lower limit in terms of standard deviations of the posterior. The default is `grid.eps`, in units of 1sd. If `object.crwFit` was fitted with `crwArgoFilter`, then the returned list will also include `p.out`, which is the approximate probability that the observation is an outlier.

Value

List with the following elements:

`x`	Longitude coordinate with NA at prediction times
`y`	Similar to above for latitude
`locType`	Indicates prediction types with a "p" or observation times with an "o"
`P1.y`	Initial state covariance for latitude
`P1.x`	Initial state covariance for longitude
`a1.y`	Initial latitude state
`a1.x`	Initial longitude state
`n.errX`	number of longitude error model parameters
`n.errY`	number of latitude error model parameters
`delta`	vector of time differences
`driftMod`	Logical. indicates random drift model
`stopMod`	Logical. Indicated stop model fitted
`stop.mf`	stop model design matrix
`err.mfX`	Longitude error model design matrix
`err.mfY`	Latitude error model design matrix
`mov.mf`	Movement model design matrix
`fixPar`	Fixed values for parameters in model fitting
`Cmat`	Covaraince matrix for parameter sampling distribution
`Lmat`	Cholesky decomposition of Cmat
`par`	fitted parameter values
`N`	Total number of locations
`loglik`	log likelihood of the fitted model
`Time`	vector of observation times
`coord`	names of coordinate vectors in original data
`Time.name`	Name of the observation times vector in the original data
`thetaSampList`	A list containing a data frame of parameter vectors and their associated probabilities for a resample

Author(s)

Devin S. Johnson

See Also

See `demo(northernFurSealDemo)` for example.

`crw_as_sf` *Coerce to sf/sfc object*

Description

Provides reliable conversion of `"crwIS"` and `"crwPredict"` objects into simple features objects supported in the `"sf"` package. Both `"sf"` objects with "POINT" geometry and `"sfc_LINESTRING"` objects are created. Coercion of `"crwPredict"` objects to `"sfc_LINESTRING"` has an option `"group"` argument when the `"crwPredict"` object includes predictions from multiple deployments. The grouping column will be used and a tibble of multiple `"sf_LINESTRING"` objects will be returned

Usage

```
crw_as_sf(data, ftype, locType, group)

## S3 method for class 'crwIS'
crw_as_sf(data, ftype, locType = c("p", "o", "f"), group = NULL, ...)

## S3 method for class 'crwPredict'
crw_as_sf(data, ftype, locType = c("p", "o", "f"), group = NULL, ...)

## S3 method for class 'list'
crw_as_sf(data, ftype, locType = c("p", "o", "f"), ...)
```

Arguments

`data`	an object of class `"crwIS"` or `"crwPredict"`
`ftype`	character of either "POINT" or "LINESTRING" specifying the feature type
`locType`	character vector of location points to include ("p","o")
`group`	(optional) character specifying the column to group by for multiple LINESTRING features
`...`	Additional arguments that are ignored

Methods (by class)

- `crw_as_sf(crwIS)`: coerce crwIS object to sf (POINT or LINESTRING geometry)
- `crw_as_sf(crwPredict)`: coerce crwPredict object to sf (POINT or LINESTRING geometry)
- `crw_as_sf(list)`: coerce list of crwIS objects to sf (LINESTRING or MULTILINESTRING geometry)

crw_as_tibble *Coerce crawl objects (crwIS and crwPredict) to tibbles*

Description

Coerce crawl objects (crwIS and crwPredict) to tibbles

Usage

```
crw_as_tibble(crw_object, ...)

## S3 method for class 'crwIS'
crw_as_tibble(crw_object, ...)

## S3 method for class 'crwPredict'
crw_as_tibble(crw_object, ...)

## S3 method for class 'tbl'
crw_as_tibble(crw_object, ...)
```

Arguments

`crw_object`	an object of class `"crwIS"` or `"crwPredict"`
`...`	Additional arguments that are ignored

Methods (by class)

- `crw_as_tibble(crwIS)`: coerce crwIS object to tibble
- `crw_as_tibble(crwPredict)`: coerce crwPredict object to tibble
- `crw_as_tibble(tbl)`:

Author(s)

Josh M. London

detect_timescale *Detect appropriate time scale for movement analysis*

Description

This function examines the time vector and evaluates the median time interval. With this, we determine what the best time scale for the movement model is likely to be.

Usage

```
detect_timescale(time_vector)
```

Arguments

`time_vector` a vector of class POSIXct

Value

character of either "seconds","minutes","hours","days","weeks"

`displayPar` *Display the order of parameters along with fixed values and starting values*

Description

This function takes the model specification arguments to the `crwMLE` function and displays a table with the parameter names in the order that `crwMLE` will use during model fitting. This is useful for specifying values for the `fixPar` or `theta` (starting values for free parameters) arguments.

Usage

```
displayPar(
  mov.model = ~1,
  err.model = NULL,
  activity = NULL,
  drift = FALSE,
  data,
  Time.name,
  theta,
  fixPar,
  ...
)
```

Arguments

`mov.model` formula object specifying the time indexed covariates for movement parameters.

`err.model` A 2-element list of formula objects specifying the time indexed covariates for location error parameters.

`activity` formula object giving the covariate for the stopping portion of the model.

`drift` logical indicating whether or not to include a random drift component.

`data` data.frame object containing telemetry and covariate data. A `SpatialPointsDataFrame` object from the package 'sp' will also be accepted.

`Time.name` character indicating name of the location time column

`theta` starting values for parameter optimization.

`fixPar` Values of parameters which are held fixed to the given value.

`...` Additional arguments (probably for testing new features.)

Value

A data frame with the following columns

`ParNames`	The names of the parameters specified by the arguments.
`fixPar`	The values specified by the `fixPar` argument for fixed values of the parameters. In model fitting, these values will remain fixed and will not be estimated.
`thetaIndex`	This column provides the index of each element of the theta argument and to which parameter it corresponds.
`thetaStart`	If a value is given for the `theta` argument it will be placed in this column and its elements will correspond to the `thetaIdx` column.

Author(s)

Devin S. Johnson

See Also

`demo(northernFurSealDemo)` for example.

`expandPred` *Expand a time indexed data set with additional prediction times*

Description

Expands a covariate data frame (or vector) that has a separate time index by inserting prediction times and duplicating the covariate values for all prediction time between subsequent data times.

Usage

```
expandPred(x, Time = "Time", predTime, time.col = FALSE)
```

Arguments

`x`	Data to be expanded.
`Time`	Either a character naming the column which contains original time values, or a numeric vector of original times
`predTime`	prediction times to expand data
`time.col`	Logical value indicating whether to attach the new times to the expanded data

Value

data.frame expanded by `predTime`

Author(s)

Devin S. Johnson

Examples

```
#library(crawl)
origTime <- c(1:10)
x <- cbind(rnorm(10), c(21:30))
predTime <- seq(1,10, by=0.25)
expandPred(x, Time=origTime, predTime, time.col=TRUE)
```

---

fillCols *Fill missing values in data set (or matrix) columns for which there is a single unique value*

Description

Looks for columns in a data set that have a single unique non-missing value and fills in all NA with that value

Usage

```
fillCols(data)
```

Arguments

data	data.frame

Value

data.frame

Author(s)

Devin S. Johnson

Examples

```
#library(crawl)
data1 <- data.frame(constVals=rep(c(1,NA),5), vals=1:10)
data1[5,2] <- NA
data1
data2 <- fillCols(data1)
data2

mat1 <- matrix(c(rep(c(1,NA),5), 1:10), ncol=2)
mat1[5,2] <- NA
mat1
mat2 <- fillCols(mat1)
mat2
```

`fix_path` *fix_path function id depreciated.*

Description

fix_path function id depreciated.

Usage

```
fix_path(...)
```

Arguments

`...`	Any arguments are ignored.

`flatten` *'Flattening' a list-form crwPredict object into a data.frame*

Description

"Flattens" a list form `crwPredict` object into a flat data.frame.

Usage

```
flatten(predObj)
```

Arguments

`predObj`	A crwPredict object

Value

a `data.frame` version of a crwPredict list with columns for the state standard errors

Author(s)

Devin S. Johnson

See Also

`northernFurSeal` for use example

`harborSeal` *Harbor seal location data set used in Johnson et al. (2008)*

Description

Harbor seal location data set used in Johnson et al. (2008)

Format

A data frame with 7059 observations on the following 5 variables.

Time a numeric vector.

latitude a numeric vector.

longitude a numeric vector.

DryTime a numeric vector.

Argos_loc_class a factor with levels `0` `1` `2` `3` `A` `B`.

Author(s)

Devin S. Johnson

Source

Marine Mammal Laboratory, Alaska Fisheries Science Center, National Marine Fisheries Service, NOAA 7600 Sand Point Way NE Seattle, WA 98115

References

Johnson, D., J. London, M. -A. Lea, and J. Durban (2008) Continuous-time random walk model for animal telemetry data. Ecology 89:1208-1215.

`harborSeal_sf` *Harbor seal location data updated since Johnson et al. (2008)*

Description

The original location data used in Johnson et al. (2008) was geographic (latitude/longitude) (but not explicitly documented) and provided as a simple data frame. This data updates the data to a Simple Feature Collection (as part of the sf package) with the CRS explicitly set.

Format

A Simple Feature Collection with 7059 features and 3 fields.

Time a numeric vector.

DryTime a numeric vector.

Argos_loc_class a factor with levels `0` `1` `2` `3` `A` `B`.

geometry a list column with geometry data; CRS = EPSG:4326

Author(s)

Josh M. London

Source

Marine Mammal Laboratory, Alaska Fisheries Science Center, National Marine Fisheries Service, NOAA 7600 Sand Point Way NE Seattle, WA 98115

References

Johnson, D., J. London, M. -A. Lea, and J. Durban (2008) Continuous-time random walk model for animal telemetry data. Ecology 89:1208-1215.

`intToPOSIX` *Reverse as.numeric command that is performed on a vector of type POSIXct*

Description

Takes integer value produced by `as.numeric(x)`, where `x` is a POSIXct vector and returns it to a POSIXct vector

Usage

```
intToPOSIX(timeVector, tz = "GMT")
```

Arguments

`timeVector`	A vector of integers produced by as.numeric applied to a PSIXct vector
`tz`	Time zone of the vector (see `as.POSIXct`).

Value

POSIXct vector

Note

There is no check that as.numeric applied to a POSIX vector produced `timeVector`. So, caution is required in using this function. It was included simply because I have found it useful

Author(s)

Devin S. Johnson

Examples

```
#library(crawl)
timeVector <- as.numeric(Sys.time())
timeVector
intToPOSIX(timeVector, tz="")
```

mergeTrackStop *Merge a location data set with a dry time (or other stopping) covariate*

Description

The function merges a location data set with a stopping variable data set.

Usage

```
mergeTrackStop(
  data,
  stopData,
  Time.name = "Time",
  interp = c("zeros", "ma0"),
  win = 2,
  constCol
)
```

Arguments

`data`	Location data.
`stopData`	stopping variable data set.
`Time.name`	character naming time index variable in both data sets
`interp`	method of interpolation.
`win`	window for "ma0" interpolation method.
`constCol`	columns in `data` for which the user would like to be constant, such as id or sex.

Details

Simply merges the data frames and interpolates based on the chosen method. Both data frames have to use the same name for the time variable. Also contains `stopType` which = "o" if observed or "p" for interpolated.

The merged data is truncated to the first and last time in the location data set. Missing values in the stopping variable data set can be interpolated by replacing them with zeros (full movement) or first replacing with zeros then using a moving average to smooth the data. Only the missing values are then replace with this smoothed data. This allows a smooth transition to full movement.

Value

Merged data.frame with new column from `stopData`. Missing values in the stopping variable will be interpolated

Author(s)

Devin S. Johnson

Examples

```
track <- data.frame(TimeVar=sort(runif(20,0,20)), x=1:20, y=20:1)
track
stopData <- data.frame(TimeVar=0:29, stopVar=round(runif(30)))
stopData
mergeTrackStop(track, stopData, Time.name="TimeVar")
```

`northernFurSeal` *Northern fur seal pup relocation data set used in Johnson et al. (2008)*

Description

Northern fur seal pup relocation data set used in Johnson et al. (2008)

Format

A data frame with 795 observations on the following 4 variables:

GMT A POSIX time vector

loc_class a factor with levels `3` `2` `1` `0` `A`.

lat a numeric vector. Latitude for the locations

long a numeric vector. Longitude for the locations

Source

Marine Mammal Laboratory, Alaska Fisheries Science Center, National Marine Fisheries Service, NOAA 7600 Sand Point Way NE Seattle, WA 98115

References

Johnson, D., J. London, M. -A. Lea, and J. Durban (2008) Continuous-time random walk model for animal telemetry data. Ecology 89:1208-1215.

tidy_crwFit *tidy-like method for crwFit object*

Description

this function mimics the approach taken by broom::tidy to present model output parameters in a tidy, data frame structure.

Usage

```
tidy_crwFit(fit)
```

Arguments

fit	crwFit object from crawl::crwMLE

[.crwIS *Generic subset/bracket method for crwIS classes*

Description

Generic subset/bracket method for crwIS classes

Usage

```
## S3 method for class 'crwIS'
x[i, ..., drop = TRUE]
```

Arguments

x	crwIS object
i	elements to extract or replace. These are numeric or character or, empty or logical. Numeric values are coerced to integer as if by as.integer
...	other arguments
drop	logical. If TRUE the result is coerced to the lowest possible dimension.

Index