

Package ‘crawl’

April 10, 2012

Type Package

Title Fit continuous-time correlated random walk models to animal movement data

Version 1.3-4

Date 2012-4-7

Author Devin S. Johnson

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

Depends mvtnorm, sp, raster

Description The (C)orrelated (RA)ndom (W)alk (L)ibrary 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.

License Unlimited

LazyLoad yes

Collate 'AllMethod.R' 'crawl-internal.R' 'crawl-package.R' 'crwMLE.R' 'crwN2ll.R' 'crwPostIS.R' 'crwPredict.R' 'crwPredictPlot.R' 'crwSamplePar.R' 'crwSimulator.R'

Repository CRAN

Date/Publication 2012-04-10 04:17:02

R topics documented:

crawl-package	2
aic.crw	3
as.flat	3
crwMLE	4
crwN2ll	8
crwPostIS	9

crwPredict	11
crwPredictPlot	12
crwSamplePar	13
crwSimulator	14
crwUseGrid	16
expandPred	17
fillCols	18
harborSeal	19
intToPOSIX	20
logSpeed	21
mergeTrackStop	21
northernFurSeal	22

Index	25
--------------	-----------

crawl-package	<i>Fit Continuous-Time Correlated Random Walk models to animal movement data</i>
---------------	--

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.

Details

Package:	crawl
Type:	Package
Version:	1.3-4
Date:	2012-4-7
License:	Unlimited
LazyLoad:	yes

Author(s)

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	<i>Calculates AIC for all objects of class crwFit listed as arguments</i>
---------	---

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

as.flat	<i>'Flattening' a list-form crwPredict object into a data.frame</i>
---------	---

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

crwMLE	<i>Fit Continuous-Time Correlated Random Walk Models to Animal Telemetry Data</i>
--------	---

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(mov.model = ~1, err.model = NULL,
       stop.model = NULL, drift.model = FALSE, data,
       coord = c("x", "y"), polar.coord, Time.name,
       initial.state, theta, fixPar, method = "L-BFGS-B",
       control = NULL,
       constr = list(lower = -Inf, upper = Inf), prior = NULL,
       need.hess = TRUE, initialSANN = NULL, attempts = 1)
```

Arguments

<code>mov.model</code>	formula object specifying the time indexed covariates for movement parameters.
<code>err.model</code>	A 2-element list of formula objects specifying the time indexed covariates for location error parameters.
<code>stop.model</code>	formula object giving the covariate for the stopping portion of the model.
<code>drift.model</code>	logical indicating whether or not to include a random drift component.
<code>data</code>	<code>data.frame</code> object containing telemetry and covariate data. A <code>'SpatialPointsDataFrame'</code> object from the package <code>'sp'</code> will also be accepted. In which case the <code>polar.coord</code> and <code>coord</code> values will be taken from the spatial data set and ignored in the arguments.
<code>coord</code>	A 2-vector of character values giving the names of the "X" and "Y" coordinates in data.

<code>polar.coord</code>	logical indicating location are in degrees latitude and longitude.
<code>Time.name</code>	character indicating name of the location time column
<code>initial.state</code>	list object containing the initial state of the Kalman filter.
<code>theta</code>	starting values for parameter optimization.
<code>fixPar</code>	Values of parameters which are held fixed to the given value.
<code>method</code>	Optimization method that is passed to <code>optim</code> .
<code>control</code>	Control list which is passed to <code>optim</code> .
<code>constr</code>	Named list with elements <code>lower</code> and <code>upper</code> that are vectors the same length as <code>theta</code> giving the box constraints for the parameters
<code>prior</code>	A function returning the log-density function of the parameter prior specification
<code>need.hess</code>	A logical value which decides whether or not to evaluate the Hessian for parameter standard errors
<code>initialSANN</code>	Control list for <code>optim</code> when simulated annealing is used for obtaining start values. See details
<code>attempts</code>	The number of times likelihood optimization will be attempted

Details

A full model specification involves 4 components: a movement model, a stopping 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 $\text{par} = \exp(\eta)$, where η is the linear predictor based on the covariates. The `err.model` specification is a list of 2 such models, one for “longitude” and one for “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. If `polar.coord`=TRUE then the ad-hoc longitude correction factor described by Johnson et al. (2008) (Ecology 89:1208-1215) is used to adjust the variance scale for the longitude model.

The `initial.state` is a list with the following elements (with the exact names):

`a1.y` A vector with initial state values for the “latitude” coordinate. It has 2 elements (location at time 1, velocity at time 1) for non-drift models and 3 elements for drift models (location at time 1, velocity at time 1, drift velocity at time 1) for drift models,

`P1.y` Covariance matrix for the state at time 1 (measure of uncertainty for your initial state) `a1.y`,

`a1.x` Same as `a1.y`, but in the “longitude” coordinate,

`P1.x` Same as `P1.y`, but in the “longitude” coordinate.

`theta` and `fixPar` are vectors with the appropriate number of 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. If a POSIXct column is used it is internally converted to a numeric vector with units of hours. If this is not appropriate for your data, it is better to convert it yourself prior to analysis with `crawl`. Also, for stopping models,

the stopping covariate must be between 0 and 1 inclusive, with 1 representing complete stop of the animal (no true movement, however, location error can still occur) and 0 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.

Value

A list with the following elements:

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

Author(s)

Devin S. Johnson

See Also[northernFurSeal](#) for additional examples.**Examples**

```

data(harborSeal)
head(harborSeal)
## Calculate Log multipliers for Argos error
argosClasses <- c("3", "2", "1", "0", "A", "B")
ArgosMultFactors <- data.frame(Argos_loc_class=argosClasses,
                              errX=log(c(1, 1.5, 4, 14, 5.21, 20.78)),
                              errY=log(c(1, 1.5, 4, 14, 11.08, 31.03)))
hsNew <- merge(harborSeal, ArgosMultFactors, by=c("Argos_loc_class"), all=TRUE)
hsNew <- hsNew[order(hsNew$Time), ]
head(hsNew)

## Initial state values
initial.dry <- list(
  a1.x=c(harborSeal$longitude[1],0),
  a1.y=c(harborSeal$latitude[1],0),
  P1.x=diag(c(1,1)),
  P1.y=diag(c(1,1))
)

##Fit model as given in Johnson et al. (2008) Ecology 89:1208-1215
## Start values for theta come from the estimates in Johnson et al. (2008)

fit1 <- crwMLE(
  mov.model=~1, err.model=list(x=~errX, y=~errY), stop.model=~DryTime,
  data=hsNew, coord=c("longitude","latitude"), polar.coord=TRUE, Time.name="Time",
  initial.state=initial.dry, fixPar=c(NA, 1, NA, 1, NA, NA, NA), theta=c(-6,-7,-4,-0.5,-1),
  control=list(maxit=2000, trace=1, REPORT=1),
)

fit1
str(fit1)

##Use simulated annealing to obtain start values and place constraints on the parameters

set.seed(123)
fit2 <- crwMLE(
  mov.model=~1, err.model=list(x=~errX, y=~errY), stop.model=~DryTime,
  data=hsNew, coord=c("longitude","latitude"), Time.name="Time", polar.coord=TRUE,
  control=list(maxit=2000, trace=1, REPORT=1),
  constr=list(lower=c(-6, -6, -Inf, -Inf, -Inf), upper=Inf),
  initialSANN=list(maxit=100, temp=5, tmax=5, trace=1, REPORT=2)
)

```

```

fit2

##See simulated annealing start values
fit2$init$par

```

```

crwN211          -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

```

crwN211(theta, fixPar, y, x, loctype, delta, a1.y, a1.x,
        P1.x, P1.y, lonAdj, mov.mf, err.mfX, err.mfY, stop.mf,
        n.errX, n.errY, n.mov, stopMod, driftMod, prior,
        need.hess, constr = list(lower = -Inf, upper = Inf))

```

Arguments

<code>theta</code>	parameter values.
<code>fixPar</code>	values of parameters held fixed (contains NA for theta values).
<code>y</code>	latitude locations.
<code>x</code>	longitude loations.
<code>loctype</code>	vector with 1 or observed location, else 0.
<code>delta</code>	time difference to next location.
<code>a1.y</code>	initial state value for latitude.
<code>a1.x</code>	initial state value for longitude.
<code>P1.x</code>	intial state covariance matrix for latitude.
<code>P1.y</code>	inital state covariance matrix for longitude.
<code>lonAdj</code>	= $1/\cos(y*\pi/180)$ as described in Johnson et al. (2008) for polar coords. = 1 for non-polar coords.
<code>mov.mf</code>	Movement covariate data.
<code>err.mfX</code>	longitude error covariate data.
<code>err.mfY</code>	latitude error covariate data.
<code>stop.mf</code>	stopping covariate.
<code>n.errX</code>	number or longitude error parameters.
<code>n.errY</code>	number of latitude error parameters.

n.mov	number of movement parameters.
stopMod	Logical. indicates whether a stop model is specified.
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 Fortran code which can be viewed in the src directory of the crawl library.

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

<code>object.sim</code>	A <code>crwSimulator</code> object from crwSimulator .
<code>fullPost</code>	logical. Draw parameter values as well to simulate full posterior
<code>df</code>	degrees of freedom for multivariate t distribution approximation to parameter posterior
<code>scale</code>	Extra scaling factor for t distribution approximation
<code>thetaSamp</code>	If multiple parameter samples are available in <code>object.sim</code> , setting <code>thetaSamp=n</code> will use the <code>n</code> th 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 psuedo-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:

<code>alpha.sim.y</code>	A matrix a simulated latitude state values
<code>alpha.sim.x</code>	Matrix of simulated longitude state values
<code>locType</code>	Indicates prediction types with a "p" or observation times with an "o"
<code>Time</code>	Initial state covariance for latitude
<code>loglik</code>	log likelihood of simulated parameter
<code>par</code>	Simulated parameter value
<code>log.isw</code>	non normalized log importance sampling weight

Author(s)

Devin S. Johnson

See Also

See [northernFurSeal](#) for example.

crwPredict	<i>Predict animal locations and velocities using a fitted CTCRW model and calculate measurement error fit statistics</i>
------------	--

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,
           speedEst = FALSE, flat = TRUE, getUseAvail = FALSE)
```

Arguments

<code>object.crwFit</code>	A model object from <code>crwMLE</code> .
<code>predTime</code>	vector of additional prediction times (numeric or <code>POSIXct</code>).
<code>speedEst</code>	logical. Estimate animal speed or not.
<code>flat</code>	logical. Should the result be returned as a flat data.frame.
<code>getUseAvail</code>	logical. This is a test function. Not for general use yet.

Details

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

Value

List with the following elements:

<code>originalData</code>	A data.frame with is data merged with <code>predTime</code> .
<code>alpha.hat.y</code>	A data.frame with predicted state values for each time. First column in latitude location (<code>mu.y</code>), second in velocity (<code>nu.y</code> or <code>theta.y</code> for drift models), and third is drift velocity (<code>gamma.y</code> if specified).
<code>alpha.hat.x</code>	longitude state predictions.
<code>Var.hat.y</code>	array where <code>Var.hat.y[, , i]</code> is the prediction covariance matrix for <code>alpha.hat.y[, i]</code> .
<code>Var.hat.x</code>	array or covariance matrices for <code>alpha.hat.x</code> .
<code>speed</code>	(If <code>speedEst=TRUE</code>) Gives log speed estimates for each time and standard errors based on delta method. If coordinates are polar, units are meters/unit Time, else, units are those specified by the coordinates.
<code>fit.test</code>	A data.frame of chi-square fit (<code>df=2</code>) statistics and naive (pointwise) p-values.

If `flat` is set to `TRUE` then a data set is returned with the columns of the original data plus the state estimates, standard errors (`se`), speed estimates, and the fit statistics and naive p-values.

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	<i>Plot CRW predicted object</i>
----------------	----------------------------------

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).

Value

A plot.

Author(s)

Devin S. Johnson and Sebastian Luque

See Also

See [northernFurSeal](#) for additional examples.

crwSamplePar	<i>Create a weighted importance sample for posterior predictive track simulation.</i>
--------------	---

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)
```

Arguments

<code>object.sim</code>	A simulation object from <code>crwSimulator</code> .
<code>method</code>	Method for obtaining weights for movement parameter samples
<code>size</code>	Size of the parameter importance sample
<code>df</code>	Degrees of freedom for the t approximation to the parameter posterior
<code>grid.eps</code>	Grid size for <code>method="quadrature"</code>
<code>crit</code>	Criterion for deciding "significance" of quadrature points (difference in log-likelihood)
<code>scale</code>	Scale multiplier for the covariance matrix of the t approximation

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:

<code>x</code>	Longitude coordinate with NA at prediction times
<code>y</code>	Similar to above for latitude
<code>locType</code>	Indicates prediction types with a "p" or observation times with an "o"
<code>P1.y</code>	Initial state covariance for latitude
<code>P1.x</code>	Initial state covariance for longitude
<code>a1.y</code>	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 [northernFurSeal](#) 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)
```

Arguments

<code>object.crwFit</code>	A model object from crwMLE .
<code>predTime</code>	vector of additional prediction times.
<code>method</code>	Method for obtaining weights for movement parameter samples
<code>parIS</code>	Size of the parameter importance sample
<code>df</code>	Degrees of freedom for the t approximation to the parameter posterior
<code>grid.eps</code>	Grid size for <code>method="quadrature"</code>
<code>crit</code>	Criterion for deciding "significance" of quadrature points (difference in log-likelihood)
<code>scale</code>	Scale multiplier for the covariance matrix of the t approximation

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:

<code>x</code>	Longitude coordinate with NA at prediction times
<code>y</code>	Similar to above for latitude
<code>locType</code>	Indicates prediction types with a "p" or observation times with an "o"
<code>P1.y</code>	Initial state covariance for latitude
<code>P1.x</code>	Initial state covariance for longitude
<code>a1.y</code>	Initial latitude state
<code>a1.x</code>	Initial longitude state
<code>n.errX</code>	number of longitude error model parameters
<code>n.errY</code>	number of latitude error model parameters
<code>delta</code>	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 [northernFurSeal](#) for example.

crwUseGrid

Compute a spatial use grid from a crawl prediction

Description

This function take a SpatialPoints object and a spatial GridTopology object from the 'sp' package and outputs the number of point locations in each grid cell

Usage

```
crwUseGrid(object, grid, subset = TRUE)
```

Arguments

object	A 'SpatialPoints' object created with the sp package. T
grid	A GridTopology object from the 'sp' package
subset	An indicator of which times should be used for calculation of the use grid. Can be a logical vector or a vector of integers, such as from a call to which

Value

A SpatialGridDataFrame with data column 'use' which gives the count of locations within the grid cell

Author(s)

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

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	<i>Fill missing values in data set (or matrix) columns for which there is a single unique value</i>
----------	---

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
```

harborSeal	<i>Harbor seal relocation data set used in Johnson et al. (2008)</i>
------------	--

Description

Harbor seal relocation 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

Polar Ecosystems Program National 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.

Examples

```
data(harborSeal)
head(harborSeal)
```

intToPOSIX	<i>Reverse as.numeric command that is performed on a vector of type POSIXct</i>
------------	---

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

<code>timeVector</code>	A vector of integers produced by <code>as.numeric</code> applied to a <code>POSIXct</code> vector
<code>tz</code>	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="")
```

logSpeed	<i>calculate estimate of log speed from a crwPredict object</i>
----------	---

Description

Calculates log speed estimates and standard errors based on the delta method. Typically this is used within the function `crwPredict`.

Usage

```
logSpeed(predx, predy, varx, vary, polar.coord)
```

Arguments

predx	matrix of longitude state estimates.
predy	matrix of latitude state estimates.
varx	an array of covariance matrices for predx.
vary	ana array of covariance matrices for predy.
polar.coord	Logical. TRUE if coordinates are polar.

Value

data.frame with columns:

ln.speed	log speed estimate (in meters/time unit if polar.coord=TRUE).
var.ln.speed	log speed estimated standard error.

Author(s)

Devin S. Johnson

mergeTrackStop	<i>Merge a location data set with a dry time (or other stopping) covariate</i>
----------------	--

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:

Time a numeric vector.

Argos_loc_class a factor with levels 0 1 2 3 A.

latitude a numeric vector.

longitude a numeric vector.

Source

Alaska Ecosystems Program National 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.

Examples

```
data(northernFurSeal)

argosClasses <- c("3", "2", "1", "0", "A", "B")
ArgosMultFactors <- data.frame(Argos_loc_class=argosClasses,
                              errX=log(c(1, 1.5, 4, 14, 5.21, 20.78)),
                              errY=log(c(1, 1.5, 4, 14, 11.08, 31.03)))
nfsNew <- merge(northernFurSeal, ArgosMultFactors,
               by=c("Argos_loc_class"), all.x=TRUE)
nfsNew <- nfsNew[order(nfsNew$Time), ]

# State starting values
initial.drift <- list(a1.x=c(189.686, 0, 0), a1.y=c(57.145, 0, 0),
                    P1.x=diag(c(0, 0.001, 0.001)),
                    P1.y=diag(c(0, 0.001, 0.001)))

##Fit random drift model
fit <- crwMLE(mov.model=~1, err.model=list(x=~errX, y=~errY), drift.model=TRUE,
            data=nfsNew, coord=c("longitude", "latitude"), polar.coord=TRUE,
            Time.name="Time", initial.state=initial.drift,
            fixPar=c(NA, 1, NA, 1, NA, NA, NA, NA),
            control=list(maxit=2000,trace=1, REPORT=10),
            initialSANN=list(maxit=300, trace=1, REPORT=1)
            )

##Make hourly location predictions
predTime <- seq(ceiling(min(nfsNew$Time)), floor(max(nfsNew$Time)), 1)
predObj <- crwPredict(object.crwFit=fit, predTime, speedEst=TRUE, flat=TRUE)
head(predObj)
crwPredictPlot(predObj)
```

```
##Create simulation object with 100 parameter draws
simObj <- crwSimulator(fit, predTime, parIS=100, df=20, scale=18/20)

## Examine IS weight distribution
w <- simObj$thetaSampList[[1]][,1]
dev.new()
hist(w*100, main='Importance Sampling Weights', sub='More weights near 1 is desirable')

##Approximate number of independent samples
round(100/(1+(sd(w)/mean(w))^2))

dev.new(bg=gray(0.75))
jet.colors <-
  colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan",
                    "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000"))
crwPredictPlot(predObj, 'map')

## Sample 20 tracks from posterior predictive distribution
iter <- 20
cols <- jet.colors(iter)
for(i in 1:iter){
  samp <- crwPostIS(simObj)
  lines(samp$alpha.sim.x[, 'mu'], samp$alpha.sim.y[, 'mu'], col=cols[i])
}
```

Index

*Topic **datasets**

harborSeal, [19](#)

northernFurSeal, [22](#)

aic.crw, [3](#)

as.flat, [3](#)

as.POSIXct, [20](#)

crawl (crawl-package), [2](#)

crawl-package, [2](#)

crwMLE, [3](#), [4](#), [8](#), [9](#), [11](#), [15](#)

crwN211, [8](#)

crwPostIS, [9](#), [13](#), [14](#)

crwPredict, [3](#), [11](#), [21](#)

crwPredictPlot, [12](#)

crwSamplePar, [13](#)

crwSimulator, [10](#), [13](#), [14](#)

crwUseGrid, [16](#)

data.frame, [4](#)

expandPred, [17](#)

fillCols, [18](#)

harborSeal, [19](#)

intToPOSIX, [20](#)

logSpeed, [21](#)

mergeTrackStop, [21](#)

northernFurSeal, [4](#), [7](#), [10](#), [12](#), [14](#), [16](#), [22](#)

optim, [5](#)