Package: availability (via r-universe)

September 5, 2024

Type Package

Title Estimating Geographic Space Available to Animals Based on Telemetry Data

Version 0.16.0

Description Estimating geographic space available to animals based on telemetry data.

URL https://github.com/AustralianAntarcticDivision/availability

BugReports https://github.com/AustralianAntarcticDivision/availability/issues

Encoding UTF-8

Depends R (>= 3.3.0)

Imports assertthat, geosphere, raster, testthat, png

Suggests crawl, knitr, ggplot2, maptools, rgdal, rmarkdown, sp, trip

License MIT + file LICENSE

LazyLoad yes

RoxygenNote 7.2.3

VignetteBuilder knitr

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

RemoteUrl https://github.com/AustralianAntarcticDivision/availability

RemoteRef HEAD

RemoteSha d0353c9b01da1613bd70cb82661434b4278b9d8b

Contents

`availability` **availability**

Description

Estimating geographic space available to animals based on telemetry data

Author(s)

Ben Raymond <`ben.raymond@aad.gov.au`>, Simon Wotherspoon, Ryan Reisinger

`calc_distbearing` *Distance and Bearing*

Description

Calculate distances in metres and bearing between successive points along track.

Usage

```
calc_distbearing(lonlat)
```

Arguments

`lonlat`	a 2-column matrix or dataframe with longitude and latitude of each point

Details

This is an internal function used by `randomize_track`.

Value

A dataframe with columns

`lon`	the longitude of the randomized track
`lat`	the latitude of the randomized track

etopoMask *Land Mask*

Description

A land mask based on the full ETOPO1 dataset

Usage

```
etopoMask(
  basename = "ETOPO1_Bed_c_geotiff",
  path = ".",
  tmp = path,
  land = FALSE
)
```

Arguments

basename	the name of the etopo geotiff (without file extension).
path	the path to a folder containing the etopo geotiff
tmp	the path to a writeable folder
land	the logical value to return for land.

Details

Generate a land mask function based on ETOPO1 topography. The etopo geotiff is not bundled with the package and must be downloaded from https://www.ngdc.noaa.gov/mgg/global/global.html.

When the mask is intially created, a native raster (grd, gri) version of the geotiff is created in the directory `tmp`, which must be writable. This file can be deleted when the computation is finished.

The `land` argument determines whether the mask function returns `TRUE` or `FALSE` for land. The mask is constant and the `tm` argument to the mask is ignored.

Value

a logical indicating whether the point is land or sea.

gshhsMask *A land mask based on the GSHHS data set*

Description

Generate a land mask function based on the Global Self-consistent, Hierarchical, High-resolution Geography Database. The mask is provided at two (approximate) spatial resolutions: 0.1 degree and 0.05 degrees. The latter requires significantly more memory. The mask is constant and the `tm` argument to the mask is ignored.

Usage

```
gshhsMask(res = 0.1, latmin = -90, latmax = 90)

landmask_init(res = 0.1, latmin = -90, latmax = 90)
```

Arguments

`res`	numeric: the spatial resolution of the mask, in degrees (either 0.1 or 0.05)

Value

function that returns a logical indicating whether the point is at sea (TRUE) or on land (FALSE)

References

Wessel P, Smith WHF (1996) A Global Self-consistent, Hierarchical, High-resolution Shoreline Database. J. Geophys. Res. 101: 8741-8743. https://www.ngdc.noaa.gov/mgg/shorelines/gshhs.html

See Also

`surrogateAR`

Examples

```
mask  <- gshhsMask() ## initialize land mask function
mask(0, c(100, -65)) ## test point lon,lat
```

`ptt_data` *Example PTT tracking data*

Description

Three example animal tracks, from Reisinger et al. (2018)

Usage

```
data(ptt_data)
```

Format

A list containing three tracks collected using Argos PTT tags: `DMS` from a sooty shearwater, `SES` from an elephant seal, and `AFS` from an Antarctic fur seal.

Source

Reisinger RR, et al. (2018) Habitat modelling of tracking data from multiple marine top predators reveals important habitat in the Southern Indian Ocean. Diversity and Distributions.

References

https://doi.org/10.1111/ddi.12702

`randomize_track` *Randomized track*

Description

Compute surrogate track by randomizing the steps of an observed track

Usage

```
randomize_track(lonlat, rotate = c(-pi, pi), reorder = FALSE)
```

Arguments

`lonlat` a 2-column matrix or dataframe with longitude and latitude of each point

`rotate` a 2-element numeric vector giving the lower and upper limits of the random rotation to apply to the randomized track

`reorder` should the track steps be randomly reordered.

Details

Converts a track to a distance/bearing representation, and then reconstructs a new track by randomly perturbing the bearings of each increment, and otpionally, randomly reordering the increments.

Value

A dataframe with columns

lon	the longitude of the randomized track
lat	the latitude of the randomized track

surrogateAR *VAR bridge sampler*

Description

Generate new tracks from a VAR(1)

Usage

```
surrogateAR(
  model,
  xs,
  ts = seq_len(nrow(xs)),
  fixed = rep(c(TRUE, FALSE, TRUE), c(1, nrow(xs) - 2, 1)),
  point.check = function(tm, pt) TRUE,
  random.rotation = c(-pi, pi),
  partial = FALSE
)
```

Arguments

model	a model generated with surrogateARModel.
xs	the template sequence of states
ts	the times at which the track is sampled
fixed	a logical vector indicating which locations in the template path are to be held fixed.
point.check	function that accepts a state and returns boolean indicating whether the state is acceptable.
random.rotation	the upper and lower limits (radians) of the rotation applied to the VAR(1) model.
partial	if TRUE, a partial track is returned if the sampling fails.

Details

Given a fitted VAR(1) model and a template track, this function generates a new track of the same length that coincides with the template at the start point and optionally other specified points along the track.

The template track must be supplied as a matrix representing a sequence of locations where each row is a location and the columns represent longitude and latitude. The locations must be equispaced in time, and can be generated as the "p" location types from `crwPredict`.

The model object is generated by `surrogateARModel` from a fitted crawl track. This fits a VAR(1) model to the increments in longitude and latitude.

Locations from the template track can be marked as fixed with the `fixed` argument. In the current implementation the first location must always be fixed.

Additional constraints can be placed on the path by rejection sampling through the function `point.check`. This function must accept a state and return a boolean indicating whether the point is acceptable. For example, the track can be constrained to the ocean by supplying a `point.check` function that compares the state to a land mask and returns `FALSE` for states corresponding to locations that fall on land.

Value

An array of states the define the simulated path.

`surrogateARModel` *VAR(1) track model*

Description

Fit first-order vector-autoregressive model to track

Usage

```
surrogateARModel(lonlat)

surrogate_arfit(lonlat)
```

Arguments

`lonlat`	a 2-column matrix or dataframe with longitude and latitude of each point

Details

This function fits the vector AR(1) model used as the `model` argument to `surrogateAR`.

Value

An object of class "ar"

See Also

`ar`, `surrogateAR`

surrogateCrawl *Regressive bridge sampler*

Description

Generate new tracks from a Crawl model

Usage

```
surrogateCrawl(
  model,
  xs,
  ts = 1:nrow(xs),
  fixed = rep(c(TRUE, FALSE, TRUE), c(1, nrow(xs) - 2, 1)),
  point.check = function(tm, pt) TRUE,
  Verr = diag(c(1e-04, 1e-04, 0.1, 0.1)),
  partial = FALSE
)
```

Arguments

`model`	a list with the transition and covariance matrices corresponding to the fitted movement model.
`xs`	the template sequence of states
`ts`	the times at which the track is sampled
`fixed`	a logical vector indicating which locations in the template path are to be held fixed.
`point.check`	function that accepts a state and returns boolean indicating whether the state is acceptable.
`Verr`	error covariance for fixed points.
`partial`	if `TRUE`, a partial track is returned if the sampling fails.

Details

Given a fitted crawl model and a template track, this function generates a new track of the same length that coincides with the template at the start point and optionally other specified points along the track.

The template track must be supplied as a matrix representing a sequence of states where each row is a state and each column a state variable. The states must be equispaced in time, and can be generated as the "p" location types from `crwPredict`.

The crawl model object is generated by `crawlModel` from a fitted crawl model.

Locations from the template track can be marked as fixed with the `fixed` argument. In the current implementation the first location must always be fixed.

Additional constraints can be placed on the path by rejection sampling through the function `point.check`. This function must accept a state and return a boolean indicating whether the point is acceptable. For example, the track can be constrained to the ocean by supplying a `point.check` function that compares the state to a land mask and returns `FALSE` for states corresponding to locations that fall on land.

Value

An array of states the define the simulated path.

`surrogateCrawlModel` *Crawl Movement Model*

Description

Generate Transition and Covariance Matrices for a simple Crawl Model

Usage

```
surrogateCrawlModel(fit, dt)
```

Arguments

`fit`	a fitted crawl object
`dt`	the fixed time increment

Details

Generates the transition and covariance matrices of the state space model for time increment `dt`, corresponding to a fitted crawl model with no drift and no covariates.

Currently no polar adjustment is made.

The matrices are generated assuming the state variables are stored as a vector in the order longitude mu and nu then latitude mu and nu. The parametrization is defined as in the crawl source not the paper, which differs by a factor of β^2 in the definition of σ^2.

Value

A list containing

`A`	the transition matrix
`Q`	the covariance matrix
`dt`	the time increment

surrogate_arsimulate *Simulated VAR(1) tracks*

Description

Simulate track from fitted vector autoregressive model

Usage

```
surrogate_arsimulate(
  arfit,
  n,
  startlonlat,
  fixed = NULL,
  endlonlat = NULL,
  do.test.land = TRUE,
  random.rotation = c(-pi, pi),
  verbose = 0,
  return.all.points = FALSE,
  intermediate.tries = 10,
  original = FALSE
)

surrogate_arsimulate0(
  arfit,
  n,
  startlonlat,
  fixed = NULL,
  endlonlat = NULL,
  do.test.land = TRUE,
  random.rotation = c(-pi, pi),
  verbose = 0,
  return.all.points = FALSE,
  intermediate.tries = 10
)
```

Arguments

`arfit`	fitted object of class "ar" as returned by `surrogate_arfit`
`n`	number of points to simulate
`startlonlat`	2-element vector of starting longitude and latitude
`fixed`	a dataframe or matrix in which the first column is the index (from 1:n) of each fixed point, and the second and third columns give the associated longitude and latitude

`endlonlat` a 2-element vector with ending longitude and latitude. If NULL, no end constraint is imposed except for land masking (if land masking is used). This is a simple way of imposing a return-to-starting-location constraint; for more complex constraints use the `fixed` argument

`do.test.land` a logical or function. If TRUE, use the included land mask to avoid land. Alternatively, a function can be passed that returns TRUE (point is okay, not on land) or FALSE (point is on land) for a given lon,lat. Note that land masking is ignored for fixed points. Note also that it is possible to create a sitation where tracks are difficult or impossible to simulate, because a fixed point is sufficiently far onto land that the track cannot reach it.

`random.rotation` a 2-element vector giving the range of the rotation to apply to the randomized track (values in radians). use `random.rotation=NULL` for no such rotation. The angle can be restricted using `random.rotation=c(min.angle,max.angle)` - this may speed up computation by avoiding impossible angles (e.g. tracks over a land mass)

`verbose` an integer 0-3, if >0 spit out extra information which may be helpful if things don't work as expected. Larger numbers mean more output

`return.all.points` if TRUE, return points that were proposed but rejected due to land masking (may be helpful for debugging). If TRUE, the returned data.frame will have an extra column named "valid"

`intermediate.tries` when land-masking, try how many times to find a valid point at each step before giving up and starting again? Higher values may improve overall run-time, but too-high values may yield tracks that aren't a good representation of the fitted model

`original` if `TRUE`, use the original algorithm.

Details

Note that land masking uses a built-in land mask image, and it only covers the southern hemisphere. A future version will do something about this.

Value

2 or 3 column dataframe with the longitude and latitude of simulated track points (and point validity, if return.all.points is TRUE)

Index