Package 'tripEstimation'

July 22, 2025

Type Package

Title Metropolis Sampler and Supporting Functions for Estimating Animal Movement from Archival Tags and Satellite Fixes

Version 0.0-46

Imports lattice, mgcv, reproj, sp, zoo

Description Data handling and estimation functions for animal movement estimation from archival or satellite tags. Helper functions are included for making image summaries binned by time interval from Markov Chain Monte Carlo simulations.

License GPL-3

NeedsCompilation no

ByteCompile yes

URL https://github.com/Trackage/tripEstimation

BugReports https://github.com/Trackage/tripEstimation

Author Michael D. Sumner [aut, cre], Simon Wotherspoon [ctb]

Maintainer Michael D. Sumner <mdsumner@gmail.com>

Repository CRAN

Date/Publication 2023-04-21 23:40:02 UTC

Contents

`as.image.pimg` *Convert to image list*

Description

Converts Probability image (pimage) component to standard R xyz list image.

Usage

```
as.image.pimg(pimg)
combine(pimgs, subset = 1:length(pimgs))
coords.pimg(pimg)
unzipper(px)
as.local.pimg(pimg)
## S3 method for class 'pimg'
as.matrix(x, ...)
```

Arguments

`pimg`	Probability image component
`pimgs`	`pimgs`
`subset`	`subset`
`px`	`px`
`x`	`x`
`...`	`...`

Value

`as.image.pimg` returns a `image` list with vectors x,y and z matrix

`as.matrix.pimg` returns just the local matrix populated in the parent

`combine` returns the collective matrix, in `image` xyz form

`coords.pimg` returns the rectilinear coordinates of the pimg parent `unzipper` returns a `pimg.list` by combining multiple compatible ones together and resolving their temporal order

`as.local.pimg` returns the `pimg` in local form

Author(s)

Michael D. Sumner

`astro` *Calculations for position of the sun and moon*

Description

This set of functions provides simple position calculations for the sun and moon, taken from Pascal routines published in Montenbruck and Pfleger (1994, Dunlop).

These are completely independent from the (specifically optimized) solar elevation calculations available via [elevation and solar].

Usage

```
astro(lon, lat, astro.calc)

EQUHOR(DEC, TAU, PHI)

FRAC(x)

LMST(MJDay, LAMBDA)

lunar(time)

mini.sun(time)

MJD(date)

POLAR(X, Y, Z)
```

Arguments

`lon`	vector of `longitudes`
`lat`	vector of `latitudes`
`astro.calc`	list object containing `RA` right ascension
`DEC`	declination
`TAU`	`TAU`
`PHI`	`PHI`
`x`	number

`MJDay`	modified julian day
`LAMBDA`	`LAMBDA`
`time`	vector of date-times in POSIXct format
`date`	vector of date-times in POSIXct format
`X`	x-coordinate
`Y`	y-coordinate
`Z`	z-coordinate

Value

`astro` returns a list object with the components of the moon or sun's position,

`r`	rho component
`theta`	theta component - elevation
`phi`	phi component - azimuth

Warning

Some of this could be faster (particularly the use of LMST in "astro" is not precalculated)

Note

Thanks to Nick.Ellis@csiro.au for pointing out a mistake pre-0.0-27

Author(s)

Michael D. Sumner

References


```
@BOOK{,
  title = {Astronomy on the Personal Computer},
  publisher = {Springer-Verlag, Berlin},
  year = {1994},
  author = {Oliver Montenbruck and Thomas Pfleger},
  edition = {2  (translated from German by Storm Dunlop)},
}
```


See Also

See Also `elevation`

Examples

```
## the moon
tm <- Sys.time() + seq(by = 3600, length = 100)
moon <- lunar(tm)
rtp <- astro(147, -42, moon)
op <- par(mfrow = c(2,1))
plot(tm, rtp$theta, main = "lunar elevation, Hobart")
plot(tm, rtp$phi, main = "lunar azimuth, Hobart")
par(op)

## the sun
tm <- Sys.time() + seq(by = 3600, length = 100)
sun <- mini.sun(tm)
rtp <- astro(147, -42, sun)
op <- par(mfrow = c(2,1))
plot(tm, rtp$theta, main = "solar elevation, Hobart")
plot(tm, rtp$phi, main = "solar azimuth, Hobart")
par(op)
  elev.gmt <- mkElevationSeg(1, tm)
  plot(tm, rtp$theta, main = "solar elevation mini.sun versus NOAA")
  lines(tm, elev.gmt(1, 147, -42))
```

---

behav.bin *Bin MCMC chains.*

Description

Bin MCMC chains in probability image summaries.

Usage

```
behav.bin(z, pimgs, weights = NULL)
bin.pimg(pimg, xy, w = 1)
chunk.bin(filename, pimgs, weights = NULL, chunk = 2000, proj = NULL)
```

Arguments

z	z
pimgs	pimgs
weights	weights
pimg	pimg
xy	xy
w	w
filename	filename
chunk	chunk
proj	proj

Value

`behav.bin` returns a `pimg.list`

`bin.pimg` and `chunk.bin` provide work flow for `behav.bin`, to do the local binning and control the overal job

---

`bits` *Set and get bits from binary masks.*

Description

Utility functions to access bits from numeric values, for the efficient storage of spatial masks.

Usage

```
bits(object, bit)

bits(object, bit) <-  value
```

Arguments

`object`	a numeric value
`bit`	the desired bit
`value`	logical value to set bit to

Details

R uses 32-bit integers, so we can (easily) access 31 binary matrices in each numeric matrix. This is very useful for storing long time-series of spatial masks, required for track-location estimation from archival tags.

Value

A numeric object with the given bit set, or a logical value designating the status of the given bit.

Note

The 32nd bit is harder to access, so we ignore it.

Author(s)

Michael D. Sumner

See Also

See Also `get.mask` for a higher level access of a mask object

Examples

```
a <- 1L
bits(a, 0)  ## 1
bits(a, 2) <- 1
a   # 5
```

`chain.read` *Manage MCMC cache.*

Description

These functions read and write to cache files for storing long MCMC outputs from model functions, such as `solar.model` or `satellite.model`.

Usage

```
chain.read(filename)
chain.dim(filename)
chain.write(filename, A, append = FALSE)
```

Arguments

`filename`	cache file for model chain
`A`	chain array
`append`	append to existing file or overwrite?

Value

`chain.read` returns the actual array of MCMC samples from an archived file

`chain.dim` reports the dimensions of the archived file

`chain.write` writes an array of MCMC samples to an archive file

Author(s)

Michael D. Sumner and Simon Wotherspoon

See Also

`pimg.list`

elevation *Calculate elevation of astronomical objects*

Description

Function to calculate elevation.

Usage

```
elevation(lon, lat, sun)
```

Arguments

lon	vector of longitude values
lat	vector of latitude values
sun	pre-stored values as returned by solar or lunar

Value

elevation returns a numeric vector of solar (or lunar) elevation as degrees above or below the horizone

Author(s)

Michael D. Sumner

References

https://gml.noaa.gov/grad/solcalc/azel.html

get.mask *Create, access and manipulate spatial masks*

Description

Spatial masks are stored using the xyz-list structure used by image or as a series of masks stored as bits in the z-component as matrix or array object. get.mask is used to extract a specific mask from the binary storage, and mkSmall can be used to quickly down-sample an existing mask or image.

Usage

```
get.mask(masks, k)

mkSmall(lst, thin = 10)

set.mask(object, segment) <-  value

mkMaskObject(xs, ys, nsegs)
```

Arguments

masks	A list object with components x, y, and z containing spatial masks
k	specifies the k-th mask
lst	an xyz-list structure with z containing either a matrix or array
thin	integer factor to down-sample grid
object	array Mask object
segment	segment number to be modified in the mask
value	individual mask to be set
xs	x coordinates of mask cells
ys	y coordinates of mask cells
nsegs	number of segments to be represented

Value

matrix of type logical

Author(s)

Michael D. Sumner

See Also

mkLookup for the use of these masks to query individual locations and locations measured over time. See bits for the underlying mechanism to set and get mask bits.

For the use of the xyz-list structure see image.

Examples

```
data(volcano)
d <- list(x = seq(-10, 10, length = nrow(volcano)),
          y = seq(-5, 5, length = ncol(volcano)),
          z = array(0L, c(nrow(volcano), ncol(volcano), 2)) )
mv <- min(volcano)

for (i in 0:61) {
  blk <- (i %/% 31) + 1
```

```
  bit <- (i - 1) %% 31
  bits(d$z[,,blk], bit) <- volcano > (mv + i*1.6 )
}
for (i in 0:61) image(get.mask(d, i))

## an object with 62 masks is only twice the size of the source data
object.size(d) / object.size(volcano)

## plot a smaller version
image(get.mask(d, 20), 5)
```

---

initialize.x *Diagnose and initialize light level estimation.*

Description

Primarily for the purposes of initializing the estimation, these functions can also be used for diagnostic purposes. `position.logp` produces grids of simplistic position likelihood for each twilight and uses those to initialize positions for running estimations.

Usage

```
position.logp(model, x1, x2, xrest = NULL, subset = 1:model$n,
initialize.x = TRUE, start = NULL, end = NULL, prob = 0.8, winoffset = 5)

initialize.x(model, x1, x2, xrest = NULL)

light.quantile(model, chain, day, seg, probl = c(0.025, 0.5, 0.975))

show.segment(model, chain, segment, day, light, k, n = 50, ...)
```

Arguments

`model`	estimation model object
`x1`	vector of x-coordinates defining the prior grid
`x2`	vector of y-coordinates defining the prior grid
`xrest`	value for remaining parameters - default is light attenuation
`subset`	evaluate subset of segments - default uses all
`initialize.x`	logical - create initial points for x?
`prob`	probability - threshold to apply to overlapping quantiles, defaults to 0.8
`winoffset`	an odd-numbered window size to use when intersecting subseqent segments - defaults to 5
`chain`	chain object from estimation

day	POSIXct vector of date-times
seg	desired segment
probl	probability level for quantile
start	known position of release
end	known position of recapture
segment	vector of segment data
light	vector of light data
k	desired segment to show
n	length of vector to evaluate
...	additional arguments to be passed to plot

Details

The primary function here is `position.logp`, for initializing the estimation for `solar.model` and `metropolis0`.

Value

`initialize.x` returns a matrix with 3 columns, lon,lat,attenuation `position.logp` returns a list with model running components `show.segment` is used for its side effect, a plot of light level for a twilight segment `light.quantile` returns a numeric vector

Author(s)

Michael D. Sumner

---

julday *Julian day and Julian century calculations from date-time values*

Description

Date values required by `solar`.

Usage

```
julday(tm)

julcent(time)
```

Arguments

tm	vector of date-times
time	vector of date-times

Value

return numeric values

Author(s)

Michael D. Sumner

References

https://gml.noaa.gov/grad/solcalc/azel.html

`metropolis` *Metropolis-Hastings sampler for location estimation for archival and satellite tag*

Description

These functions provide a direct implementation of the Metropolis-Hastings algorithm, for calculating marginal posterior (locations and full-track estimates) properties using Markov Chain Monte Carlo. The sampler is written completely in R, vectorized to be as fast as possible. The sampler can include likelihood functions for large data records (including light and water temperature), as well as *mask* functions for simpler rejection sources. Behavioural constraints are implemented using a red/black update, so that location estimates *X* and *Z* may be estimated in an efficient manner. The parameter estimates may be cached and later queried arbitrarily.

Usage

```
metropolis(model, iters = 1000, thin = 10, start.x = NULL, start.z = NULL)

metropolis0(model, iters = 1000, thin = 10, start.x = NULL, start.z =
NULL)
```

Arguments

`model`	model for estimation, such as one created by `solar.model`
`iters`	number of iterations to run
`thin`	number of iterations to thin by
`start.x`	starting points for the primary locations
`start.z`	starting points for the intermediate locations (midpoints between the `start.x` points is a good first guess

Details

metropolis0 is a slightly different version of metropolis that enables an initialization step, required to find parameter estimates that are consistent with any masks used. It is difficult to make this step more elegant, and so we live with the two versions.

In terms of the estimates, X's have m records with n parameters, where m is the number of data records in time (twilights for archival tags, Argos estimates for satellite tags) and n is at least x-coordinate, y-coordinate and maybe k-attenuation for light. Z's have m-1 records with 2 parameters for 'x' and 'y' (which are usually Longitude and Latitude). These parameters may be increased or changed, they are tied only to the likelihood functions used, not the sampler itself. Also, coordinate transformations may be used inside the model and likelihood functions, in order to use an appropriate map projection. Solar calculations rely on lon/lat and so this step does slow down light level geo-location.

Value

A MCM *Chain* stored as a list containing

model	The model object used by the sampler
x	The last iters X-samples accepted, stored as an c(m, n, iters) array
z	The last iters Z-samples accepted, stored as an c(m - 1, 2, iters)
last.x	The last accepted X-sample, stored as a c(m, n) matrix
last.z	The last accepted Z-sample, stored as a c(m, 2) matrix

Author(s)

Michael D. Sumner and Simon Wotherspoon

References

Sumner, Wotherspoon and Hindell (2009). Bayesian Estimation of Animal Movement from Archival and Satellite Tags, PLoS ONE. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0007324

See Also

solar.model, satellite.model

---

mkCalibration *Create calibration of solar elevation to measured light level.*

Description

Using a set of light level data from a known location create a calibration function to return the expected light level given solar elevation.

Usage

```
mkCalibration(x, known = NULL, elim = c(-36, 12), choose = TRUE)
```

Arguments

`x`	a data frame containing at least `gmt` and `light`
`known`	a known position - as a 2-element `c(x, y)` coordinate
`elim`	a 2-element vector of the range of solar elevation to define
`choose`	logical - choose segments from a plot or use all the data?

Details

It is assumed that the data frame `x` has columns "gmt" with POSIXct date-times and "light" with numeric light level data.

Value

A function, defined by `approxfun`.

Author(s)

Michael D. Sumner

See Also

`approxfun`

`mkLookup` *Create a lookup function to query locations against spatial masks*

Description

Simple pixel spacing is used to overlay point locations on a spatial grid, or a series of grids.

Usage

```
mkLookup(x, by.segment = TRUE)
```

Arguments

`x`	an xyz-list with matrix or array of masks
`by.segment`	logical - is the mask to be queried separately for each time step?

Value

A function, with one argument - a matrix of points - that returns a logical vector indicating the overlay of each point against the masks.

Note

Very little error checking is done.

Author(s)

Michael D. Sumner

See Also

`get.mask` and related examples for creating and using masks.

See `over` for more general capabilities for overlays.

`norm.proposal` *Manage proposal functions tune variance for metropolis sampler*

Description

Generate new proposals for the x from the current. Generates all x at once.

Usage

```
norm.proposal(m, n, sigma)

mvnorm.proposal(m, n, Sigma)

bmvnorm.proposal(m, n, Sigma)
```

Arguments

`m`	number of records
`n`	number of parameters
`sigma`	variance
`Sigma`	variance

Details

norm.proposal - Independent Normal proposal - every component is independent, with variances of individual components determined by sigma. The recycling rule applies to sigma, so sigma may be a scalar, an m vector or a m by n matrix.

mvnorm.proposal - Multivariate Normal proposal - all components of all points are correlated. In this case Sigma is the joint covariance of the m*n components of the proposal points.

bmvnorm.proposal - Block Multivariate Normal proposal - components of points are correlated, but points are independent. Here Sigma is an array of m covariance matrices that determine the covariance of the m proposal points.

Value

An list object with get, set and tune functions to manage the state of the proposals.

`proposal`	propose new set of parameters from last
`get`	get variance values
`set`	set variance values
`tune`	tune the variance for proposal functions

Author(s)

Simon Wotherspoon

`old.metropolis` *Older versions of solar location estimation*

Description

Some deprecated functions, originally used purely for light level estimation before the sampling algorithm was generalized for satellite models as well.

Usage

```
mkElevationSeg(segments, day)

mkNLPosterior(segments, day, light, calib)

old.dist.gc(x1, x2 = NULL)

old.find.init(mask, nseg, nlpost, pars = c("Lon", "Lat", "k"))

old.metropolis(nlpost, lookup, p0, cov0, start, end, iter = 1000, step = 100)

old.mkLookup(x, binArray = TRUE)

k.prior(seg, ps)
```

Arguments

`segments`	vector identifying the segment of each time and light value
`day`	date-time values in POSIXct
`light`	vector of light data
`calib`	calibration function for light levels
`x1`	matrix of track locations
`x2`	matrix of track locations (optional second part)

`mask`	image object of masked areas
`nseg`	number of (twilight) segments
`nlpost`	negative log posterior function
`pars`	names of parameters
`lookup`	lookup function for masked areas
`p0`	initial locations for sampler
`cov0`	covariance matrix for sampler
`start`	known start parameters
`end`	known end parameters
`iter`	number of iterations
`step`	number of thinning iterations per `iter`
`x`	image-like object of matrix or array of binary masks
`binArray`	logical: are the masks compressed into bits?
`seg`	segment
`ps`	light attenuation value

Details

These functions are included for legacy purposes, this was the original implementation.

Value

If it is a LIST, use

Author(s)

Michael D. Sumner

See Also

Please use the more up to date function `metropolis`, with the models such as `solar.model` or `satellite.model`.

`pick` *Choose twilight segments interactively from light data.*

Description

`pick` plots up series of light data agains record ID, allowing the user to click on the beginnings and ends of twilight in sequence. `picksegs` generates a vector of segment IDs for each record.

Usage

```
pick(id, val, nsee = 10000)

picksegs(twind, n)
```

Arguments

`id`	index vector to identify records
`val`	sequence of data (light levels) to choose segments from
`nsee`	number of points to plot per screen
`twind`	vector of index pairs generated by `pick`
`n`	Number of segments values required - length of record

Value

`pick` returns a vector where each value (obtained using `locator` is the x coordinate for the begin or end of a twilight.

`picksegs` uses these paired indexes to return a vector of segment IDs, with NAs for non-twilight periods.

Warning

Segments are expected to be chosen as non-overlapping.

Note

It seems best to choose more of the light data than less, using the `ekstrom` keyword to `solar.model` we can limit the solar elevation used.

Author(s)

Michael D. Sumner

Examples

```
d <- sin(seq(0, 10, by = 0.01))
id <- 1:length(d)
## choose a series of start-begin pairs
if (interactive()) {
pk <- pick(id, d, 1000)
## your start/ends should be marked as blue versus red
plot(id, d, col = c("red", "blue")[is.na(picksegs(pk, 1000))+1])
}
```

pimg.list — *Create a collection of probability images, for MCMC binning.*

Description

Pimage lists.

Usage

```
pimg(xmin, xmax, xn, ymin, ymax, yn)
pimg.list(times, xlim, ylim, img.dim, Z = TRUE)
```

Arguments

xmin	xmin
xmax	xmax
xn	xn
ymin	ymin
ymax	ymax
yn	yn
times	times
xlim	xlim
ylim	ylim
img.dim	img.dim
Z	Z

Value

returns a Pimage list

satellite.model — *Function to create a satellite model object for metropolis location sampler*

Description

A model to manage likelihood functions, environmental masks and behavioural likelihood functions for pre-derived satellite locations. There are some options for configuration, but this may be considered a template for any given model. The model *function* exists simply to make the object construction simple.

Arguments

`day`	vector of date-times for each light level
`X`	matrix of pre-derived satellite locations
`proposal.x`	function from object managing X proposals
`proposal.z`	function from object managing Z proposals
`mask.x`	lookup function for X's against masks
`mask.z`	lookup function for Z's against masks
`fix.release`	logical - is the release point known?
`fix.recapture`	logical - is the recapture point known?
`start.x`	starting positions for the primary locations, see `position.logp`
`start.z`	starting positions for the intermediat locations.
`posn.sigma`	variance for locations
`behav.dist`	distribution to use for behavioural constraint
`behav.mean`	mean to use for behavioural distribution
`behav.sd`	variance for behavioural distribution
`proj.string`	PROJ.4 string for coordinate system used

Details

`posn.sigma` may be a single value for all estimates, or a vector of values for each position estimate.

Transformation of coordinates is supported via a simple function that only performs coordinate transforms if `proj.string` is not longlat.

Value

See `solar.model` for some related detail.

Note

These are simple wrapper functions to create the desired model for use in `metropolis`. These models are structurally very simple and may be easily edited as required.

Author(s)

Michael D. Sumner

References

Sumner, Wotherspoon and Hindell (2009). Bayesian Estimation of Animal Movement from Archival and Satellite Tags, PLoS ONE. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0007324

See Also

See also `solar.model` for the counterpart model for estimating positions for light tags.

`solar` *Calculate solar postion parameters*

Description

Pre-calculates astronomical solar position components for Earth-location sampling functions.

Usage

```
solar(day)
```

Arguments

`day`	vector of date-time values

Value

A list of the following values for each input time:

`solarTime`	solar time
`sinSolarDec`	sine solar declination
`cosSolarDec`	cosine solar declination

Note

No account is made for horizon refraction, but this was available in the original (Javascript) code.

Author(s)

Michael D. Sumner

References

https://gml.noaa.gov/grad/solcalc/azel.html

solar.model *Function to create a solar model object for metropolis location sampler*

Description

A solar model to manage likelihood functions, environmental masks and behavioural likelihood functions. There are several options for configuring the model, and this may be considered a template for any given model. The model *function* exists simply to make the object construction simple.

Usage

```
solar.model(segments, day, light,
proposal.x, proposal.z, mask.x, mask.z,
fix.release = TRUE, fix.recapture = TRUE,
calibration,
light.sigma = 7, k.sigma = 10,
behav = "speed", behav.dist = "gamma",
behav.mean, behav.sd,
proj.string = "+proj=longlat",
   ekstrom = c(-5, 3, light.sigma),
         ekstrom.limit = "light")
```

Arguments

`segments`	vector identifying twilight segment
`day`	vector of date-times for each light level
`light`	vector of light levels
`proposal.x`	function from object managing X proposals
`proposal.z`	function from object managing Z proposals
`mask.x`	lookup function for X's against masks
`mask.z`	lookup function for Z's against masks
`fix.release`	logical - is the release point known?
`fix.recapture`	logical - is the recapture point known?
`calibration`	calibration function for predicted light level for solar elevation
`light.sigma`	variance for light data
`k.sigma`	variance for light attenuation
`behav`	model distributions to be used for behaviour - defaults to "speed"
`behav.dist`	distribution to be used for behaviour
`behav.mean`	mean for behavioural distribution
`behav.sd`	variance for behavioural distribution
`proj.string`	PROJ.4 string for coordinate system used
`ekstrom`	parameters to use for ekstrom limit - min elevation, max elevation, sigma for outside that range
`ekstrom.limit`	mode of ekstrom limit to impose - defaults to "light"

Details

The vectors of `segments`, `day` and `light` are expected to be of the same length.

Fixed recapture and release points are treated specially for ease of sampling, but the sampling is written to be general for any fixed locations.

Behavioural models may be specified either as lognormal or log-gamma. By editing the function created as `logp.behavioural` this may be specified differently.

Transformation of coordinates is supported via a simple function that only performs coordinate transforms if `proj.string` is not longlat.

Value

proposal.x(x) - generates new proposals for the x from the current x. Generates all x at once.

proposal.z(z) - generates new proposals for the x from the current z. Generates all z at once.

mask.x(x) - mask function for the x. Simultaneously tests all x and returns a vector of booleans indicating which are acceptable.

mask.z(z) - mask function for the z. Simultaneously tests all z and returns a vector of booleans indicating which are acceptable.

logp.position(x) - Given the set of x, returns a vector that gives the contribution each x make to the log posterior based on position alone.

logp.behavourial(k,xa,z,xb) - Computes the contribution to the log posterior from the behavioural model on a subset of segments that make up the path. Here k is a vector of the segment numbers, where the segments pass from xa to z to xb, and the function returns the contribution to the log posterior from each segment. This is the only function expected to work with only a subset of the x and z.

start.x - suggested starting points for the x

start.z - suggested starting points for the z

The only function that must operate on a subset of the x/z is logp.behavourial. All the other functions operate on all x or z simultaneously, simplifying the implementation for the user.

Note that x can consist of several parameters, not just the locations, but we assume the first two components of each x specify the location. For example, in the light level models each x is (lon,lat,k) where k is the attenuation of the light level.

Some details of this implementation are not as nice as they could be. First, it would be better if did not calculate the contributions to the posterior for points the mask rejects. Also, it may be better to separate the specification of the functions that generate proposals from the other functions, so that we can tune the proposal distributions without re-generating the whole model specification.

Author(s)

Simon Wotherspoon and Michael Sumner

Index