The tripEstimation Package

May 1, 2007

Type Package

Title Metropolis sampler and supporting functions for estimating animal movement from archival tags and satellite fixes

Version 0.0-23

Depends zoo, mgcv, lattice

Suggests rgdal

Author Michael Sumner and Simon Wotherspoon

Maintainer Michael Sumner <mdsumner@utas.edu.au>

Description Data handling and estimation functions for animal movement estimation by light level and satellite data.

License GPL2


R topics documented:

astro .................................................. 2
bits ................................................... 4
elevation .............................................. 5
get.mask .............................................. 6
initialize.x ........................................ 8
julday ............................................... 10
k.prior ............................................... 11
metropolis .......................................... 12
mkCalibration ...................................... 13
mkLookup .......................................... 14
norm.proposal ...................................... 15
old.metropolis .................................. 17
pick ............................................... 18
0.1 Warning ....................................... 19
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
```r
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

Details

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

I don’t get perfect matchup with the NOAA routines for solar elevation. This may be the limitations
of the (mini-versions of the) algorithms, as they are good at near-horizon (for rising and setting).
Some of this could be faster (particularly the use of LMST in "astro" is not precalculated)

Note

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

```r
## 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)
## Not run:
elev.gmt <- mkElevationSeg(1, tm)
plot(tm, rtp$theta, main = "solar elevation mini.sun versus NOAA")
lines(tm, elev.gmt(1, 147, -42))
## End(Not run)
```

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

```r
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

**References**

**See Also**

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

**Examples**

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

---

**Description**

A concise (1-5 lines) description of the dataset.

**Usage**

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

**Arguments**

- `lon` vector of longitude values
- `lat` vector of latitude values
- `sun` solar calculations as returned by `solar`

**Details**

get.mask

Value

Note

Author(s)

Michael D. Sumner

References

[NOAA webpage]

Examples

---

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. The two "make" functions are included as examples of creating such a mask from a local database.

Usage

```r
get.mask(masks, k)

mkSmall(lst, thin = 10)

mkDepthMask(depth.mins, depthdata = NULL)

mkTempMask(times, temp.ranges, xlim = c(0.5, 359.5), ylim = c(-89.5, 89.5), sstdata = NULL)
```
get.mask

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
- **depth.mins**
  depth values to compare to database of topography
- **depthdata**
  xyz-list structure of topography
- **times**
  date-times at which temperatures are to be compared
- **temp.ranges**
  matrix of two columns - min/max of temperature for given period
- **xlim**
  x-range to query from temperature database
- **ylim**
  y-range to query from temperature database
- **sstdata**
  character describing sst database
- **interp**
  logical - should gappy data be interpolated?

Details

`mkDepthMask` and `mkTempMask` are included to show operational examples used by our group. Future publications will refer to this code, or to similar functions.

Value

Note

Author(s)

Michael D. Sumner

References

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(0, 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)

## Not run:
## plot a smaller version
image(get.mask(d, 20), 5)

## pretend we have only one masks
lookup <- mkLookup(get.mask(d, 30), by.segment = FALSE)

## interactive to show use of lookup function
image(get.mask(d, 30), main = "Click on the red (FALSE) and cream (TRUE) areas")
for (i in 1:10) {x <- matrix(unlist(locator(1)), ncol = 2);text(x[,1], x[,2], lookup(x))
## End(Not run)

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 both produces diagnostic images for each twilight and uses those to initialize position.

Usage

position.logp(model, x1, x2, xrest = NULL, subset = 1:model$n, initialize.x = TRUE, initialize.x(model, x1, x2, xrest = NULL)

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

dshow.segment(model, chain, segment, day, light, k, n = 50, ...)
**initialize.x**

**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?
- `chain`: chain object from estimation
- `day`: POSIXct vector of date-times
- `seg`: desired segment
- `prob`: 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**

**Note**

**Author(s)**

Michael D. Sumner

**References**

**See Also**

**Examples**
Description
[definition and rationale of Julian]

Usage
julday(tm)
julcent(time)

Arguments

<table>
<thead>
<tr>
<th>tm</th>
<th>vector of date-times</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>vector of date-times</td>
</tr>
</tbody>
</table>

Details

Value

Note

Author(s)

Michael D. Sumner

References

[Astro and NOAA]

See Also

Examples
Description

This function is used by the metropolis sampler

Usage

k.prior(seg, ps)

Arguments

seg
ps

Value

Note

Author(s)

References

See Also

Examples
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

```r
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` Describe model here
- `iters` Describe iters here
- `thin` Describe thin here
- `start.x` Describe start.x here
- `start.z` Describe start.z here

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.
mkCalibration

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 a `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

Note

Author(s)

Michael D. Sumner

References

[Ekstrom, Hill, Gilks, etc.]

See Also

- `solar.model`, `satellite.model`

Examples

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

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

```r
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`.

**Note**

**Author(s)**

Michael D. Sumner

**References**

**See Also**

`approxfun`

**Examples**

```r
mkLookup
```

---

**Description**

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

**Usage**

```r
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?

**Details**
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

References

See Also
get.mask and related examples for creating and using masks.
See overlay for more general capabilities for overlays.

Examples

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>number of records</td>
</tr>
<tr>
<td>n</td>
<td>number of parameters</td>
</tr>
<tr>
<td>sigma</td>
<td>variance</td>
</tr>
<tr>
<td>Sigma</td>
<td>variance</td>
</tr>
</tbody>
</table>
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

Note

The mv and bmv versions are included for archival purposes - they are not tested.

Author(s)

Simon Wotherspoon

References

See Also

Examples
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)

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)

Arguments

segments Describe segments here
day Describe day here
light vector of light data
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?
Details

These functions are included for archival purposes and are not fully supported.

Value

If it is a LIST, use

comp1 Description of 'comp1'
comp2 Description of 'comp2'

Author(s)

Michael D. Sumner

References

See Also

Please use the supported function metropolis, with the solar.model or satellite.model.

Examples

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

Description

pick plots up series of light data against 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

```r
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
0.1. WARNING

Details
Choosing twilight segments interactively seems far easier than trying to automate it. Mark Hindell makes the point that you get to see the data in detail, which is good.

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.

0.1 Warning
Segments are expected to be chosen as non-overlapping.

Author(s)
Michael D. Sumner

References

See Also

Examples
```r
## Not run:

d <- sin(seq(0, 10, by = 0.01))
id <- 1:length(d)
## choose a series of start-begin pairs
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])

## End(Not run)
```
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  Describe start.x here
start.z  Describe start.z here
posn.sigma  variance for locations
speed.dist  distribution to use for behavioural constraint
speed.mean  mean to use for behavioural distribution
speed.sd  variance for behavioural distribution

Details

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

Value

Note

This function is not completely consistent with its solar counterpart solar.model. These are working functions that are in constant development, but they are structurally simple and may be easily edited as required.
solar

Author(s)

Michael D. Sumner

References

[Argos manual]

See Also

See also solar.model for the counterpart model for solar problems

Examples

```
solar

Calculate solar position parameters
```

Description

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

Usage

```
solar(day)
```

Arguments

day

vector of date-time values

Details

Value

A list of the following values for each input time:

- solarTime
- sinSolarDec
- cosSolarDec

Note

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

References
[NOAA website]

See Also

Examples

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
**solar.model**

- **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 `create` 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`. See `rgdal` for the underlying functionality.

**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.behavioural(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.behavioural. 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.

Note

Author(s)

Simon Wotherspoon and Michael Sumner

References

See Also

Examples
## Index

### *Topic dplot*
- initialize.x, 8
- mkCalibration, 13
- pick, 18

### *Topic manip*
- astro, 1
- bits, 4
- elevation, 5
- get.mask, 6
- initialize.x, 8
- julday, 9
- k.prior, 10
- metropolis, 11
- mkCalibration, 13
- mkLookup, 14
- norm.proposal, 15
- pick, 18
- satellite.model, 19
- solar, 21

### *Topic misc*
- old.metropolis, 16

### *Topic models*
- solar.model, 22

<table>
<thead>
<tr>
<th>approxfun</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>astro</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bits</th>
<th>4, 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits&lt;- (bits)</td>
<td>4</td>
</tr>
<tr>
<td>bmvnorm.proposal(norm.proposal),</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>elevation</th>
<th>3, 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQUHOR(astro)</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FRAC(astro)</th>
<th>1</th>
</tr>
</thead>
</table>

| get.mask | 4, 6, 14 |

<table>
<thead>
<tr>
<th>image</th>
<th>6, 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize.x</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>julcent(julday),</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>julday,</td>
<td>9</td>
</tr>
<tr>
<td>k.prior,</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>light.quantile(initialize.x),</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMST(astro),</td>
<td>1</td>
</tr>
<tr>
<td>locator,</td>
<td>18</td>
</tr>
<tr>
<td>lunar(astro),</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>metropolis,</th>
<th>11, 12, 17</th>
</tr>
</thead>
<tbody>
<tr>
<td>metropolis0,</td>
<td>8, 12</td>
</tr>
<tr>
<td>metropolis0(metropolis),</td>
<td>11</td>
</tr>
<tr>
<td>mini.sun(astro),</td>
<td>1</td>
</tr>
<tr>
<td>MJD(astro),</td>
<td>1</td>
</tr>
<tr>
<td>mkCalibration,</td>
<td>13</td>
</tr>
<tr>
<td>mkDepthMask(get.mask),</td>
<td>6</td>
</tr>
<tr>
<td>mkElevationSeg(old.metropolis),</td>
<td>16</td>
</tr>
<tr>
<td>mkLookup,</td>
<td>7, 14</td>
</tr>
<tr>
<td>mkNLPosterior(old.metropolis),</td>
<td>16</td>
</tr>
<tr>
<td>mkSmall(get.mask),</td>
<td>6</td>
</tr>
<tr>
<td>mkTempMask(get.mask),</td>
<td>6</td>
</tr>
<tr>
<td>mvmnprop.proposal(norm.proposal),</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>norm.proposal,</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>old.dist.gc(old.metropolis),</td>
<td>16</td>
</tr>
<tr>
<td>old.find.init(old.metropolis),</td>
<td>16</td>
</tr>
<tr>
<td>old.metropolis,</td>
<td>16</td>
</tr>
<tr>
<td>old.mkLookup(old.metropolis),</td>
<td>16</td>
</tr>
<tr>
<td>overlay,</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pick,</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>picksegs(pick),</td>
<td>18</td>
</tr>
<tr>
<td>POLAR(astro),</td>
<td>1</td>
</tr>
<tr>
<td>position.logp(initialize.x),</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>satellite.model,</th>
<th>12, 17, 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>show.segment(initialize.x),</td>
<td>8</td>
</tr>
</tbody>
</table>
solar, 5, 21
solar.model, 8, 12, 17, 18, 20, 22