Package ‘marcher’

April 12, 2017

Type Package
Title Migration and Range Change Estimation in R
Version 0.0-2
Description A set of tools for likelihood-based estimation, model selection and testing of two- and three-range shift and migration models for animal movement data as described in Gurarie et al. (2017) <doi: 10.1111/1365-2656.12674>. Provided movement data (X, Y and Time), including irregularly sampled data, functions estimate the time, duration and location of one or two range shifts, as well as the ranging area and autocorrelation structure of the movement. Tests assess, for example, whether the shift was "significant", and whether a two-shift migration was a true return migration.
License GPL-2
Depends R (>= 3.3.0)
Imports stats, Matrix, graphics, grDevices, plyr, mvtnorm, RColorBrewer, minpack.lm, zoo, numDeriv, magrittr, scales
VignetteBuilder knitr
BugReports https://github.com/EliGurarie/marcher/issues
RoxygenNote 6.0.1
Suggests knitr, rmarkdown, lubridate
NeedsCompilation no
Author Eliezer Gurarie [aut, cre], Farid Cheraghi [aut]
Maintainer Eliezer Gurarie <egurarie@umd.edu>
Repository CRAN
Date/Publication 2017-04-12 14:26:18 UTC

R topics documented:
marcher-package .................................................. 2
estimate_shift ..................................................... 3
fitNSD ............................................................. 5
Description

A collection of functions for performing a migration and range change analysis (MRSA) as described in by Gurarie et al. (2017). The key features are estimation of precise times, distances, and locations of a one or two step range shift in movement data.

Details

Some key functions for using marcher are:

1. `estimate_shift` Estimate a range shift process.
2. `simulate_shift` Simulate a range shift process.
3. `plot.shiftfit` Visualize a range shift process.
4. `test_rangeshift` Test whether a range shift occurred.
5. `test_return` Test whether a migration was a return migration.
6. `test_stopover` Test whether a stopover occurred during a migration.

Several simulated datasets are in the `SimulatedTracks` data object.

One roe deer (`Capreolus capreolus`) track is in the `Michela` object.

See the respective help files and vignette("marcher") for more details and examples.

Author(s)

Maintainer: Eliezer Gurarie <egurarie@umd.edu>

Authors:

- Farid Cheraghi
References


See Also

Useful links:
- Report bugs at https://github.com/EliGurarie/marcher/issues

---

estimate_shift

Estimating range shifts

Description

Estimation and helper functions for nls fit of migration model

Usage

```r
estimate_shift(T, X, Y, n.clust = 2, p.m0 = NULL, dt0 = min(5, diff(range(T))/20), method = c("ar", "like")[1], CI = TRUE, nboot = 100, model = NULL, area.direct = NULL)
```

Arguments

- `T`: time
- `X`: x coordinate
- `Y`: y coordinate
- `n.clust`: the number of ranges to estimate. Two is relatively easy and robust, and three works fairly well (with good initial guesses). More can be prohibitively slow.
- `p.m0`: initial parameter guesses - a named vector with (e.g.) elements x1, x2, y1, y2, t1, dt. It helps if this is close - the output of `quickfit` can be helpful, as can plotting the curve and using `locator`. If left as NULL, the function will make some guesses for you - starting with `quickfit`.
- `dt0`: initial guess for duration of migration
- `method`: one of ‘ar’ or ‘like’ (case insensitive), whether or not to use the AR equivalence method (faster, needs regular data - with some tolerance for gaps) or Likelihood method, which is slower but robust for irregular data.
- `CI`: whether or not to estimate confidence intervals
- `nboot`: number of bootstraps
- `model`: one of "MWN", "MOU" or "MOUF" (case insensitive). By default, the algorithm selects the best one according to AIC using the `selectModel` function.
- `area.direct`: passed as direct argument to getArea
Details

This algorithm minimizes the square of the distance of the locations from a double-headed hockey-stick curve, then estimates the times scale using the ARMA/AR models. Confidence intervals are obtained by bootstrapping the data and reestimating. See example and vignette for implementation.

Value

a list with the following elements

- `tLxLy` Longitude coordinate with NA at prediction times
- `p.hat` Point estimates of parameters
- `p.CI` Data frame of parameter estimates with (approximate) confidence intervals.
- `model` One of "wn", "ou" or "ouf" - the selected model for the residuals.
- `hessian` The hessian of the mean parameters.

Examples

```r
# load simulated tracks
data(SimulatedTracks)

# white noise fit
MWN.fit <- with(MWN.sim, estimate_shift(T=T, X=X, Y=Y))
summary(MWN.fit)
plot(MWN.fit)

if(interactive()){
  # OUF fit
  MOUF.fit <- with(MOUF.sim.random,
                   estimate_shift(T=T, X=X, Y=Y,
                                   model = "ouf",
                                   method = "like"))
  summary(MOUF.fit)
  plot(MOUF.fit)

  # Three range fit:
  # it is helpful to have some initial values for these parameters
  # because the automated quickfit() method is unreliable for three ranges
  # in the example, we set a seed that seems to work
  # set.seed(1976)

  MOU.3range.fit <- with(MOU.3range,
                         estimate_shift(T=T, X=X, Y=Y,
                                        model = "ou",
                                        method = "ar",
                                        n.clust = 3))
  summary(MOU.3range.fit)
  plot(MOU.3range.fit)
}
```
fitNSD

Test range shift using net-squared displacement

Description

Test range shift using net-squared displacement

Usage

```
fitNSD(T, X, Y, plotme = FALSE, setpar = TRUE, ...)
```

Arguments

- **T**: time
- **X**: x coordinate
- **Y**: y coordinate
- **plotme**: whether or not to plot the result
- **setpar**: whether or not to run `par(mfrow = c(1,2))` before plotting
- **...**: additional parameters to pass to `plot`

Details

The test below assumes that the net squared displacement (NSD) for a migrating organism is well characterized by the logistic formula: \( E(\text{NSD}(t)) = \frac{a}{1 + \exp \left(\frac{b-t}{c}\right)} \) as described in border=ger and Fryxell (2012). In practice, the square root of the NSD, i.e., the linear displacement, is fitted to the square root of the formula assuming Gaussian residuals with constant variance 's'. A likelihood ratio test against a null model of no-dispersal is provided at a 95% significance level.

Value

A list with a vector of four parameter estimates, and a vector with test statistics (likelihood, AIC and p.values)

Examples

```
# simulate and compare two range shifts
A <- 20
T <- 1:100
tau <- c(tau.z = 2, tau.v = 0)

# large dispersal
Mu <- getMu(T, c(x1 = 0, y1 = 0, x2 = 4, y2 = 4, t1 = 40, dt = 20))
XY.sim <- simulate_shift(T, tau = tau, Mu, A=A)
with(XY.sim, scan_track(time = T, x = X, y = Y))
with(XY.sim, fitNSD(T, X, Y, plotme=TRUE))
```
# no dispersal
Mu <- getMu(T, c(x1 = 0, y1 = 0, x2 = 0, y2 = 0, t1 = 40, dt = 20))
XY.sim <- simulate_shift(T, tau = tau, Mu, A=A)
with(XY.sim, scan_track(time = T, x = X, y = Y))
with(XY.sim, fitNSD(T,X,Y, plotme=TRUE))

getArea

Compute area

Description

Compute predicted area at given alpha level (e.g. 50% or 90%) of a migration model fit

Usage

getArea(p, T, X, Y, alpha = 0.95, model = c("wn", "ou", "ouf")[1],
direct = NULL)

Arguments

p estimated mouf parameter vector (tau.z, tau.v, t1, dt, x1, y1, x2, y2)
T time
X x coordinate
Y y coordinate
alpha proportion of area used to be computed
model one of "wn", "ou", "ouf" - whether or not the velocity autocorrelation needs to be taken into account.
direct whether or not to compute the area directly (i.e. fitting a symmetric bivariate normal to the residuals) or to account for the autocorrelation. The default behavior (NULL) computes directly for the "wn" model, and uses the autocorrelation (which is slower) only if the estimated spatial time scale is greater that 1/30 of the total time range.

Details

For sufficient data (i.e. where the range in the times is much greater than the ) This function estimates the (symmetric) 95% area of use from a bivariate Gaussian
getCov

**Estimation Helper Functions**

**Description**

functions which provide the theoretical covariance \([\text{getCov}()]\) and area \([\text{getArea}()]\) for specific models and parameter values

**Usage**

\[
\text{getCov}(t1, t2, \text{model}, p)
\]

**Arguments**

- **t1**
  - time 1
- **t2**
  - time 2
- **model**
  - the model
- **p**
  - vector of the auto-correlation parameters i.e. \(p = (\text{tau.z}, \text{tau.v})\)

**Details**

\[
\text{getCov}(t1, t2, \text{model}, p)
\]
calculates the covariance matrix for different models. \text{mvrnorm2} is a slightly more efficient multivariate normal function.

---

getLikelihood

**Estimate likelihoods and AICs**

**Description**

Estimate likelihoods and AIC for several possible migration models.

**Usage**

\[
\text{getLikelihood}(p, T, X, Y, \text{model} = c(\text{"mouf"}, \text{"mou"}, \text{"mwn"}, \text{"ouf"}, \text{"ou"}, \text{"wn"})[1])
\]

**Arguments**

- **p**
  - initial parameters: \([\text{tau.z, tau.v, t1, t2, x1, x2, y1, y2}]\)
- **T, X, Y**
  - time, x and y coordinates
- **model**
  - \("wn", "ou", "ouf", "mou" or "mouf"\) - whether or not to estimate \(\text{tau.v}\)
getMu

*Obtain mean vector for a range shift process*

**Description**

Obtain a mean vector for a movement with one (getMu) or more (getMu_multi) range shifts. This function is mainly used within the likelihood of range shift processes, but is also useful for simulating processes.

**Usage**

```r
getMu(T, p.m)
```

**Arguments**

- **T**  
  vector of times

- **p.m**  
  mean parameters. A named vector with elements t1, dt, x1, y1, x2, y2, for a single-shift process. For multiple (n) shifts, the parameters are numbered: (x1, x2 ... xn), (y1, y2 ... yn), (t1 .. tn-1), (dt1 ... dt(n-1))

**See Also**

simulate_shift

**Examples**

```r
T <- 1:100
p.m <- c(x1 = 0, y1 = 0, x2 = 10, y2 = 20, t1 = 45, dt = 55)
scan_track(time = T, x=getMu(T, p.m))
```

---

getRSI

*Compute Range Shift Index*

**Description**

The range shift index is a dimensionless measure of the distance of the centroids of two ranges divided by the diameter of the 95% area. This function uses the 95% confidence intervals from a range shift fit to calculate a point estimate and 95% confidence intervals of the RSI.

**Usage**

```r
getRSI(FIT, n1 = 1, n2 = 2, nboot = 1000)
```
gettau

Compute time scale parameters

Description

A mostly internal function that takes the "residuals" of a range-shift process and estimates

\[ \tau_z \]

and, if necessary,

\[ \tau_v \]

.

Usage

getau(Z.res, T = T, model = c("wn", "ou", "ouf")[1], tau0 = NULL, CI = FALSE, method = c("like", "ar")[1])

Arguments

- **Z.res**: complex vector of isotropic Gaussian, possibly autocorrelated time series of points
- **T**: time vector
- **model**: one of "wn" (white noise), "ou" or "ouf" (case insensitive), denoting, respectively, no autocorrelation, position autocorrelation, or velocity and position autocorrelation. If model = NULL and method = "ar", the algorithm will select a model using AIC comparisons of the three. If the selected model is white noise, the function will return 0’s for both parameters.
- **tau0**: initial values of parameter estimates - a named vector: c(tau.z = tau0[1], tau.v = tau0[2])
- **CI**: whether or not to compute the confidence intervals (temporarily only available for like method).
method either "like" or "ar". The former refers to the likelihood method - it is most general (i.e. works with irregular sampling). The latter refers to the autoregressive model equivalence, which is faster but only works with regular sampling.

locate_shift

Interactive locating of range shifting

Description

Plots an x-y, time-x, time-y track of a potential migration process and prompts the user to click on the figure to obtain initial estimates of range centroids and timing of start and end of migrations.

Usage

locate_shift(time, x, y, n.clust = 2, ...)

Arguments

time time (can be a POSIXt)

x x and y coordinates. Can be two separate vectors OR a complex "x" OR a two-column matrix/date-frame.

y see x

n.clust number of ranges (either 2 or 3)

... additional parameters to pass to plot functions

Value

a named vector of initial estimates: if n.clust = 2, c(x1, x2, y1, y2, t1, dt) if n.clust = 3, c(x1, x2, x3, y1, y2, y3, t1, t2, dt1, dt2)

See Also

quickfit, codelocator
**Description**

GPS tracks of one roe deer (*Capreolus capreolus*) in the Italian alps. This deer performs two seasonal migrations, from a wintering ground to a summering ground, back its wintering ground. For several ways to analyze these data, see examples in the *marcher* vignette.

**Usage**

```r
data(michela)
```

**Format**

Data frame containing movements of roe deer with the following columns:

- `id` ID of animal
- `name` Names - for mnemonic convenience - of Italian authors.
- `x,y` In Easting Westing
- `latitude, longitude`
- `time` POSIXct object
- `day` Day of year, counting from January 1 of the first year of observations (thus day 367 is January 2 or the following year).

**References**

For more details, see: Eurodeer.org

**Examples**

```r
data(michela)
with(michela, scan_track(time = time, x = x, y = y))
```

**plot.shiftfit**  
*Plot results of an range-shift fit*

**Description**

Plotting functions for illustrating the results of a range-shift fit.
Usage

```r
## S3 method for class 'shiftfit'
plot(x, ns = c(n.sims = 1000, n.times = 100, n.bins = 10),
     plot.ts = TRUE, stretch = 0, pt.cex = 0.8, pt.col = "antiquewhite",
     CI.cols = NULL, layout = NULL, par = NULL, ...)
```

Arguments

- **x**: a fitted range shift object, i.e. output of the `estimate_shift`
- **ns**: a vector of 3 simulation values, useful for smoothing the bars in the dumbbell plot. For smoothing, it might be recommended to increase the first value, `n.sims` - the number of draws from the fitted migration process.
- **plot.ts**: whether or not to plot the time series as well
- **stretch**: an extra parameter to extend the bars on the dumbbells (in real distance units).
- **pt.cex**: point character expansion.
- **pt.col**: points color.
- **CI.cols**: three shading colors, from lightest to darkest. The default is a sequence of blues.
- **layout**: the default layout places the x-y plot on the left and - if `plot.ts==TRUE` - the respective 1-d time series on the right.
- **par**: graphics window parameters that, by default, look nice with the default layout.
- **...**: additional parameters to pass to plot function (e.g. labels, title, etc.)

Examples

```r
# load simulated tracks
data(SimulatedTracks)

# white noise fit
MWN.fit <- with(MWN.sim, estimate_shift(T=T, X=X, Y=Y))
summary(MWN.fit)
plot(MWN.fit)

if(interactive()){
  # OU fit
  MOUF.fit <- with(MOUF.sim.random,
                   estimate_shift(T=T, X=X, Y=Y,
                                  model = "ouf",
                                  method = "like"))
  summary(MOUF.fit)
  plot(MOUF.fit)

  # Three range fit:
  # it is helpful to have some initial values for these parameters
  # because the automated quickfit() method is unreliable for three ranges
  # in the example, we set a seed that seems to work
  # set.seed(1976)

  MOU.3range.fit <- with(MOU.3range,
```
quickfit

Quick fit of one-step migration

Description

Using k-means clustering to get quick fits of 2 or 3 cluster centers in X-Y coordinates.

Usage

quickfit(T, X, Y, dt = 1, n.clust = 2, plotme = TRUE)

Arguments

T       time
X       x coordinate of movement
Y       y coordinate of movement
dt      duration of migration (arbitrarily = 1)
n.clust number of clusters (2 or 3)
plotme whether or not to plot the result

Details

This function does estimates the locations and times of migration, but not the duration (dt). It is most useful for obtaining a "null" estimate for seeding the likelihood estimation.

Value

a named vector of initial estimates:

- if n.clust = 2 returns t1, dt, x1, y1, x2, y2
- if n.clust = 3 returns t1, dt1, t2, dt2, x1, y1, x2, y2, x3, y3
Examples

```r
require(marcher)

## Load simulated data
data(SimulatedTracks)

# plot the MOU simulation
scan_track(MOU.sim)

# quick fit - setting dt = 10
(pm.0 <- with(MOU.sim, quickfit(T, X, Y, dt = 10)))

# interactive locator process
if(interactive()){
  (with(MOU.sim, locate_shift(T, X, Y)))
}

# fit the model
fit <- with(MOU.sim, estimate_shift(T, X, Y))

## Three cluster example

# plot the three range shift simulation
scan_track(MOU.3range)

# quick fit
## (note - this may not always work!)
with(MOU.3range, quickfit(T, X, Y, dt = 10, n.clust = 3))

if(interactive()){
  with(MOU.3range, locate_shift(T, X, Y, n.clust = 3))
}
```

Description

Plotting x-y, time-x, time-y scan of a track. This function will take x, y, and time coordinates or a track class object.

Usage

```r
scan_track(track = NULL, time, x, y = NULL, layout = NULL,
  auto.par = NULL, col = 1, alpha = 0.5, cex = 0.5, ...)```
selectModel

Select residual model

**Description**

Given a complex vector of movement residuals, will use AIC to select the order of the autocorrelation, i.e. white noise (WN), position autocorrelation (OU), or position and velocity autocorrelation (OUF).

### Arguments

- **track**: a track class object, or any data-frame that contains (at least) three columns labeled "T", "X" and "Y"
- **time**: time (can be a POSIXt)
- **x**: x Coordinate. x,y coordinates can be two separate vectors OR a complex "x" OR a two-column matrix/date-frame.
- **y**: y coordinate.
- **layout**: the default layout places the x-y plot on the left and the respective 1-d time series on the right.
- **auto.par**: by default, uses a decent looking default layout. Otherwise can be a par list, or, e.g. FALSE to keep externally defined settings.
- **col**: color vector
- **alpha**: intensity of the color
- **cex**: character expansion of the points
- **...**: options to be passed to plot functions

### Examples

```r
## Roe deer data

data(Michela)
par(bty="l", mar = c(4,4,2), oma=c(4,4,4,0), xpd=NA)
with(Michela, scan_track(time = time, x = x, y = y, main="Michela"))

## Simulated track

time <- 1:200
Mean <- getMu(T = time, p.m = c(x1 = 0, y1 = 0, x2 = 10, y2 = 10, t1 = 90, dt = 20))
SimTrack <- simulate_shift(T = time, tau = c(tau.z = 5), mu = Mean, A = 40)
with(SimTrack, scan_track(time = T, x = X, y = Y))

# OR (because SimTrack is a "track")
scan_track(SimTrack)
```
Usage

selectModel(Z.res, T = NULL, method = c("ar", "like")[1],
             showtable = FALSE)

Arguments

Z.res complex vector of residuals
T time vector (only needed for method = 'like')
method One of 'ar' or 'like' - whether to use the AR equivalence (faster, but needs to be
         regular) or likelihood estimation.
showtable whether to return the AIC values of the respective models

Value

A character string - 'wn', 'ou' or 'ouf'. Optionally also the AIC table.

Examples

require(marcher)

# white noise example
Z1 <- rnorm(100) + 1i*rnorm(100)

# OU example
T <- 1:100
p.s2 <- c(tau.z = 5, tau.v = 0)
S2 <- outer(T, T, getCov, psp.s2, model="ou")
Z2 <- mvnrnorm(n = 1, mu = rep(0,length(T)), S2) +
     1i * mvnrnorm(n = 1, mu = rep(0,length(T)), S2)

# OUF example
p.s3 <- c(tau.z = 5, tau.v = 2)
S3 <- outer(T, T, getCov, psp.s3, model="ouf")
Z3 <- mvnrnorm(n = 1, mu = rep(0,length(T)), S3) +
     1i * mvnrnorm(n = 1, mu = rep(0,length(T)), S3)

# plot all three
par(mfrow=c(1,3), mar = c(2,2,2,2))
plot(Z1, asp=1, type="o")
plot(Z2, asp=1, type="o")
plot(Z3, asp=1, type="o")

# select models using 'ar' method (results might vary!)

selectModel(Z1, T = T, method = "ar", showtable = TRUE)
selectModel(Z2, T = T, method = "ar", showtable = TRUE)
selectModel(Z3, T = T, method = "ar", showtable = TRUE)

selectModel(Z1, T = T, method = "like", showtable = TRUE)
selectModel(Z2, T = T, method = "like", showtable = TRUE)
selectModel(Z3, T = T, method = "like", showtable = TRUE)

# repeat using irregular times (requiring "like" method)
T <- cumsum(rexp(100))

# white noise example
p.s1 <- c(tau.z = 0, tau.v = 0)
S1 <- outer(T, T, getCov, p=p.s1, model="wn")
Z1 <- mvrnorm2(n = 1, mu = rep(0,length(T)), S1) +
  1i * mvrnorm2(n = 1, mu = rep(0,length(T)), S1)

# OU example
p.s2 <- c(tau.z = 5, tau.v = 0)
S2 <- outer(T, T, getCov, p=p.s2, model="ou")
Z2 <- mvrnorm2(n = 1, mu = rep(0,length(T)), S2) +
  1i * mvrnorm2(n = 1, mu = rep(0,length(T)), S2)

# OUF example
p.s3 <- c(tau.z = 5, tau.v = 2)
S3 <- outer(T, T, getCov, p=p.s3, model="ouf")
Z3 <- mvrnorm2(n = 1, mu = rep(0,length(T)), S3) +
  1i * mvrnorm2(n = 1, mu = rep(0,length(T)), S3)

Z.list <- list(Z1, Z2, Z3)

# plot
par(mfrow=c(1,3), mar = c(2,2,2,2))
lapply(Z.list, function(z) plot(z, asp=1, type="o"))

# select model
lapply(Z.list, function(z) selectModel(z, T = T, method = "like", showtable = TRUE))
Details

The data frames are also `track` class object frame.

- **MOU.range** Simulated migratory Ornstein-Uhlenbeck with 3 range
- **MOU.sim** Simulated migratory Ornstein-Uhlenbeck
- **MOUF.sim** Simulated migratory Ornstein-Uhlenbeck Flemming
- **MOUF.sim.random** Simulated migratory Ornstein-Uhlenbeck Flemming at random or arbitrary times of observation
- **MWN.sim** Simulated migratory white noise ranging model

Source

Code to simulate tracks like these are provided in the marcher vignette.

Examples

```r
data(SimulatedTracks)
simulate_shift()
simulate_shift()
simulate_shift()
simulate_shift()
simulate_shift()
```

Description

Simulate MOUF process

Usage

```r
simulate_shift(T, tau = NULL, mu, A)
```

Arguments

- **T** time
- **tau** variance parameters - named vector with 'tau.z' and 'tau.v'
- **mu** mean vector - typically output of `getMu`. Can also be any complex or a two-column matrix, or a multi-column matrix with some named columns "x" and "y" (case-insensitive).
- **A** 95% area parameter

Value

a data frame with Time, X, and Y columns.
test_rangeshift

See Also

getMu

Examples

```r
require(marcher)

# 95% home range area
A <- 20
# distance of migration
D <- 100
# centers of attraction
x1 <- 0; y1 <- 0
x2 <- sqrt(D); y2 <- sqrt(D)
# time scales
tau.z <- 5
tau.v <- 0.5

t1 <- 90
dt <- 20

# mean parameters (t1,dt)
mus <- c(t1=t1,dt=dt,x1=x1,y1=y1,x2=x2,y2=y2)
# time-scale parameters
taus <- c(tau.z = tau.z, tau.v = tau.v)

# generate and plot mean vector
T <- 1:200
Mu <- getMu(T, mus)

# simulate and plot MOUF process
SimTrack <- simulate_shift(T, tau=taus, Mu, A=A)
with(SimTrack, scan_track(time=T,x=X,y=Y))
```

description

Three tests for three hypotheses to test on fitted range shifts: Was the range shift significant? Did an animal that performed two consecutive seasonal migrations return to the same location it began? Was there a stopover during a migration?

Usage

test_rangeshift(FIT, verbose = TRUE)

test_return(FIT, verbose = TRUE)

test_stopover(FIT, verbose = TRUE)
Arguments

FIT a fitted range shift (output of `estimate_shift`)
verbose whether to print verbose message

Value

Outputs a summary of the test results and returns a list of test results including:

- aic.table an AIC table comparing models
- lrt a likelihood ratio test statistic
- df degrees of freedom for the l.r.t.
- p.value a p.value for the l.r.t.

Functions

- test_rangeshift: Compare a two range fitted model to a null model of no range shift.
- test_return: Compares a three range fitted model in which the first and third ranges have the same centroid against a model where the first and third centroid are different.
- test_stopover: Compare a three range model with an apparent stopover (shorter intermediate range), and see if a more parsimonious model excludes the stopover.
Index

*Topic data
  Michela, 11
  SimulatedTracks, 17

estimate.mouf.nls (estimate_shift), 3
estimate_shift, 2, 3, 9, 12, 20
fitNSD, 5
getMu.nls (estimate_shift), 3
getAIC.nls (getLikelihood), 7
getArea, 6
getCov, 7
getLikelihood, 7
getMu, 8, 18, 19
getMu_multi (getMu), 8
getP.nls (estimate_shift), 3
getRSI, 8
getTau, 9
locate_shift, 10
locator, 3, 10

marcher, 11
marcher (marcher-package), 2
marcher-package, 2
Michela, 2, 11
MOU.3range (SimulatedTracks), 17
MOU.sim (SimulatedTracks), 17
MOUF.sim (SimulatedTracks), 17
mvnorm2 (getCov), 7
MWN.sim (SimulatedTracks), 17

par, 15
plot.shiftfit, 2, 11
POSIXt, 10, 15

quickfit, 3, 10, 13
scan_track, 14
selectModel, 3, 15

simulate_shift, 2, 8, 18
SimulatedTracks, 2, 17
test_rangeshift, 2, 19
test_return, 2
test_return (test_rangeshift), 19
test_stopover, 2
test_stopover (test_rangeshift), 19