Package ‘bcpa’

February 19, 2015

Type Package
Title Behavioral change point analysis of animal movement
Version 1.1
Date 2014-10-21
Author Eliezer Gurarie <eliezg@uw.edu>
Maintainer Eliezer Gurarie <eliezg@uw.edu>
Description The Behavioral Change Point Analysis (BCPA) is a method of identifying hidden shifts in the underlying parameters of a time series, developed specifically to be applied to animal movement data which is irregularly sampled. The method is based on: E. Gurarie, R. Andrews and K. Laidre A novel method for identifying behavioural changes in animal movement data (2009) Ecology Letters 12:5 395-408.
License Unlimited
Depends Rcpp (>= 0.10.2), plyr
LinkingTo Rcpp
Suggests knitr, circular
VignetteBuilder knitr
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-11-02 09:15:10

R topics documented:

bcpa-package .................................................. 2
ChangePointSummary ............................................. 3
DiagPlot .......................................................... 5
GetBestBreak ..................................................... 6
GetDoubleL ....................................................... 7
GetL ............................................................... 7
GetModels ......................................................... 9
Description

A collection of functions that allows one to perform the behavioral change point analysis (BCPA) as described by Gurarie et al. (2009, Ecology Letters, 12: 395-408). The key features are estimation of discrete changes in time-series data, notable linear and turning components of gappy velocity times series extracted from movement data.

There is a fairly detailed vignette accessible by entering `vignette("bcpa")`. Alternatively, the key analysis function is `WindowSweep`, and reading its documentation is a good way to start using this package. This function uses a suite of functions that might also be useful for more narrow analysis, listed hierarchically (from bottom-up) below:

- `GetRho` maximizes the likelihood to estimate autocorrelation rho or characteristic time-scale tau.
- `GetDoubleL` estimates the parameters and returns the log-likelihood at either side of a given break.
- `GetBestBreak` finds the single best change point according to the likelihood returned by `GetDoubleL`.
- `GetModels` uses a (modified) BIC model selection for all combinations from M0 ($\mu_1 = \mu_2, \sigma_1 = \sigma_2, \rho_1 = \rho_2$) to M7 ($\mu_1 \neq \mu_2, \sigma_1 \neq \sigma_2, \rho_1 \neq \rho_2$) to characterize the “Best Break”.
- `WindowSweep` sweeps a longer time series with the Best Break / Model Selection analysis, identifying most likely break points.

Summary, diagnostic, and plotting functions are:

- `PartitionParameters` outputs the estimated parameters of a bcpa.
- `ChangePointSummary` provides a summary table of the change points.
- `plot.bcpa` a plotting method for visualizing the time series with vertical lines as change points.
- `PathPlot` a method for drawing a color-coded track of the analysis.
- `DiagPlot` diagnostic plots for BCPA.

A few preprocessing functions available:

- `plot.track` method for plotting a generic “track” object.
- `GetVT` returns step-lengths, absolute and turning angles from track data.
Details

Package: bcpa
Type: Package
Version: 1.0
Date: 2013-10-23
License: Unlimited
LazyLoad: yes

This is a suite of functions needed to perform a complete behavioral change point analysis.

Author(s)

Eliezer Gurarie <eliezg@uw.edu>

References


Examples

# Running through a complete analysis here:
## loading the data
data(Simp)
## plotting the track (using the plot.track method)
plot(Simp)
## Obtaining the movement summary table (with turning angles and step lengths)
Simp.VT <- GetVT(Simp)
## Applying the analysis
Simp.ws <- WindowSweep(Simp.VT, "V*cos(Theta)", windowsize = 50, windowstep = 1, progress=TRUE)
## plotting outputs
plot(Simp.ws, threshold=7)
plot(Simp.ws, type="flat", clusterwidth=3)
PathPlot(Simp, Simp.ws)
PathPlot(Simp, Simp.ws, type="flat")
## Diagnostic of assumptions
DiagPlot(Simp.ws)

**Obtain summary of BCPA analysis**

ChangePointSummary
Description

Produces a summary of change points for a "flat" analysis, identifying phases (periods between change points) with estimated parameters, clustering neighboring ones according to a kernel density of the windowsweep breaks.

Usage

`ChangePointSummary(windowsweep, clusterwidth = 1, tau = TRUE)`

Arguments

- `windowsweep`: a windowsweep object, i.e. the output of the `windowsweep` function.
- `clusterwidth`: the temporal range within which change points are considered to be within the same cluster. Corresponds to the bandwidth of the density of the break distribution.
- `tau`: logical, whether to estimate the characteristic time tau (preferred) or not. If FALSE, the autocorrelation parameter rho is calculated.

Value

A list containing two elements:

- `breaks`: a data frame containing columns: `middle` - each change point, `size` - the number of windows that selected the change point, `modelmode` - the most frequently selected of the seven possible models (M0 - the null model - is excluded), and `middle.POSIX` - the mid-point as a POSIX time object.

- `phases`: a data frame containing columns `mu.hat`, `s.hat`, `rho.hat` - the estimated mean, standard deviation, and time scale (or auto-correlation) within each phase (i.e. period between change points), `t0` - the beginning of the phase, `t1` - the end of the phase, and `interval` - the total duration of the phase.

Author(s)

Eliezer Gurarie

See Also

`windowsweep`

Examples

```r
if(!exists("Simp.VT")){
  data(Simp)
  Simp.VT <- GetVT(Simp)}
if(!exists("Simp.ws"))
  Simp.ws <- WindowSweep(Simp.VT, "V*cos(Theta)", windowsize = 50, windowstep = 1, progress=TRUE)
# too many change points:
  ChangePointSummary(Simp.ws)
# about the right number of change points:
  ChangePointSummary(Simp.ws, clusterwidth=3)
```
DiagPlot

Diagnostic plot for BCPA

Description

Draws diagnostic plots for BCPA analysis. Specifically: a qqplot, a histogram (with a N(0,1) density curve), and an acf of the standardized residuals of an analysis.

Usage

DiagPlot(windowsweep, type = c("smooth", "flat")[-1], plotme = TRUE, values = FALSE, ...)

Arguments

windowsweep a windowsweep object, i.e. the output of the WindowSweep analysis.
type whether to diagnose the model fitted for a smooth or flat BCPA.
plotme logical - whether or not to plot the diagnostics
values logical - whether or not to return the values of the standardized residuals.
... additional arguments to pass to the PartitionParameters function.

Value

If values is TRUE, return the values of the standardized residuals.

Author(s)

Eliezer Gurarie

See Also

PartitionParameters

Examples

data(Simp)
if(!exists("Simp.VT"))
  Simp.VT <- GetVT(Simp)
if(!exists("Simp.ws"))
  Simp.ws <- WindowSweep(Simp.VT, "V*cos(Theta)", windowsize = 50, windowstep = 1, progressTRUE)
DiagPlot(Simp.ws)
DiagPlot(Simp.ws, type="flat")
# The Simp's diagnostic plots are excellent.
GetBestBreak

Find most likely change point in irregular time series

Description

Finds the single best change point according to the likelihood function. Used internally within WindowSweep.

Usage

GetBestBreak(x, t, range = 0.6, ...)

Arguments

- **x**: vector of time series values.
- **t**: vector of times of measurements associated with x.
- **range**: range of possible breaks. Default (0.6) runs approximately from 1/5th to 4/5ths of the total length of the time series.
- **...**: additional parameters to pass to GetDoubleL function.

Value

returns a single row (vector) with elements: breaks,tbreaks,mu1,sigma1,rho1,LL1,mu2,sigma2,rho2,LL2,LL. The breakpoint is calculated for a range of possible values of width range*l (where l is the length of the time series). The output of this function feeds WindowSweep.

Author(s)

Eliezer Gurarie

See Also

- WindowSweep which uses it, and GetDoubleL for the likelihood estimation.

Examples

```r
# An example with a single break:
x <- c(arima.sim(list(ar = 0.9), 20) + 10, arima.sim(list(ar = 0.1), 20))
t <- 1:length(x)
plot(t,x, type="l")
(bb <- GetBestBreak(x,t, tau=FALSE))
abline(v = bb[2], col=2)
```
GetDoubleL

Obtain log-likelihood and parameter estimates for a given break point.

Description
Takes a time series with values x obtained at time t and a time break tbreak, and returns the estimates of μ, σ and τ (or ρ) as well as the negative log-likelihood of those estimates before and after the break. Mostly for use internally within GetBestBreak.

Usage
GetDoubleL(x, t, tbreak, ...)

Arguments
x vector of time series values.
t vector of times of measurements associated with x.
tbreak breakpoint to test (in terms of the INDEX within "t" and "x", not actual time value).
... additional parameters to pass to GetRho.

Value
a vector containing the parameters and the negative log-likelihoods in order: mu1, sigma1, tau1, LL1,mu2, sigma2, tau2

Author(s)
Eliezer Gurarie

See Also
GetBestBreak uses this function, while this function uses GetRho

GetL
Obtain likelihood of gappy Gaussian time series

Description
Obtain likelihood of gappy standardized Gaussian time series "x" sampled at times "t" given parameter "rho" (autocorrelation). Alternatively computes the characteristic time scale "tau".

Usage
GetL(x, t, rho, tau = FALSE)
**Arguments**

- `x` Time series
- `t` Sampling times
- `rho` Auto-correlation
- `tau` logical: Whether or not to compute characteristic time scale instead of `rho`.

**Value**

Returns the log-likelihood of the data.

**Author(s)**

Eliezer Gurarie

**See Also**

Core function of BCPA, used directly in `GetRho`

**Examples**

```r
# create full time-series
rho <- 0.8
x.full <- arima.sim(1000, model=list(ar = rho))
t.full <- 1:1000

# subsample time series
keep <- sort(sample(1:1000, 200))
x <- x.full[keep]
t <- t.full[keep]
plot(t,x, type="l")

# sweep values of rho
rhos <- seq(0, .99, .01)
L <- rep(NA, length(rhos))
for(i in 1:length(rhos))
  L[i] <- GetL(x,t,rhos[i])

# plot likelihood profile
plot(rhos, L, type="l")
abline(v = rhos[L == max(L)], lty=3, lwd=2)
abline(v = rho, lty=2, lwd=2)
legend("bottomleft", legend=c("true value","MLE"), lty=3:2, lwd=2)
```
Model selection at a known breakpoint

Description

Returns all parameter estimates and log-likelihoods for all possible models at a selected breakpoint. These are:

- M0 - all parameters equal
- M1 - $\mu_1 \neq \mu_2$
- M2 - $\sigma_1 \neq \sigma_2$
- M3 - $\tau_1 \neq \tau_2$
- M4 - $\mu_1 \neq \mu_2$ AND $\sigma_1 \neq \sigma_2$
- M5 - $\mu_1 \neq \mu_2$ AND $\tau_1 \neq \tau_2$
- M6 - $\sigma_1 \neq \sigma_2$ AND $\tau_1 \neq \tau_2$
- M7 - all parameters unequal

Usage

GetModels(x, t, breakpoint, K = 2, tau = TRUE)

Arguments

- x: vector of time series values.
- t: vector of times of measurements associated with x.
- breakpoint: breakpoint to test (in terms of the INDEX within "t" and "x", not actual time value).
- K: sensitivity parameter. Standard definition of BIC, K = 2. Smaller values of K mean less sensitive selection, i.e. higher likelihood of selecting null (or simpler) models.
- tau: whether or not to estimate time scale $\tau$ (preferred) or autocorrelation $\rho$.

Value

Returns a names matrix with 8 rows (one for each model) and columns: Model, LL, bic, mu1, s1, rho1, mu2, s2, rho2. Fairly self-explanatory. Note that the rho columns include the tau values, if tau is TRUE (as it usually should be).

Author(s)

Eliezer Gurarie

See Also

Used directly within WindowSweep. Relies heavily on GetRho.
GetRho

*Characteristic time / auto-correlation for irregular time series*

**Description**

Estimates characteristic time $\tau$ or auto-correlation $\rho$ from a gappy time series dataset. It works first by estimating the mean and standard deviation directly from the time series $X$, using these to standardize the time series, and then optimizes for the likelihood of the value of $\tau$ or $\rho$.

**Usage**

GetRho(x, t, tau = TRUE)

**Arguments**

- **x**: vector of time series values.
- **t**: vector of times of measurements associated with $x$.
- **tau**: whether or not to estimate time scale $\tau$ (preferred) or autocorrelation $\rho$.

**Value**

Returns a vector of length two: the estimate and the negative log-likelihood

**Author(s)**

Eliezer Gurarie

GetVT

*Obtain VT table from Track*

**Description**

The VT table contains speeds, steplengths, orientations and other summaries derived from a track. The output of this function is (typically) meant to feed the WindowSweep function.

**Usage**

GetVT(Data, units = "hour", skiplast = TRUE)

**Arguments**

- **Data**: a track to analyze. Must contain columns: X, Y and Time (as a POSIX object). The track class is a robust entry.
- **units**: the time units for the analysis; one of sec, min, hour, day.
- **skiplast**: filters away last step.
Value

a data frame with the following columns:

- Z.start, Z.end: the start and end locations (as complex coordinates)
- S: the step length
- Phi, Theta: absolute and turning angle, respectively
- T.start, T.end: start and time of steps (numeric - in given units)
- T.mid: temporal midpoint of the step
- dT: duration of the step
- V: approximate speed (S/dT)
- T.POSIX: the temporal midpoint of the step as a POSIX objects.

Author(s)

Eliezer Gurarie

See Also

WindowSweep

Examples

data(simpI)
plot(simpI)
Simp.VT <- GetVT(Simp)
head(Simp.VT)
# Distribution of estimated speeds
hist(Simp.VT$V, col="grey", breaks=20)
# Distribution of turning angles
require(circular)
rose.diag(Simp.VT$theta, bins=24)

Description

Simple convenience function for creating a track class object from X-Y-Time movement data. A track class object can be conveniently plotted and analyzed within bcpa.

Usage

MakeTrack(X, Y, Time)
Arguments

X vector of X locations
Y vector of Y locations
Time vector of time (can be POSIX)

Value

a track class data frame, with three columns: X, Y and Time.

See Also

plot.track

Examples

X <- cumsum(arima.sim(n=100, model=list(ar=0.8)))
Y <- cumsum(arima.sim(n=100, model=list(ar=0.8)))
Time <- 1:100
mytrack <- MakeTrack(X, Y, Time)
plot(mytrack)
PathPlot

Author(s)
Eliezer Gurarie

See Also
used in ChangePointSummary and PhasePlot

PathPlot
Path plot of BCPA output

Description
Plots the animal’s trajectory, with segments color-coded according to the time scale / auto-correlation of the BCPA output, and width of segments proportional to the estimated mean of the BCPA.

Usage
PathPlot(Data, windowsweep, type = c("smooth", "flat")[1], clusterwidth = 1, plotlegend = FALSE, tauwhere = "topleft", n.legend = 5, ncol.legend = 1, bty.legend = "n", ...)

Arguments
- **Data**: the track data to be plotted - most typically, output of the GetVT function.
- **windowsweep**: windowsweep object, i.e. the output of the WindowSweep function.
- **type**: whether to plot smooth or flat bcpa output
- **clusterwidth**: for flat BCPA, this is the temporal range within which change points are considered to be within the same cluster.
- **plotlegend**: whether to plot a legend.
- **tauwhere**: where to place the legend for the time-scale / auto-correlation. Can be one of "nowhere", "top", "bottom", "left", "right", "topleft", "topright", "bottomright", "bottomleft".
- **n.legend**: number of labels in legend.
- **ncol.legend**: number of columns in the legend.
- **bty.legend**: whether to draw a box around the legend.
- **...**: additional arguments to pass to the plot base function.

Author(s)
Eliezer Gurarie
Examples

```r
if(!exists("Simp.ws"))
{
  data(Simp)
  Simp.ws <- WindowSweep(GetVT(Simp), "V*cos(Theta)", windowsize = 50, windowstep = 1, progress=TRUE)
}

PhasePlot(Simp, Simp.ws, plotlegend=TRUE, n.legend=3)
PhasePlot(Simp, Simp.ws, type="flat", clusterwidth=3, plotlegend=TRUE)
```

---

**PhasePlot**

*Phase plot of BCPA output*

**Description**

Behavioral phase plot of BCPA output. Mean and standard deviation are on the x and y axis. Size and color of points represent the time scale of autocorrelation.

**Usage**

```r
PhasePlot(windowsweep, type = c("smooth", "flat")[1], clusterwidth = 1, ..., legend.where = "bottomright")
```

**Arguments**

- `windowsweep`  
  windowsweep object, i.e. the output of the `WindowSweep` function.
- `type`  
  whether to plot smooth or flat bcpa output
- `clusterwidth`  
  for flat BCPA, this is the temporal range within which change points are considered to be within the same cluster.
- `...`  
  additional arguments passed to the `plot` base function.
- `legend.where`  
  where to place the tau legend (see `legend`).

**Author(s)**

Eliezer Gurarie

**See Also**

`WindowSweep, PartitionParameters`

**Examples**

```r
if(!exists("Simp.ws"))
{
  data(Simp)
  Simp.ws <- WindowSweep(GetVT(Simp), "V*cos(Theta)", windowsize = 50, windowstep = 1, progress=TRUE)
}

PhasePlot(Simp.ws)
```
plot.bcpa

Plotting method for BCPA output

Description

Plotting method for the output of a BCPA analysis with vertical break points, superimposed estimates of the partitioned mean and variance estimates and color-coded autocorrelation estimates.

Usage

```r
## S3 method for class 'bcpa'
plot(x, type = c("smooth", "flat")[1], threshold = 3,
     clusterwidth = 1, col.cp = rgb(0.5, 0, 0.5, 0.5), pt.cex = 0.5,
     legend = TRUE, rho.where = "topleft", mu.where = "nowhere",
     col.sd = "red", col.mean = "black", ...)
```

Arguments

- `x`: a windowsweep object, i.e. the output of the `WindowSweep` function.
- `type`: whether to plot smooth or flat bcpa output
- `threshold`: for smooth BCPA, this is the minimum number of windows that must have identified a given changepoint to be illustrated.
- `clusterwidth`: for flat BCPA, this is the temporal range within which change points are considered to be within the same cluster.
- `col.cp`, `col.mean`, `col.sd`: color of the vertical change points, mean estimate, and prediction interval (mu + sigma), respectively.
- `pt.cex`: expansion coefficient for point sizes.
- `legend`: logical - whether to draw a legend or not.
- `rho.where`: where to place the legend for the time-scale / auto-correlation. Can be one of "nowhere", "top", "bottom", "left", "right", "topleft", "topright", "bottomright", "bottomleft"
- `mu.where`: where (and whether) to place the legend box for the mean - same options as for `rho.where`
- `...`: other arguments to pass to the `plot` base function.

Author(s)

Eliezer Gurarie

See Also

Plots output of the `WindowSweep` function.
plot.track

Plot Track

Description

Default method for plotting tracks

Usage

## S3 method for class 'track'
plot(x, pch = 19, col = rgb(0, 0, 0.2), ...)

Arguments

x object of class "track" to plot
pch default point style (filled circle)
col default color (translucent grey)
... other arguments to pass to plot

Examples

data(Simp)
is(Simp)
plot(Simp)
Simulated Chimp (Simp) Data

Description

This is a simulated movement track from a continuous auto-correlated process of total duration 60 time units with four behavioral phases that switch at times T=10, 20 and 50 from, variously, higher or lower velocity and longer or shorter characteristic time scale of autocorrelation. The original data was simulated with time intervals of 0.01 (leading to a complete track with 6000 data points). 200 points were randomly sampled from the “true” movement track, such that the intervals between locations are random with mean interval 0.3 units.

Usage

data(Simp)

Format

Data frame (of class "track") containing 200 rows and three columns:

- **Time**: times of observation
- **X, Y**: coordinates

Source

For more details, see: http://wiki.cbr.washington.edu/qerm/index.php/Behavioral_Change_Point_Analysis

Examples

data(Simp)
plot(Simp)

---

Perform window sweep for BCPA

WindowSweep

Description

This is the workhorse function of the BCPA. It performs a sweep of the time series, searching for most significant change points and identifying the parsimonious model according to an adjusted BIC.

Usage

WindowSweep(data, variable, windowsize = 50, windowstep = 1, K = 2, tau = TRUE, range = 0.6, progress = TRUE, plotme = FALSE, ...)
Arguments

data the data to be analyzed. Most typically, output of the `GetVT` function containing step lengths, absolute and turning angles, etc.

variable a character string representing the response to apply the BCPA to. For example "$v \times \cos(\theta)$" for persistence velocity, or "$\log(v)$" for log of velocity.

windowsize integer size of the analysis window as a number of data points (not time units). Should probably be no smaller than 20.

windowstep integer step size of analysis. Values greater than 1 speed the analysis up.

K sensitivity parameter for the adjusted BIC. Smaller values make for a less sensitive model selection, i.e. more likely that the null model of no significant changes will be selected.

tau a logical indicating whether the autocorrelation "rho" or the characteristic time "tau" should be estimated.

range a number between 0 and 1 that determines the extent of each window that is scanned for changepoints. i.e., if the window is 100 datapoints long, at the default range=0.6, changepoints will be scanned between 20 and 80.

progress logical - whether or not to output a progress bar as the analysis is being performed.

plotme logical - whether or not to plot the analysis as it is happening. This slows the analysis considerably, especially in non-dynamic graphic environments like RStudio.

additional parameters to be passed to the `PartitionParameters` function.

Value

an object of class `windowsweep`, which is a list containing:

ws a data frame containing the change point, selected model, and parameter estimates

x the response variable

t the time variable - in the units specified in the data object

POSIX the time variable as a POSIX objects (contianing Y-M-D H:S)

windowsize the window size

windowstep the window step size

Author(s)

Eliezer Gurarie

See Also

for internal functions: `GetModels, GetBestBreak, GetDoubleL`; for summarizing output: `ChangePointSummary`; for plotting output: `plot.bcpa`
Examples

data(Simp)
plot(Simp)
Simp.VT <- GetVT(Simp)
Simp.ws <- WindowSweep(Simp.VT, "V*cos(Theta)", windowsize = 50, windowstep = 1, progress=TRUE)
plot(Simp.ws, threshold=7)
plot(Simp.ws, type="flat", clusterwidth=3)
PathPlot(Simp, Simp.ws)
PathPlot(Simp, Simp.ws, type="flat")
DiagPlot(Simp.ws)
Index

*Topic **bcpa**
  - bcpa-package, 2
*Topic **datasets**
  - Simp, 17

bcpa (bcpa-package), 2
bcpa-package, 2

**ChangePointSummary**, 2, 3, 13, 18

**DiagPlot, 2, 5**

**GetBestBreak, 2, 6, 7, 18**
**GetDoubleL, 2, 6, 7, 18**
**GetL, 7**
**GetModels, 2, 9, 18**
**GetRho, 2, 7–9, 10**
**GetRho2 (GetRho), 10**
**GetVT, 2, 10, 13, 18**

**legend, 14**

**MakeTrack, 11**

**PartitionParameters, 2, 5, 12, 14, 18**
**PathPlot, 2, 13**
**PhasePlot, 13, 14**
**plot.bcpa, 2, 15, 18**
**plot.track, 2, 16**

**Simp, 17**

**WindowSweep, 2, 4–6, 9–15, 17**