Behavioral Change Point Analysis in R: The `bcpa` package

Eliezer Gurarie,
Department of Statistics
School of Environmental and Forest Sciences
University of Washington, Seattle
eliezg@u.washington.edu

October 2013
Version 1.1 updated: October 2014

Contents

1 Background

The `bcpa` package is designed to streamline the implementation of the "behavioral change point analysis" (BCPA, Gurarie et al. 2009) for any time-stamped movement data, i.e. in which there are X, Y and T coordinates representing spatial locations and time of observation.

The BCPA was developed in order to identify changes in animal behaviors that were obscured by visual inspection or standard techniques. Specific difficulties associated with movement data include the multi-dimensionality, auto- and cross-correlation, considerable internal structure (reflecting behavioral complexity), and data collection that can be error-ridden or be irregularly sampled. The irregular sampling is a particulaly vexing problem for marine organism data, for which locations are usually transmitted only when the animal is at the surface, while most standard approaches to modeling movement (e.g. the Correlated Random Walk) are only meaningful for regularly sampled data.

The paper had attracted considerable interest from biologists and ecologists that collect animal movement data and are eager to identify structure in the behaviors. Unfortunately, the code that was originally posted as supplementary material was incomplete and poorly documented, making the method somewhat inaccesible to practitioners without strong statistical or programming backgrounds. After responding to (about) the hundredth email to share the code and offer some suggestions, and in light of a bevy of improvements to the code itelf (e.g. greater speed by dropping some routines into C++, greater flexibility in visualizing and presenting the results, more usable "tuning knobs" and flexible syntax) it became clear that bundling the implementation in an R package would be the most efficient and tidy way to make it more accessible.

2 Summary of method

The BCPA uses a likelihood-based method for identifying significant changes in movement parameter values across long, complex datasets by sweeping an analysis window over the timeseries and identifying the most likely changepoints, while simultaneously testing which, if any, of the parameters might have changed at that changepoint.

Implementing the BCPA involves the following steps:

1. Pick a response time-series variable X. One fairly robust variable is the persistence velocity $V_p = V\cos(\theta)$ where V is speed = displacement/time interval and θ is turning angle. Another alternative is simply V.
2. Assume that the observations $X(t)$ are a observations from a stationary continuous-time Gaussian process with mean μ_i, standard deviation σ_i and time-scale of autocorrelation τ_i, where $i \in (1, 2, ...N)$ represents an *a priori* unknown number of behavioral states. *Note: the use of a continuous time scale* $\tau > 0$ *is a change from the original model, which estimates a discrete autocorrelation* $0 < \rho < 1$*. The time-scale is more biologically meaningful, as it is estimated in units of time: the longer the time-scale the longer the "memory" of the movement.*
3. Obtain a likelihood for μ_i, σ_i and ρ_i within a given stationary state i (See Gurarie et al. 2009 for details).
4. Find the location within a window of observations which splits a subset of the data into two sets of the three parameters.

5. Within this window, use a version of BIC to determine which combination (if any) of the three parameters most parsimoniously describes the separation in the data. Often, the null model is returned. I say "modified" because the BIC is usually defined as: $BIC = -2\log(L) + k\log(n)$, where L is the likelihood, k is the number of parameters, and n is the number of data points; however, the 2 is replace with a constant $K > 0$. The smaller this value, the more conservative the analysis, i.e. the more likely it is to select a simpler or null model. This is one of several "tuning knobs" in the BCPA.
6. Sweep a window of fixed size across the time series and collect all the changepoints, the associated models, and the values of the estimated parameters according to the selected model on either side of the changepoint within the window. Note, the window size is another "knob" - larger windows are more robust but more coarse, smaller windows are more sensitive but more likely to give slightly spurious results, compensated by adjusting the K.

These steps summarize the actual analysis. The output of the analysis can be summarized and presented in two ways:

1. Smooth BCPA: A "smooth" output is the one described in the original paper: is the average over all the estimated parameters, and the location of all the change points on the other. This gives interesting output - with, in particular, the opportunity to have parameters change both suddenly and gradually, and to visualize a phase plot of the behavioral shifts - both sudden and gradual.
2. Flat BCPA: A "flat" output takes the result of the window sweep and finds the most frequently chosen change points, clustering those unique changepoints which are close to each other (within some interval dT_c). The three parameters μ, σ and τ are then estimated within each section, and the location of these "flat" changepoints is recorded. This output is directly comparable to the BPMM segmentation.

3 Detailed implementation

The complete analysis can now be performed in, essentially, two or three lines of code (see final sections of this vignette). Before leaping into them, I review the assumptions and fundamental pieces of the method, using the functions within `bcpa` to facilitate implementation.

3.1 Estimating auto-correlation / characteristic time-scale for irregular data

The BCPA (as currently implemented) analyzes a one-dimensional, arbitrarily sampled autocorrelated Gaussian time-series $X(t)$, specified by three parameters, a mean μ, a standard deviation σ and a characteristic time-scale τ (or autocorrelation coefficient ρ), such that:

$$\begin{aligned} \mathrm{E}X(t) &= \mu \\ \mathrm{Var}X(t) &= \sigma^2 \\ \mathrm{Cor}X(t)X(t-\Delta t) &= \exp(-\Delta t/\tau) = \rho^{\Delta t}, \end{aligned}$$

The characteristic time-scale is an innovation over the original implementation, which estimated ρ. The interpretation of ρ, which ranges from 0 to 1, depends on the units of the time measurement - and can flirt with being uninformatively close to 0 or 1, whereas the time scale is a measurement of the temporal range of correlation in the movement, with somewhat more intuitive biological interpretation (Gurarie and Ovaskainen 2011).

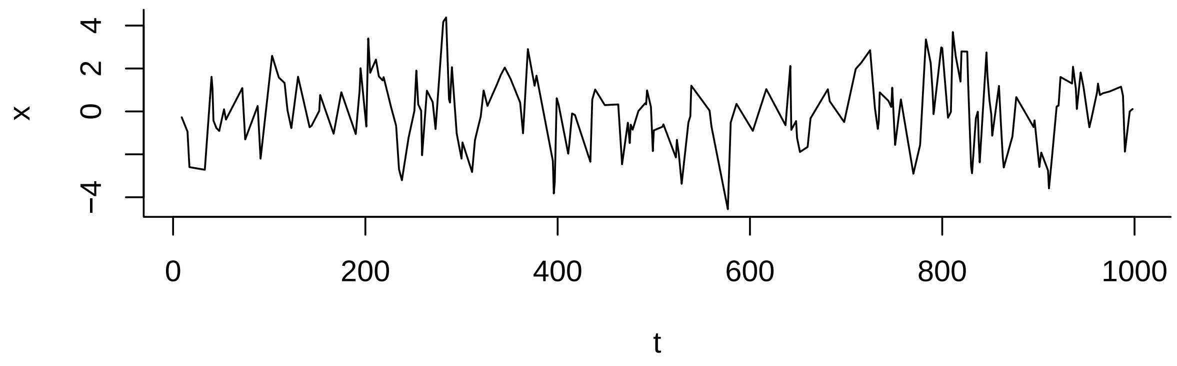
These relationships are used to obtain a likelihood for observations X_i observed at times T_i, and the likelihood is maximized to estimate the characteristic time scale. In the `bcpa` package, this is done with the `GetRho` function, which is driven by the `GetL` function (encoded in C++). Examples are given below:

Loading the package

```
library(bcpa)
```

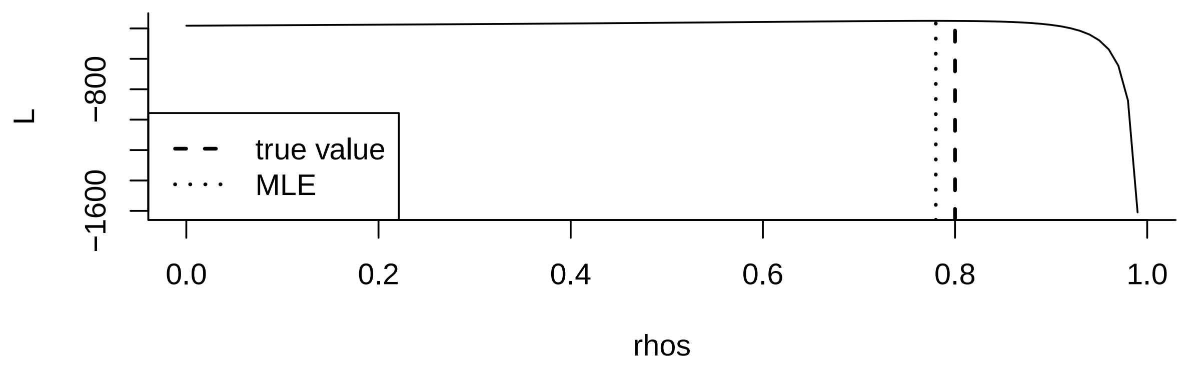
Simulating a gappy, Gaussian, time series

```
rho <- 0.8
x.full <- arima.sim(1000, model=list(ar = rho))
t.full <- 1:1000
keep <- sort(sample(1:1000, 200))
x <- x.full[keep]
t <- t.full[keep]
plot(t,x, type="l")
```



Obtaining the likelihood function for different values of ρ.

```
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 = rho, lty=2, lwd=2); abline(v = rhos[L == max(L)], lty=3, lwd=2)
legend("bottomleft", legend=c("true value","MLE"), lty=2:3, lwd=2)
```



Using the `GetRho` function to estimate ρ or τ:

```
GetRho(x, t, tau=FALSE)


##      rho.hat           LL
##    0.7764189 -350.2888257


GetRho(x, t, tau=TRUE)


##     rho.hat          LL
##    3.951054 -350.288825
```

Future work: It is straightforward to obtain approximate confidence intervals around this estimate using the Hessian of the log-likelihood.

3.2 Speeds and turning angles

To apply the method to movement data of the form $\mathbf{Z}_i, T_i$, where $\mathbf{Z}_i$ represents the location vector at time T_i, it is necessary to extract a one-dimensional time series that conforms to the assumption. Examples which generally conforms to the assumptions are the "persistence" velocity velocity $V_p(t)$ and turning velocity $V_t(t)$:

$$V_p(T_i) = V(T_i)\cos(\Theta(T_i)) \quad (1)$$
$$V_t(T_i) = V(T_i)\sin(\Theta(T_i)) \quad (2)$$

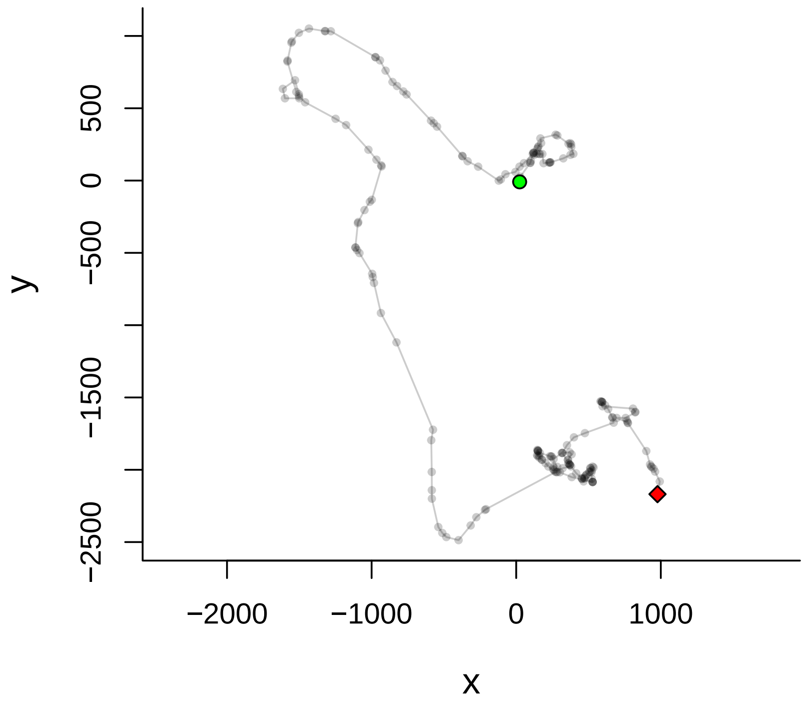
where $V(T_i) = ||\mathbf{Z}_i - \mathbf{Z}_{i-1}||/(T_i - T_{i-1})$ is the scalar speed at time T_i. V_p captures the tendency and magnitude of a movement to persist in a given direction while V_t captures the tendency of movement to head in a perpendicular direction in a given time interval. Thus, the primary descriptive features of movement, namely speed, directional persistence, and variability are captured in these variables. Alternatively, the log of step lengths can have a roughly Gaussian distribution.

For demonstration purposes, we include a simulated movement data set called `Simp`:

```
data(Simp)
head(Simp)
```

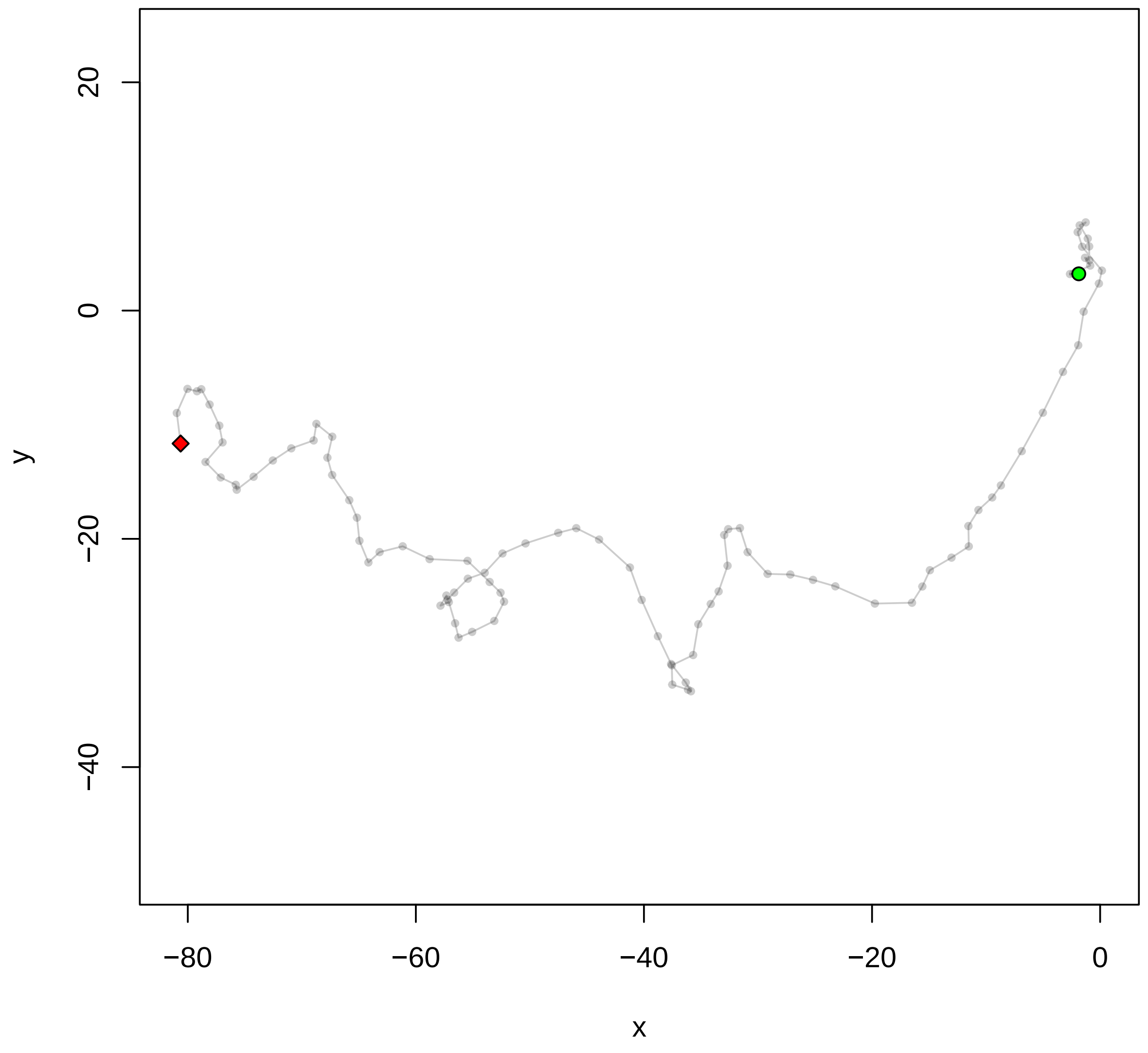
```
##   Time        X          Y
## 1 0.18 23.74779  -9.063781
## 2 0.22 26.74591 -11.325314
## 3 0.74 13.25269   2.945536
## 4 0.88 28.46158  26.634281
## 5 1.40 96.77801 121.783294
## 6 1.41 97.85599 123.347554
```

```
plot(Simp)
```



The `Simp` objects is of class "track", which is simply a data frame with columns X, Y and T, and the `bcpa` package contains a plotting method for a track of this form with a green circle illustrating the start and a red rhombus indicating the end of the track. The `MakeTrack` function was added to facilitate the creation of a "track" class object, e.g.:

```
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)
```



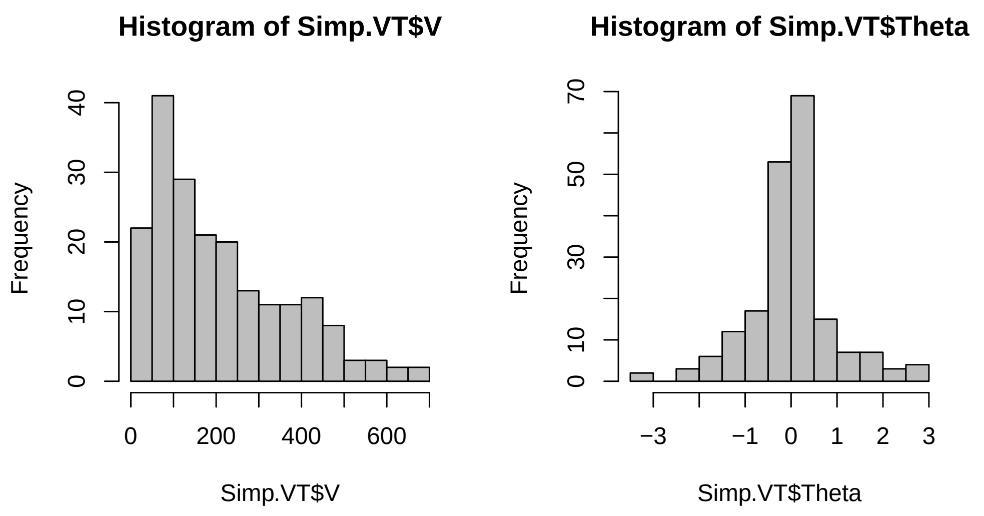
To obtain the step length and turning angles, use the `GetVT` function, which decomposes the data into single step and all the relevant statistics:

```
Simp.VT <- GetVT(Simp)
head(Simp.VT)
```

```
##                  Z.start                  Z.end          S       Phi       Theta
## 2  26.74591- 11.325314i  13.25269+  2.945536i  19.639857 2.3281931  2.97445205
## 3  13.25269+  2.945536i  28.46158+ 26.634281i  28.150790 1.0000442 -1.32814890
## 4  28.46158+ 26.634281i  96.77801+121.783294i 117.134403 0.9480956 -0.05194857
## 5  96.77801+121.783294i  97.85599+123.347554i   1.899727 0.9673998  0.01930417
## 6  97.85599+123.347554i 101.46842+135.403697i  12.585716 1.2796754  0.31227561
## 7 101.46842+135.403697i 117.73868+193.351099i  60.188228 1.2970678  0.01739243
##   T.start T.end T.mid   dT         V T.POSIX
## 2    0.22  0.74 0.480 0.52  37.76896   0.480
## 3    0.74  0.88 0.810 0.14 201.07707   0.810
## 4    0.88  1.40 1.140 0.52 225.25847   1.140
## 5    1.40  1.41 1.405 0.01 189.97268   1.405
## 6    1.41  1.48 1.445 0.07 179.79594   1.445
## 7    1.48  2.00 1.740 0.52 115.74659   1.740
```

The overall persistence of the movement and distribution of step lengths:

```
par(mfrow=c(1,2))
hist(Simp.VT$V, breaks=20, col="grey")
hist(Simp.VT$Theta, breaks=20, col="grey")
```



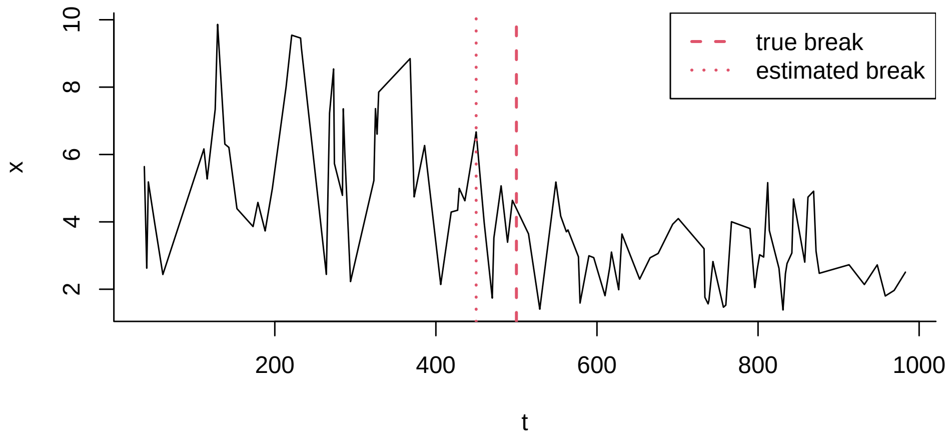
3.3 Obtaining a changepoint

A single changepoint in a time-series where the parameters change at some unknown timepoints t^* is done by sweeping all possible breaks and finding the most likely changepoint according to the likelihood. This is illustrated below:

```
par(bty="l")
mu1 <- 5; mu2 <- 3
sigma1 <- 2; sigma2 <- 1
rho1 <- 0.5; rho2 <- 0.5

SimTS <- function(n, mu, rho, sigma){
  X.standard <- arima.sim(n, model=list(ar = rho))
  X.standard/sd(X.standard)*sigma + mu
}

# create time series with break at 500
t.full <- 1:1000
t.break <- 500
x.full <- c(SimTS(t.break, mu1, rho1, sigma1),
            SimTS(max(t.full)-t.break+1, mu2, rho2, sigma2))

# subsample 100 observations and estimate
keep <- sort(sample(1:length(x.full), 100))
x <- x.full[keep]
t <- t.full[keep]
(BB <- GetBestBreak(x,t, tau=FALSE))


##      bb.index       bb.time           mu1            s1       rho.hat            LL
##    42.0000000   450.0000000     5.7929429     2.0770893     0.7414713   -86.0806849
##           mu2            s2       rho.hat            LL   ll.total.LL
##     3.0699498     1.1395844     0.7687888   -82.4455930  -168.5262779
```

The estimates should be fairly good (note that we are choosing to estimate ρ rather than τ).

```
plot(t,x, type="l")
abline(v = 500, col=2, lwd=2, lty=2); abline(v = BB[2], col=2, lwd=2, lty=3)
legend("topright", legend=c("true break", "estimated break"), col=2, lwd=2, lty=2:3)
```



The likelihood is used to obtain a BIC value for all of the possible models. The possible models are numbered M0 to M7, corresponding to no significant changes (M0), only μ, σ or ρ changing (M1, M2, M3, respectively), both of μ and σ, μ and ρ and σ and ρ changing (M4, M5, M6), and all three parameters changing (M7). The model are compared with the `GetModels` function:

```
GetModels(x,t,BB[1], tau=FALSE)
```

```
##      Model        LL      bic      mu1       s1      rho1      mu2       s2
## [1,]     0 -192.7610 399.3375 4.177607 2.079177 0.8898607 4.177607 2.079177
## [2,]     1 -178.8822 380.7902 5.792943 1.553832 0.7153161 3.007881 1.553832
## [3,]     2 -195.8923 414.8104 4.177607 2.077089 0.8963446 4.177607 1.044103
## [4,]     3 -197.4125 417.8509 4.177607 2.079177 0.7414713 4.177607 2.079177
## [5,]     4 -166.9409 361.5128 5.792943 2.077089 0.7294510 3.007881 1.044103
## [6,]     5 -179.1183 385.8676 5.792943 1.553832 0.7414713 3.007881 1.553832
## [7,]     6 -198.5710 424.7730 4.177607 2.077089 0.7414713 4.177607 1.044103
## [8,]     7 -166.9345 366.1052 5.792943 2.077089 0.7414713 3.007881 1.044103
##           rho2
## [1,] 0.8898607
## [2,] 0.7153161
## [3,] 0.8963446
## [4,] 0.7239186
## [5,] 0.7294510
## [6,] 0.7239186
## [7,] 0.7239186
## [8,] 0.7239186
```

The model selection should select model 4 (μ and σ change) as having the lowest BIC.

This is with the default sensitivity parameter $K = 2$, which comes from the definition of the BIC $= -Kn\log(L) + k\log(n)$ (see summary in section 2). If we lower this value, a simpler model is more likely to be selected:

```
GetModels(x,t,BB[1], tau=FALSE, K=0.5)
```

```
##      Model        LL      bic      mu1       s1      rho1      mu2       s2
## [1,]     0 -192.7610 110.1960 4.177607 2.079177 0.8898607 4.177607 2.079177
## [2,]     1 -178.8822 112.4669 5.792943 1.553832 0.7153161 3.007881 1.553832
## [3,]     2 -195.8923 120.9720 4.177607 2.077089 0.8963446 4.177607 1.044103
## [4,]     3 -197.4125 121.7321 4.177607 2.079177 0.7414713 4.177607 2.079177
## [5,]     4 -166.9409 111.1015 5.792943 2.077089 0.7294510 3.007881 1.044103
## [6,]     5 -179.1183 117.1902 5.792943 1.553832 0.7414713 3.007881 1.553832
## [7,]     6 -198.5710 126.9165 4.177607 2.077089 0.7414713 4.177607 1.044103
## [8,]     7 -166.9345 115.7035 5.792943 2.077089 0.7414713 3.007881 1.044103
##           rho2
## [1,] 0.8898607
## [2,] 0.7153161
## [3,] 0.8963446
## [4,] 0.7239186
## [5,] 0.7294510
## [6,] 0.7239186
## [7,] 0.7239186
## [8,] 0.7239186
```

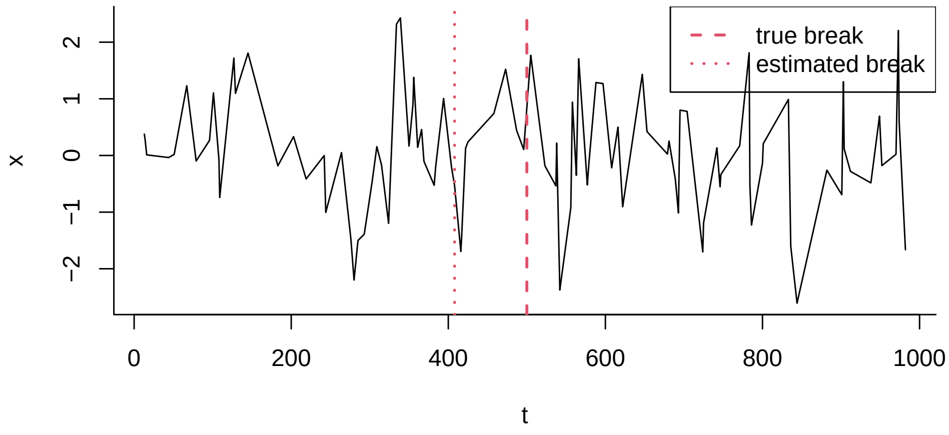
And if we increase it, more complex models are likely to be selected

```
GetModels(x,t,BB[1], tau=FALSE, K=5)
```

```
##      Model        LL       bic      mu1       s1      rho1      mu2       s2
## [1,]     0 -192.7610  977.6205 4.177607 2.079177 0.8898607 4.177607 2.079177
## [2,]     1 -178.8822  917.4367 5.792943 1.553832 0.7153161 3.007881 1.553832
## [3,]     2 -195.8923 1002.4871 4.177607 2.077089 0.8963446 4.177607 1.044103
## [4,]     3 -197.4125 1010.0885 4.177607 2.079177 0.7414713 4.177607 2.079177
## [5,]     4 -166.9409  862.3355 5.792943 2.077089 0.7294510 3.007881 1.044103
## [6,]     5 -179.1183  923.2225 5.792943 1.553832 0.7414713 3.007881 1.553832
## [7,]     6 -198.5710 1020.4860 4.177607 2.077089 0.7414713 4.177607 1.044103
## [8,]     7 -166.9345  866.9088 5.792943 2.077089 0.7414713 3.007881 1.044103
##           rho2
## [1,] 0.8898607
## [2,] 0.7153161
## [3,] 0.8963446
## [4,] 0.7239186
## [5,] 0.7294510
## [6,] 0.7239186
## [7,] 0.7239186
## [8,] 0.7239186
```

Below, we change only the autocorrelation parameter, which is quite a bit more difficult to detect by eye:

```
mu1 <- 0; mu2 <- 0
sigma1 <- 1; sigma2 <- 1
rho1 <- 0.9; rho2 <- 0.2
```



The model selection should select model 4 (only ρ changes):

```
GetModels(x,t,BB[1], tau=FALSE)


##      Model        LL      bic        mu1        s1      rho1         mu2
## [1,]     0 -140.6990 295.2135 0.03553484 1.0237827 0.5835633  0.03553484
## [2,]     1 -138.3282 299.6823 0.11595449 1.0214698 0.5799637 -0.02270007
## [3,]     2 -138.1012 299.2283 0.03553484 0.9951849 0.6168638  0.03553484
## [4,]     3 -130.7690 284.5638 0.03553484 1.0237827 0.8967592  0.03553484
## [5,]     4 -138.0851 303.8012 0.11595449 0.9951849 0.6142914 -0.02270007
## [6,]     5 -130.7364 289.1039 0.11595449 1.0214698 0.8967592 -0.02270007
## [7,]     6 -130.7627 289.1564 0.03553484 0.9951849 0.8967592  0.03553484
## [8,]     7 -130.7333 293.7027 0.11595449 0.9951849 0.8967592 -0.02270007
##            s2      rho2
## [1,] 1.023783 0.5835633
## [2,] 1.021470 0.5799637
## [3,] 1.048730 0.6168638
## [4,] 1.023783 0.3068741
## [5,] 1.048730 0.6142914
## [6,] 1.021470 0.3068741
## [7,] 1.048730 0.3068741
## [8,] 1.048730 0.3068741
```

3.4 Applying the window sweep

The main wrapper function for the complete analysis is `WindowSweep`. We select a windowsize and sensitivity parameter K and sweep analysis windows across the entire time series:

```
  Simp.ws <- WindowSweep(Simp.VT, "V*cos(Theta)", windowsize=50, progress=FALSE, K=2)
```

Note that the second argument of the function is a character string within which any function of the columns of the VT table can be analyzed as a response.

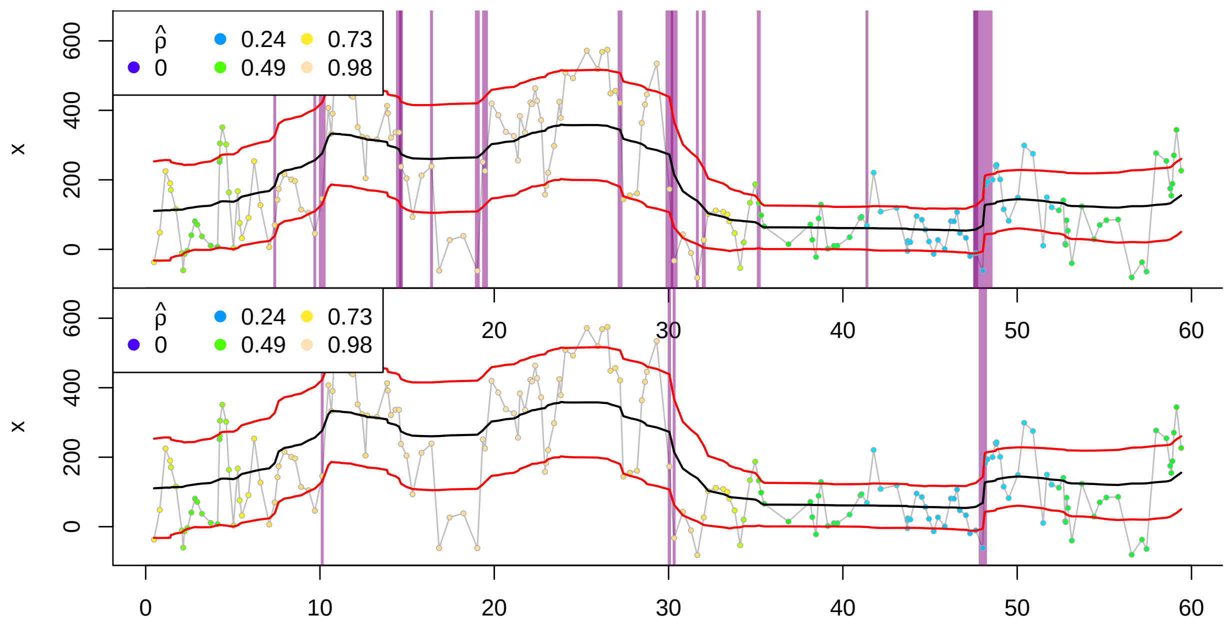
The key portion of the output of this function is the "windowsweep" data frame, which contains the proposed break (last column), the parameters to the left and right of the break, and the selected model:

```
head(Simp.ws$ws)


##   Model        LL      bic      mu1       s1      rho1      mu2       s2
## 1     1 -288.5915 596.8421 110.5384 142.6230 0.6908193 348.3828 142.6230
## 2     1 -288.2855 596.2301 112.2943 145.3151 0.7267682 329.0768 145.3151
## 3     1 -287.9583 595.5757 114.5005 146.0369 0.7486855 323.4264 146.0369
## 4     1 -289.7589 599.1769 111.6543 142.8630 0.5500623 331.5079 142.8630
## 5     1 -285.2421 590.1433 117.9271 111.5537 0.3900876 427.4309 111.5537
## 6     1 -285.2849 590.2289 116.4078 111.3103 0.3851086 426.5288 111.3103
##        rho2 Break.bb.time
## 1 0.6908193         7.625
## 2 0.7267682         7.405
## 3 0.7486855         7.405
## 4 0.5500623         7.565
## 5 0.3900876        10.130
## 6 0.3851086        10.130
```

Note that in this example, the first 4 windows detect no changes in the parameter values (Model 0) so the values are the same to the left and to the right of each changepoint. (A minor note: the `rho1` and `rho2` columns correspond to τ_1 and τ_2 - i.e. the time-scales.)

The following functions plot the output of the "smooth" summary, i.e. the summary in which all the windows are averaged to obtain the "smooth" model. In these plots, the vertical lines represent the significant change points, the width of the lines is proportional to the number of time that change point was selected, the black and red lines represent the mean and standard deviation estimate, and the colors reflect the autocorrelation time-scale (bluer colors have smaller autocorrelation time scales).

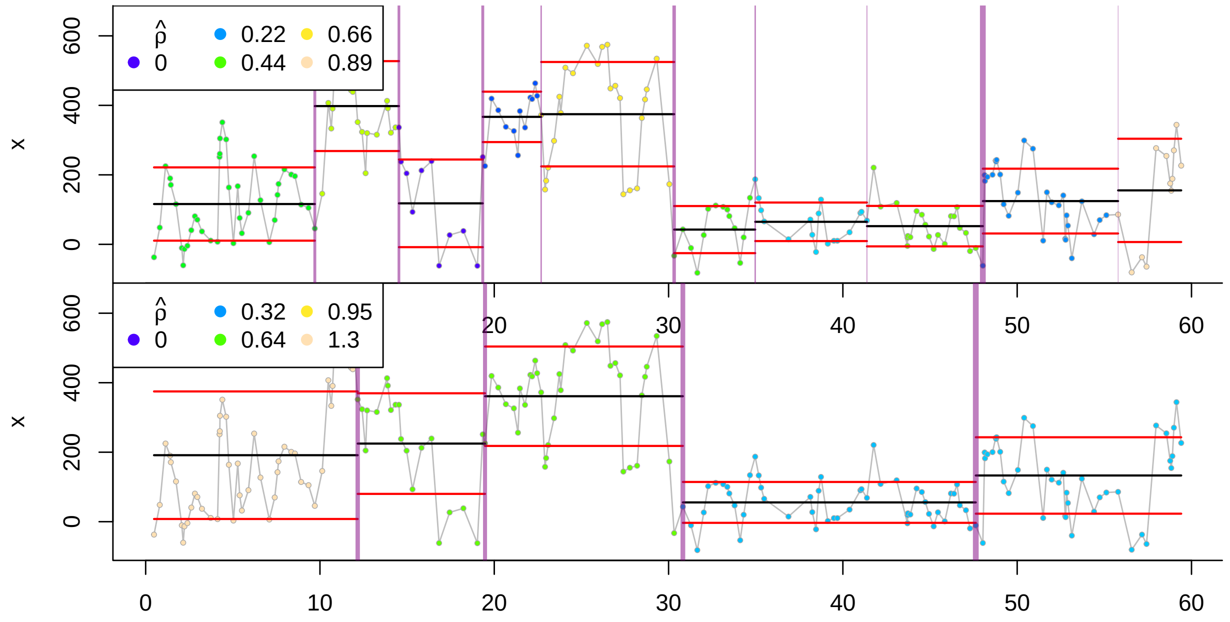
```
plot(Simp.ws, type="smooth")
plot(Simp.ws, type="smooth", threshold = 7)
```



The `threshold` parameter indicates how many of the windows that were swept over the data must have selected that changepoint for it to be considered significant. The changepoints selected (at that threshold) are almost exactly the ones that were encoded in the original simulation.

The "flat" analysis first selects changepoints that it deeps significant by clustering neighboring changepoints, and then estimates a homogeneous behavior between those changepoints. Note that by increasing the clusterwisth to 3, many of the more minor changepoints are filtered away and the resulting profile is fairly uniform.

```r
plot(Simp.ws, type="flat")
plot(Simp.ws, type="flat", clusterwidth=3)
```



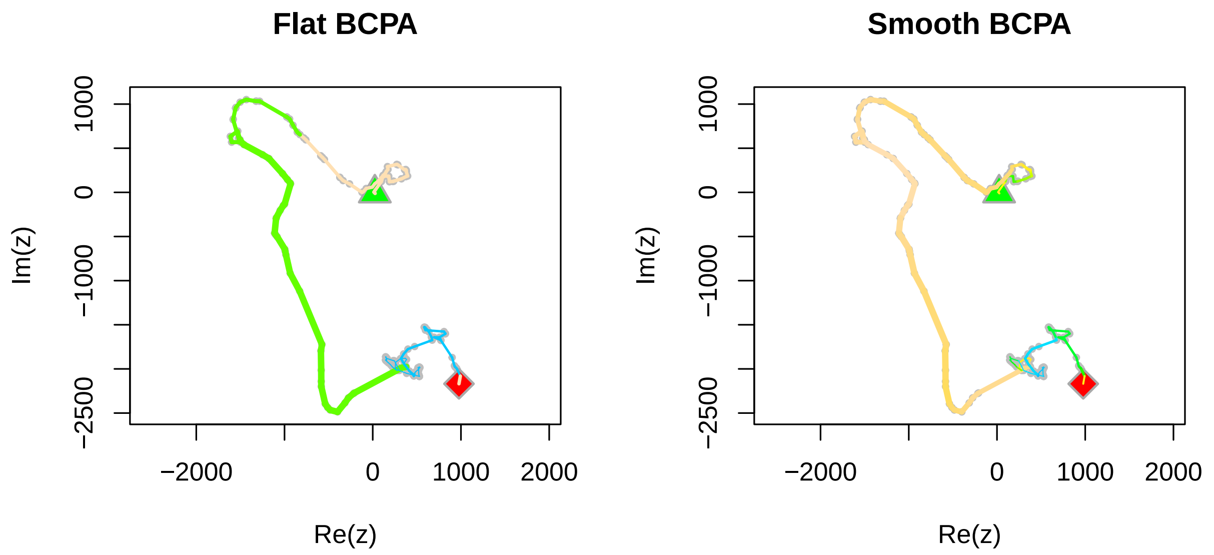
A summary of the flat changepoints can be obtained as follows:

```
ChangePointSummary(Simp.ws, clusterwidth=3)


## $breaks
##   X1   middle size modelmode middle.POSIX
## 1  1 12.15240   25         1      12.0800
## 2  2 19.45261   23         1      19.4050
## 3  3 30.79111   27         4      30.5700
## 4  4 47.64242   33         1      47.8275
##
## $phases
##          t.cut    mu.hat     s.hat   rho.hat       t0       t1  interval
## 1 (-0.52,12.2] 191.33698 183.54184 1.2710058 -0.52000 12.15240 12.672400
## 2  (12.2,19.5] 224.75311 144.77498 0.6654533 12.15240 19.45261  7.300209
## 3  (19.5,30.8] 361.06136 143.07515 0.6676718 19.45261 30.79111 11.338502
## 4  (30.8,47.6]  55.50484  58.81393 0.3713984 30.79111 47.64242 16.851313
## 5  (47.6,59.4] 132.98216 109.99431 0.3827311 47.64242 59.41000 11.767576
```

This summmary suggests five phases, with phases 2 and 3 consisting of a much higher velocity and longer time-scale movement than in the other phases.

The results of the BCPA can also be visualized with a so-called "path plot":

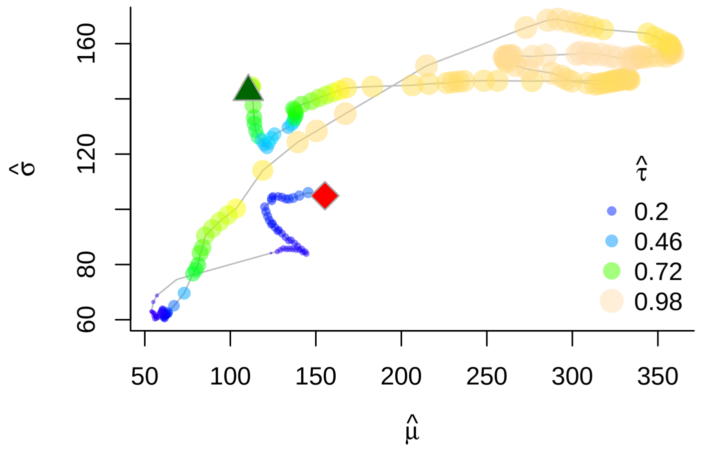
```r
PathPlot(Simp, Simp.ws, type="flat", clusterwidth = 3, main="Flat BCPA")
PathPlot(Simp, Simp.ws, type="smooth", main="Smooth BCPA")
```



The width of the line is proportional to the mean speed, while the colors correspond to the time-scales in the plots above. Both of thse plots clearly separate the period of faster, more directed movement

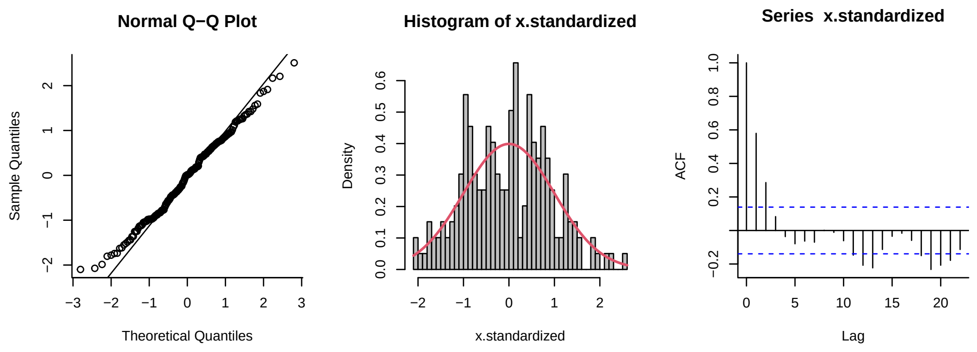
An additional, potentially interesting visualization of the analysis is the “phase plot”, which illustrates how the three parameters change with respect to each other:

```r
PhasePlot(Simp.ws, type="smooth", clusterwidth = 3)
```



Finally, it is important to assess the assumptions of the BCPA using diagnostic plots. In particular, to assess whether the standardized residuals of the final model are indeed distributed roughly as $\mathcal{N}(0,1)$ random variables

```
DiagPlot(Simp.ws)
```



This plot illustrates the qq-norm plot, the histogram and the auto-correlation function of the standardized data ($Z_i = X_i - \widehat{\mu}(T_i)/\widehat{\sigma}(T_i)$). Overall the results seem to satisfy the assumptions of normality.

4 Conclusions

We hope this package will facilitate analysis of complex behavioral movement data. However, it should be stressed that this is perhaps first and foremost an exploratory tool. Its strength is that it can distill complex information into some tabulated and visual summaries that outline underlying structures. It is also relatively fast - a long data set with tens of thousands of locations takes well under a minute to analyze on most machines.

That said, this tool merely *describes* and does not *explain* complex behavioral profiles. The BCPA can be used to propose some appropriate movement models or behavioral hypotheses for further testing. Alternatively, an explanatory or predictive analysis of behaviors with respect to covariates can be performed by extracting some biologically meaningful summary from the BCPA, and performing a post-hoc analysis to model the observed patterns with respect to covariates.

5 Acknowledgments

Gratitude is extended to F. Cagnacci, M. Panzacchi, B. van Moorter and colleagues at the Norwegian Institute of Nature Institute, who organized an animal movement analysis workshop and spearheaded a special issue of J. Animal Ecology (in progress), lighting the fuse to finally follow through on this package. C. Bracis and G. Passolt helped with the technical aspects of finalizing this package. Thanks, finally, to the many diverse colleagues and animal movement ecologists that tested earlier versions on actual data.

6 References

Gurarie, E., R. Andrews and K. Laidre. 2009. A novel method for identifying behavioural changes in animal movement data. Ecology Letters. 12: 395-408.

Gurarie, E., O. Ovaskainen. 2011. Characteristic spatial and temporal scales unify models of animal movement. American Naturalist. 178: 113-123.