smam: Statistical Modeling for Animal Movements

Jun Yan, Vladimir Pozdnyakov, Chaoran Hu

1. Introduction

This package is designed for animal movement analysis. We totally implemented three models, Brownian motion with measurement error [4], moving-resting process with embedded Brownian motion [1,2], and moving-resting-hunting process with embedded Brownian motion [3]. We also provided toolbox for seasonal analysis within this package, because wild animal usually has obvious seasonal behavior.

In this vignette, we play with a mountain lion GPS dataset and quantify the movement behavior of lion. First of all, let us take a look at the dataset.

```
library(smam)
head(f109)
```

```
##           t1   dt..hr.     e1      n1
## 1 2009-01-18  1.000000 534145 4831369
## 2 2009-01-18  1.000000 534163 4831303
## 3 2009-01-18  1.016667 534155 4831291
## 4 2009-01-23 23.983333 533764 4830986
## 5 2009-01-24  7.983333 534450 4830236
## 6 2009-01-25  8.000000 533352 4831062
```

The first column, `t1`, tells us when the observation is collected. The second column, `dt..hr.`, stands for the difference of time between two obvservations. The last two column, `e1` and `n1`, record the location information. Within this dataset, the location is based on the Universal Transverse Mercator mapping system. In order to play with `smam`, we should transfer the data to the 'standart' format.

```
f109_std1 <- transfData(f109, dateFormat = '%Y-%m-%d', roundValue = 100)
head(f109_std1)
```

```
##         date   cumTime centerE centerN
## 1 2009-01-18  1.000000     0.0     0.0
## 2 2009-01-18  2.000000     0.0    -0.1
## 3 2009-01-18  3.016667     0.0    -0.1
## 4 2009-01-23 27.000000    -0.4    -0.4
## 5 2009-01-24 34.983333     0.3    -1.1
## 6 2009-01-25 42.983333    -0.8    -0.3
```

By the above function, `transfData`, we modify the original GPS data with 1. rename the column; 2. the second column become the cumulated time line; 3. the location information here is centered by the first observation and round to 100 meters. This data format is required for seasonal analysis.

We mention that even if you do not have the same format original dataset as us, you can still modify them yourself, and as long as the data has the correct format, the functions in this package can be applied properly. The data format for overall analysis is shown below.

```
f109_std2 <- f109_std1[, -1] ##simply deleted the first column date
head(f109_std2)
```

```
##     cumTime centerE centerN
## 1  1.000000     0.0     0.0
## 2  2.000000     0.0    -0.1
## 3  3.016667     0.0    -0.1
## 4 27.000000    -0.4    -0.4
## 5 34.983333     0.3    -1.1
## 6 42.983333    -0.8    -0.3
```

We summary the data format by the following table.

Data formats	f109_std1	f109_std2
Analysis types	Seasonal analysis	Overall analysis
R funtions	seasonFilter	
	fitBmme.seasonal	fitBmme
	fitMovRes.seasonal	fitMovRes
	fitMovResHan.seasonal	fitMovResHan; fitMovResHan.parallel

2. Overall analysis

In this section, we show how to play with `fitBmme`, `fitMovRes`, `fitMovResHan` and `fitMovResHan.parallel`. All of these functions are designed for analysis whole dataset. Note that the last function `fitMovResHan.parallel` is the parallel computation wrapper for `fitMovResHan`, which inproves the speed of code significantly. Because these processes are time consuming, we do not show the result in this section and the result is self-explained.

```
## Brownian motion with measurement error
fitBmme(f109_std2)

## Moving-resting process with embedded Brownian motion
fitMovRes(f109_std2, start = c(4, 0.4, 1), likelihood = "full")

## Moving-resting-hunting process with embedded Brownian motion
```

```
fitMovResHan(f109_std2, start = c(4, 0.4, 0.1, 1, 0.5))
fitMovResHan.parallel(f109_std2, start = c(4, 0.4, 0.1, 1, 0.5),
                      ## number of threads allocated for parallel computation
                      numThreads = 6)
```

3. Seasonal analysis

Most wild animal has seasonal behavior, especially in the field of movement. In order to apply seasonal analysis, the first step is subset the whole data with given season. In this section, we need to use `f109_std1` and we know the data is from 2009 to 2012. We are going to focus on the summer part, which in the mountain area is from June 1st to August 31th. The following code will return the summer part data to us, which is a list with each element is for one year.

```
f109_summer <- seasonFilter(f109_std1,
                            startDate = '06-01', endDate = '08-31')
lapply(f109_summer, head)
```

```
## [[1]]
##           date  cumTime centerE centerN
## 299 2009-06-01 2586.833     6.9    -7.4
## 300 2009-06-01 2594.833     4.4    -5.8
## 301 2009-06-01 2602.850     4.3    -5.7
## 302 2009-06-02 2610.867     3.3    -2.6
## 303 2009-06-02 2618.833     4.1    -1.9
## 304 2009-06-03 2634.833     6.8    -2.4
##
## [[2]]
##            date  cumTime centerE centerN
## 1137 2010-06-01 10266.62    12.3    -6.7
## 1138 2010-06-01 10274.62    12.1    -8.6
## 1139 2010-06-01 10282.62    12.3    -8.4
## 1140 2010-06-02 10298.60    12.1    -5.7
## 1141 2010-06-02 10306.60     7.4    -7.2
## 1142 2010-06-03 10322.62    12.1    -4.8
##
## [[3]]
##            date  cumTime centerE centerN
## 1894 2011-06-01 17377.18    10.8    -1.8
## 1895 2011-06-01 17386.20    10.9    -1.6
## 1896 2011-06-02 17405.18    10.7    -1.8
## 1897 2011-06-03 17410.18    10.9    -1.7
## 1898 2011-06-03 17415.22    10.4    -1.9
```

```
## 1899 2011-06-03 17425.20    10.4    -2.0
##
## [[4]]
##            date  cumTime centerE centerN
## 3396 2012-06-01 25594.17     5.6   -12.0
## 3397 2012-06-01 25604.15     4.8   -12.3
## 3398 2012-06-01 25609.15     5.6   -13.6
## 3399 2012-06-01 25613.15     5.5   -13.9
## 3400 2012-06-02 25623.15     6.5   -14.2
## 3401 2012-06-02 25628.15     6.4   -14.3
```

Now, we can apply `fitBmme.seasonal`, `fitMovRes.seasonal`, `fitMovResHan.seasonal` on `f109_summer`. Because of the same result as before, we do not show the result here. For details of result, please read the document for each function.

```
## Brownian motion with measurement error
fitBmme.seasonal(f109_summer)

## Moving-resting process with embedded Brownian motion
fitMovRes.seasonal(f109_summer, start = c(4, 0.4, 1), likelihood = "full")

## Moving-resting-hunting process with embedded Brownian motion
fitMovResHan.seasonal(f109_summer, start = c(4, 0.4, 0.1, 1, 0.5),
                      ## number of threads allocated for parallel computation
                      numThreads = 6)
```

4. Simulation

All models in the package, `smam`, can be simulated with corresponding functions, `rbmme`, `rMovRes`, and `rMovResHan`. The simulation result is based on a given time gird.

```
## create the time grid for sampling
tgrid <- seq(0, 10, length = 1001)

## simulation
simul_bmme <- rbmme(tgrid, sigma = 1, delta = 1)
simul_movres <- rMovRes(tgrid, lamM = 4, lamR = 0.4,
                        sigma = 1, s0 = "m")
simul_MovResHan <- rMovResHan(tgrid, lamM = 4, lamR = 0.4, lamH = 0.1,
                              sigma = 1, p = 0.8, s0 = "m")

head(simul_bmme)
```

```
##      time
## [1,] 0.00 -1.5403038 -0.52321295
```

```
## [2,] 0.01  0.4903250  1.09756077
## [3,] 0.02 -0.4114490  0.36288233
## [4,] 0.03  1.8803439  2.75729203
## [5,] 0.04  0.4042052 -0.04610571
## [6,] 0.05  0.2478379 -1.49397405
```

```
head(simul_movres)
```

```
##   time            X1         X2
## 1 0.00  0.0000000000 0.00000000
## 2 0.01 -0.0060648035 0.02944376
## 3 0.02 -0.0008435964 0.06566980
## 4 0.03  0.1990677537 0.12829796
## 5 0.04  0.1990677537 0.12829796
## 6 0.05  0.1990677537 0.12829796
```

```
head(simul_MovResHan)
```

```
##   time          X1         X2
## 1 0.00  0.00000000 0.00000000
## 2 0.01  0.14442792 0.07877683
## 3 0.02  0.17014068 0.22891500
## 4 0.03  0.17986461 0.41031067
## 5 0.04 -0.04653456 0.52032006
## 6 0.05 -0.02369740 0.69668877
```

5. High-dimension case

We proudly mention that all our models and code also handle high-dimension case! We give a quick simulation example with moving-resting model here. Let's consider five dimension case.

```
tgrid <- seq(0, 10, length=500)
set.seed(123)
## make it irregularly spaced
tgrid <- sort(sample(tgrid, 30)) # change to 400 for a larger sample
dat <- rMovRes(tgrid, 1, 2, 25, "m", dim = 5)
head(dat)
```

```
##        time         X1         X2         X3        X4        X5
## 1 0.4008016   0.000000   0.000000   0.000000  0.000000  0.000000
## 2 0.4408818  -5.433932   2.020178  -1.145625  6.468001  4.813868
## 3 1.0020040   6.924395 -18.029748 -14.298473  7.857143 13.965453
## 4 1.3827655   4.681486 -17.584789 -36.322624  6.561494 19.209781
## 5 2.3847695 -24.486318 -28.398853 -20.825030 16.830854 34.135341
## 6 2.7254509 -36.430082  -3.740269 -20.915475 19.511083 61.443033
```

```r
fitMovRes(dat, start=c(1, 2, 25), likelihood = "full")
```

```
## $estimate
## [1]  1.566407  3.561696 24.998064
##
## $varest
##          [,1]      [,2]      [,3]
## [1,] 1.547370 1.4224375 1.3132608
## [2,] 1.422438 3.3937343 0.7881797
## [3,] 1.313261 0.7881797 5.1191437
##
## $loglik
## [1] -431.093
##
## $convergence
## [1] 0
##
## $likelihood
## [1] "full"
```

6. Wrapper functions

After finishing the first five sections above, you have already known everything you need to play with this package. So, you can stop here if you want. However, if you want to make your code more clear and concise, you can try our wrapper functions, `rSmam` and `fitSmam`. The first thing first, what is a wrapper function? I copy this definition from Professor Bryan Hanson, "The wrapper wraps, i.e. calls, another function that does the real work but provides a different or more convenient (or more convenient for a specific purpose) interface to it or specific syntax."

In this package, `rSmam` is based on `rbmme`, `rMovRes`, and `rMovResHan`. `fitSmam` is based on `fitBmme`, `fitMovRes`, `fitMovResHan`, `fitMovResHan.parallel`, `fitBmme.seasonal`, `fitMovRes.seasonal`, and `fitMovResHan.seasonal`. Let's take a look to the following short example and please note that these pairs of code give us the same result.

```r
## seasonal analysis with moving-resting model
fitMovRes.seasonal(f109_summer, start = c(4, 0.4, 1), likelihood = "full")
```

```
## $estimate
## [1] 0.4876013 0.1929312 0.7289605
##
## $varest
##              [,1]         [,2]         [,3]
## [1,] 0.003779778 3.543580e-04 1.208666e-03
```

```
## [2,] 0.000354358 1.063486e-04 8.356825e-05
## [3,] 0.001208666 8.356825e-05 6.301848e-04
##
## $loglik
## [1] -2871.15
##
## $convergence
## [1] 0
##
## $likelihood
## [1] "full"
```

```
fitSmam(f109_summer, model = "MovRes", start = c(4, 0.4, 1), likelihood = "full")
```

```
## $estimate
## [1] 0.4876013 0.1929312 0.7289605
##
## $varest
##              [,1]         [,2]         [,3]
## [1,] 0.003779778 3.543580e-04 1.208666e-03
## [2,] 0.000354358 1.063486e-04 8.356825e-05
## [3,] 0.001208666 8.356825e-05 6.301848e-04
##
## $loglik
## [1] -2871.15
##
## $convergence
## [1] 0
##
## $likelihood
## [1] "full"
```

```
## overall analysis with moving-resting model
fitMovRes(f109_std2, start = c(4, 0.4, 1), likelihood = "full")
```

```
## $estimate
## [1] 0.4322101 0.1546266 0.6101694
##
## $varest
##               [,1]         [,2]         [,3]
## [1,] 6.752680e-04 5.926061e-05 1.979397e-04
## [2,] 5.926061e-05 1.857507e-05 1.342072e-05
## [3,] 1.979397e-04 1.342072e-05 1.085230e-04
##
## $loglik
## [1] -8650.16
```

```
##
## $convergence
## [1] 0
##
## $likelihood
## [1] "full"
```

```
fitSmam(f109_std2, model = "MovRes", start = c(4, 0.4, 1), likelihood = "full")
```

```
## $estimate
## [1] 0.4322101 0.1546266 0.6101694
##
## $varest
##               [,1]         [,2]         [,3]
## [1,] 6.752680e-04 5.926061e-05 1.979397e-04
## [2,] 5.926061e-05 1.857507e-05 1.342072e-05
## [3,] 1.979397e-04 1.342072e-05 1.085230e-04
##
## $loglik
## [1] -8650.16
##
## $convergence
## [1] 0
##
## $likelihood
## [1] "full"
```

The parameters needed by `fitSmam` and `rSmam` depend on the selected model. We recommend you reading the original functions' documentation to clarify which parameters should be given.

References

[1] Yan, J., Chen, Y., Lawrence-Apfel, K., Ortega, I. M., Pozdnyakoc, V., Williams, S., and Meyer, T. (2014) A moving-resting process with an embedded Brownian motion for animal movements. Population Ecology. 56(2): 401–415. doi:10.1007/s10144-013-0428-8

[2] Pozdnyakov, V., Elbroch, L., Labarga, A., Meyer, T., and Yan, J. (2017) Discretely observed Brownian motion governed by telegraph process: estimation. Methodology and Computing in Applied Probability. doi:10.1007/s11009-017-9547-6.

[3] Pozdnyakov, V., Elbroch, L., Hu, C., Meyer, T., and Yan, J. (2018+) On estimation for Brownian motion governed by telegraph process with multiple off states. Under Review.

[4] Pozdnyakov, V. , Meyer, T. , Wang, Y. and Yan, J. (2014), On modeling animal movements using Brownian motion with measurement error. Ecology, 95: 247-253. doi:10.1890/13-0532.1