`moveHMM`
An R package for animal movement modelling

Michelot T., Langrock R., Patterson T., and Rexstad E.

August 29, 2015

Introduction

Animal movement data is growing rapidly, due to the substantial improvement of telemetry technologies. As a result, statistical methods used to analyse this data are brought to their computational limit.

Novel models have been developed in the last decade to reduce the computational cost of statistical inference in movement ecology. In particular, hidden Markov models are increasingly popular in this field, due to their flexibility, and to the efficient algorithms that they offer, see Patterson et al. (2009) and Langrock et al. (2012).

`moveHMM` is an R package which implements hidden Markov models (HMMs) for animal movement. A special attention was paid to performance, and the fitting algorithm is implemented in C++ to make it significantly faster.

The goal of this vignette is to give a global overview of the possibilities offered by the package, and to demonstrate its use on a detailed example.

1 Package features

In this section, we describe different features included in `moveHMM`. We describe the global structure of the package, and then describe in more detail the main functions required to fit a HMM to movement data. In particular, we introduce the different options that the functions offer, and explain how the functions' arguments should be chosen.

1.1 Structure

The package is articulated around two S3 classes : `moveData` and `moveHMM`. The first one is a data frame of the data, essentially gathering time series of the movement metrics of interest, namely the step lengths and turning angles, as well as the covariate values. A `moveHMM` object is a fitted model, which stores in particular the values of the MLE of the parameters.

In order to create a `moveData` object, the function `prepData` is called on the tracking data (track points coordinates). Then, the function `fitHMM` is called on the `moveData`, and returns a `moveHMM`.

Both classes can be used through their methods, e.g. `plot.moveData`, `decode.moveHMM`, `AIC.moveHMM`... All the functions are described in more detail in Section 1.3, and their use is explained on an example in Section 2.

Figure 1 illustrates the links between the main components of the package.

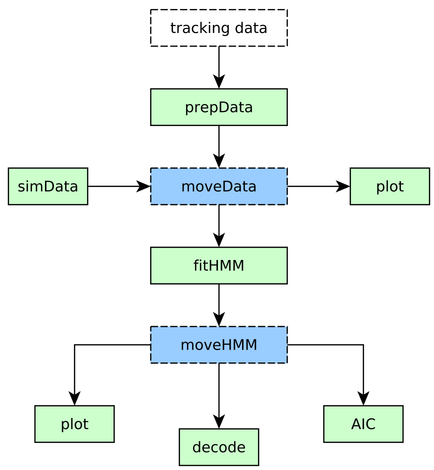

Figure 1: *Structure of the main components of the package. The blue boxes are S3 classes, and the green boxes are functions. The arrows indicate input and output of data.*

1.2 Model options

1.2.1 Distributions

Here is the list of distributions included, with the names they have in the package.

- Step length : Gamma (“gamma”), Weibull (“weibull”), exponential (“exp”), and log-normal (“lnorm”).
- Turning angle : Von Mises (“vm”), and wrapped-Cauchy (“wrpcauchy”). It is also possible to specify `angleDist="none"`, if the angles should not be modelled.

The parameters depend on the distribution used. The Gamma distribution expects the mean and standard deviation, and all other distributions expect the same parameters as the corresponding R density function, i.e.

Distribution	Parameters	
Gamma	mean	standard deviation
Weibull	shape	scale
Log-normal	log-mean	log-standard deviation
Exponential	rate	
Von Mises	mean	concentration
Wrapped Cauchy	mean	concentration

For the Gamma distribution, the link between the mean/standard deviation (expected by `fitHMM`) and scale/rate (expected by `dgamma`) is given by :

$$scale = \frac{(mean)^2}{(sd)^2}$$
$$rate = \frac{(mean)^2}{sd}$$

1.2.2 Zero-inflation

It is possible to inflate the step length distribution at 0, by specifying `zeroInflation=TRUE` in `fitHMM`. Zero-inflation can be defined as follows.

Let X_t be the random variable of the step length of the animal at time t. Let $\mathcal{D}(\theta)$ be the distribution of X_t, and $z \in [0,1]$ its inflation in 0. Then,

$$\begin{cases} X_t = 0 \text{ with probability } z \\ X_t \sim \mathcal{D}(\theta) \text{ with probability } 1-z \end{cases}$$

In the package, the distribution $\mathcal{D}$ will be one of the Gamma, Weibull, log-normal or exponential distributions.

1.2.3 Covariates

It is possible to model the state transition probabilities as functions of time-varying covariates. To do so, an additional parameter is included in the model, `beta`. This matrix contains the coefficients of the multinomial logistic regression which links the covariate values to the transition probabilities.

Let (C_t) be the state process, and $\gamma_{ij} = \Pr(C_{t+1} = j | C_t = i)$. Then,

$$\gamma_{ij}(t) = \frac{\exp\left(\beta_0^{(ij)} + \sum_k \beta_k^{(ij)} x_k(t)\right)}{1 + \exp\left(\beta_0^{(ij)} + \sum_k \beta_k^{(ij)} x_k(t)\right)}$$

where the x_k are the different covariates.

In addition, we consider the following constraint on the row sums of the transition probability matrix :

$$\forall t, \forall i, \sum_j \gamma_{ij}(t) = 1$$

This implies a constraint on the $\beta_k^{(ij)}$ coefficients. If n is the number of states of the HMM, then only $n(n-1)$ coefficients are required, for each covariate, to deduce the corresponding transition probabilities.

In practice, we chose to store coefficients for the non-diagonal transition probabilities. For example, for a 3-state HMM with two covariate, the matrix `beta` is,

$$\beta = \begin{pmatrix} \beta_0^{(12)} & \beta_0^{(13)} & \beta_0^{(21)} & \beta_0^{(23)} & \beta_0^{(31)} & \beta_0^{(32)} \\ \beta_1^{(12)} & \beta_1^{(13)} & \beta_1^{(21)} & \beta_1^{(23)} & \beta_1^{(31)} & \beta_1^{(32)} \\ \beta_2^{(12)} & \beta_2^{(13)} & \beta_2^{(21)} & \beta_2^{(23)} & \beta_2^{(31)} & \beta_2^{(32)} \end{pmatrix}$$

1.2.4 Stationarity

Ask Roland to write this.

1.3 Main functions

1.3.1 prepData

Most of the time, tracking data consists in time series of either easting-northing coordinates or longitude-latitude values. However, the data needed to use hidden Markov modelling are the time series of step lengths and turning angles.

The function `prepData` computes the steps/angles from the coordinates. As an input, this function takes an R data frame with mandatory column names "x" (either easting or longitude) and "y" (either northing or latitude). If several animals were observed, there should also be a colum "ID" which identifies the animal being observed. If there is no "ID" column, all observations will be considered to concern a single animal. All additional columns are considered as covariates.

In addition to the data frame, `prepData` takes an argument `type`, which can either be "GCD" (default) or "euclidean". The former indicates that the coordinates are longitude-latitude values, and the latter that they are easting-northing values. This option is used in the computation of the step lengths.

To do so, `prepData` calls the function `spDistN1`, from the package `sp`. The step lengths are in the metrics of the input if easting/northing are provided, and in kilometres if longitude/latitude are provided.

prepData outputs a data frame, with the same columns as the input, plus columns "step" and "angle". This object is of the class moveData, and can be plotted using the generic method plot.

1.3.2 fitHMM

To an object moveData can then be fitted an HMM, using the function fitHMM. The list of its arguments is detailed in the documentation.

The maximum likelihood estimation is carried out using the R function nlm.

This function outputs a list of information about the model. Most elements of that list are only meant to be used by the moveHMM methods (see Section 1.3.3), but a few can be informative *per se* :

- mle contains the estimates of the parameters of the model;
- mod contains the output of the optimization function nlm, including mod$minimum (minimum of the negative log-likelihood) and mod$hessian, the hessian of the negative log-likelihood function at its minimum.
- states contains the sequence of most probable states, as computed by the Viterbi algorithm (Zucchini and MacDonald, 2009).

1.3.3 Classes methods

Methods (i.e. class functions) are available for both moveData and moveHMM objects, to operate on them. Here is a list of them ; for details on the options, see the documentation, and for an example of their use, see Section 2.

- plot.moveData plots a few graphs to illustrate the data : a map of each animal's track, time series of the steps and angles, histograms of the steps and angles.
- plot.moveHMM plots a few graphs to illustrate the fitted model : a map of each animal's track, colored by states, plots of the estimated density functions, plots of the transition probabilities as functions of the covariates.
- AIC.moveHMM returns the AIC of the fitted model.
- pseudoRes.moveHMM computes the pseudo-residuals of the model.
- stateProbs.moveHMM computes the state probabilities for each observation.
- decode.moveHMM wraps pseudoRes and stateProbs.
- confIntervals.moveHMM computes the confidence intervals for the step length distribution parameters and for the regression coefficients of the transition probabilities.
- deltaMethod.moveHMM computes the confidence intervals for the turning angle distribution parameters, using the delta method.

1.3.4 simData

The function `simData` simulates movement data from an HMM, given its parameters. The returned object is of the class `moveData`, and can then be visualized using `plot.moveData`, or fitted using `fitHMM`.

The arguments of `simData` are detailed in the documentation.

2 Application

In this section, we illustrate the possibilities of the package on a real data example. We use the data from Morales et al. (2004), collected on four elks in Canada.

2.1 Movement data

In order to be preprocessed and fitted, the data needs to have the correct format. It needs to be a `data.frame`, with two mandatory columns :

- Easting or longitude (default name : `x`)
- Northing or latitude (default name : `y`)

It is possible to have a column "ID", which contains the identifiers of the observed animals. If no column named "ID" is provided, all the observations will be considered to concern a single animal.

Additional columns are considered as covariates. Note that covariates need to have numerical values.

2.1.1 Load and format the tracking data

The data is available from the following URL :

http://www.esapubs.org/archive/ecol/E085/072/elk_data.txt

We load the relevant rows and columns into a data frame using the `read.table` command.

```
trackData <- read.table(
    "http://www.esapubs.org/archive/ecol/E085/072/elk_data.txt",
    sep="\t",header=TRUE)[1:735,c(1,2,3)]
```

The dataframe `trackData` now has three columns : "Individual", "Easting", and "Northing".

```
> head(trackData)
  Individual Easting Northing
1    elk-115  769928  4992847
2    elk-115  766875  4997444
3    elk-115  765949  4998516
4    elk-115  765938  4998276
5    elk-115  766275  4998005
6    elk-115  766368  4998051
```

The animals' identifiers need to be stored in a column named "ID", before we can preprocess the data. Thus, we modify it accordingly.

```
colnames(trackData)[1] <- "ID"
```

2.1.2 Use `prepData`

Now that the data has the right format, it is possible to call the preprocessing function `prepData`. We choose the arguments carefully :

- `type` specifies whether the coordinates are easting/northing or longitude/latitude values. The latter is the default, so we need to call the function with the argument `type="euclidean"`, to indicate that we want to use the euclidean distance.
- `coordNames` are the names of the coordinates in the input dataframe. The default is "x" and "y", so we need to call the function with the argument `coordNames=c("Easting","Northing")`.

Eventually, the call to the function is,

```
data <- prepData(trackData,type="euclidean",coordNames=c("Easting",
    "Northing"))
```

The step lengths and turning angles are computed, and the returned object is a data frame.

```
> head(data)
       ID      step      angle      x       y
1 elk-115        NA         NA 769928 4992847
2 elk-115 5518.4434  0.1262112 766875 4997444
3 elk-115 1416.5663  2.3832412 765949 4998516
4 elk-115  239.7525  0.9385238 765938 4998276
5 elk-115  432.7600  1.1375066 766275 4998005
6 elk-115  103.7545 -0.9687435 766368 4998051
```

Note that the coordinate have been renamed "x" and "y". This makes the processing of the data simpler.

If the data contains covariates, which have missing values, those are replaced by the closer non-missing value (by default, the previous one if it is available). This is arbitrary, and might result in misinterpretation of the data.

2.1.3 Use `plot.moveData`

Once the data has been preprocessed, it is possible to plot it, using the generic method `plot`. It displays maps of the animals' tracks, times series of the steps and angles, and histograms of the steps and angles. A few plotting options are available, and described in the documentation.

We want to plot all animals' tracks on a single map, so we call :

```
plot(data,compact=T)
```

The resulting map, and the steps and angles graphs for the first animal, are displayed in Figure 2.

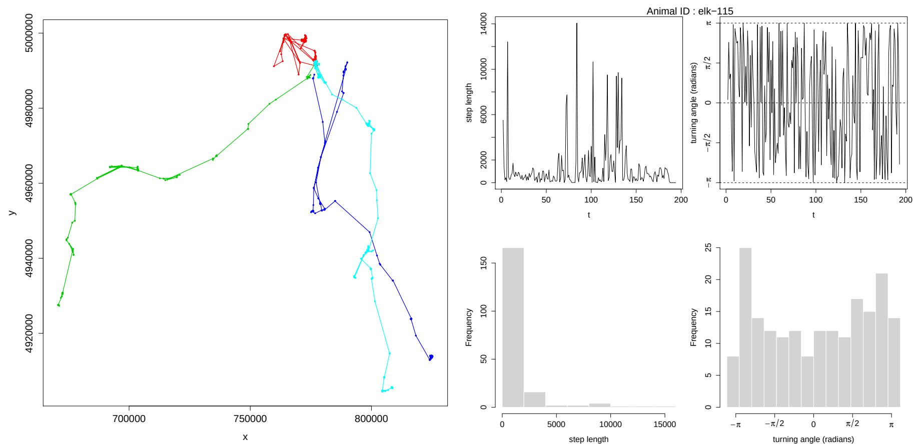

Figure 2: *Map of the animals' tracks (left) – each color represents an animal. Graphs of the step lengths and turning angles for one individual (right).*

The time series of the step lengths is one way to check the data for outliers.

2.2 Fitting the model

The function `fitHMM` is used to fit an HMM to the data. Its arguments are described in the documentation. Here are a few choices we make :

- `nbStates=2`, i.e. we fit a 2-state HMM to the data;
- `beta0=NULL` and `delta0=NULL`, i.e. `beta0` and `delta0` will take their default values.
- `formula= 1`, i.e. covariate free model;
- `stepDist="gamma"`, to model the step lengths with the Gamma distribution;
- `angleDist="vm"`, to model the turning angles with the Von Mises distribution;
- `angleMean=NULL`, because we want to estimate the mean of the angle distribution;
- `zeroInflation=TRUE`, because some step lengths are zero;
- `stationary=TRUE`, i.e. we consider the initial distribution as the stationary distribution. This is possible because there are no covariates in the model.

We also need to specify initial values for the parameters, which are used by the optimization function. Note that this choice is crucial, and that the algorithm might not find the global optimum of the likelihood function if the initial parameters are poorly chosen.

```
# initial parameters
mu0 <- c(100,1000) # mean of the steps
sigma0 <- c(100,1000) # sd of the steps
zeromass0 <- c(0.05,0.0001) # zero-mass of the steps
stepPar0 <- c(mu0,sigma0,zeromass0)
angleMean0 <- c(pi,0) # mean of the angles
kappa0 <- c(1,1) # concentration of the angles
anglePar0 <- c(angleMean0,kappa0)

# call to fitting function
m <- fitHMM(data=data,nbStates=2,stepPar0=stepPar0,
    anglePar0=anglePar0,zeroInflation=TRUE,stationary=TRUE)
```

The returned object, `m`, is of the class `moveHMM`. It can be printed, to get the MLE of the parameters.

```
> m
Value of the maximum log-likelihood : -6939.139

Step length parameters :
----------------------
mean
[1]  370.2011 3230.1156
sd
[1]  394.7855 4357.0133
zero-mass
[1] 2.010399e-03 3.301028e-09

Turning angle parameters :
------------------------
mean
[1] -3.0223613 -0.0391857
concentration
[1] 0.6029967 0.2028802

Transition probabilities parameters :
-----------------------------------
          [,1]      [,2]
[1,] -2.312046 -1.400165

Transition probability matrix :
-----------------------------
          [,1]       [,2]
[1,] 0.9098698 0.09013023
[2,] 0.1977899 0.80221012

Initial distribution :
--------------------
[1] 0.686961 0.313039
```

2.3 Using the model

Various methods are available for the class `moveHMM`, and here we explain how to use them on the example.

2.3.1 Plot the model

The fitted model can be plotted, using the generic method `plot`. A few graphical options are available, and listed in the documentation.

Here, we plot the map of the first animal's track, colored by states, and the fitted densities of the steps and angles for that animal (elk-115). To do so, we call :

```
plot(m,animals="elk-115")
```

2.3.2 Decode the model

2.3.3 Assess the model

References

Langrock R., King R., Matthiopoulos J., Thomas L., Fortin D., Morales J.M. (2012), "Flexible and practical modeling of animal telemetry data: hidden Markov models and extensions" *Ecology*, 93 (11), 2336–2342.

Morales, J.M., Haydon, D.T., Frair, J., Holsinger, K.E., Fryxell, J.M. (2004), "Extracting more out of relocation data : building movement models as mixtures of random walks", *Ecology*, 85 (9), 2436–2445.

Patterson T.A., Basson M., Bravington M.V., Gunn J.S. (2009), "Classifying movement behaviour in relation to environmental conditions using hidden Markov models" *Journal of Animal Ecology*, 78 (6), 1113–1123.

Zucchini, W. and MacDonald, I.L. (2009). *Hidden Markov Models for Time Series: An Introduction Using R.* Chapman & Hall (London).