moveHMM
An R package for animal movement modelling

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

August 20, 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 (2009) and Langrock (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 1.2, and their use is explained on an example in Section 2.

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

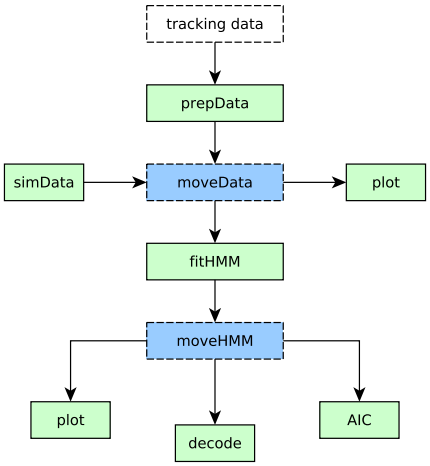

Figure 1: *Structure of the main components of the package. The blue boxes are S3 classes, and the green boxes are functions. The arrows indicate input and output of data.*

1.2 Main functions

1.2.1 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.

`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`.

1.2.2 fitHMM

To an object `moveData` can then be fitted an HMM, using the function `fitHMM`. This function takes the following arguments :

- `data` : a `moveData` object.
- `nbStates` : the number of states of the HMM to be fitted.
- `stepPar0` : initial values for the step length distribution parameters. Note that they should be in a vector, in the order expected by the density function of the used distribution. For an example, see Section 2.
- `anglePar0` : initial values for the turning angle distribution parameters. Same remark as for `stepPar0`.
- `beta0` : initial values for the regression coefficients of the transition probabilities. If set to `NULL`, a default value is chosen such that the transition probabilities do not depend on the covariates, and diagonal elements of the transition probability matrix are dominant.
- `delta0` : initial value for the initial distribution of the HMM. If set to `NULL`, the default value is chosen to be `rep(1/nbStates,nbStates)`.
- `formula` : the formula that determines the relationship between the transition probabilities and the covariates. Default is `1`, i.e. covariate-free model.
- `stepDist` : distribution for the step lengths. Possible values are "gamma" (default), "weibull", "lnorm", and "exp".
- `angleDist` : distribution for the turning angles. Possible values are "vm" (Von Mises distribution, default), "wrpcauchy", and "none" (only steps should be modelled).
- `angleMean` : mean parameter of the angle distribution. If set to `NULL`, the angle mean is estimated.
- `zeroInflation` : boolean value, `TRUE` if zero-inflation should be included in the model, `FALSE` otherwise (default).

- `stationary` : boolean value, `TRUE` if the initial distribution is considered to be the stationary distribution, `FALSE` otherwise (default).
- `verbose` : level of printing of the optimization function. Can be `0` (no printing), `1` (first and last iteration of the optimization are printed), or `2` (all iterations are printed).

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 1.2.3), 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.

1.2.3 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.

1.2.4 simData

2 Application

We take the user through a detailed example. I think that we will include a (relatively small) real data set in `data/`, which will serve in this example.

2.1 Movement data

We describe the format of the data that the user needs to input :

- R data frame;
- regular time intervals;
- mandatory column names : "ID", "x", "y", ...;
- additional columns are considered as covariates;
- warn the user about missing covariate values;
- warn the user about outliers.

We explain how to use the function `prepData` to transform the tracking data into movement data.

We explain how to use `plot.moveData`, which graphical options are available, and we display the output.

2.2 Fitting the model

We go through all the options of `fitHMM`, and demonstrate its use on the real data example.

Here we can warn the user about the choice of the initial values.

2.3 Using the model

We need to mention :

- Plotting : describe the function `plot.moveHMM` and the graphical options, and display all the plots;
- Decoding : describe the decoding functions, such as `viterbi`, `stateProbs`, and `pseudoRes`, and apply them to the data;
- Assessing : we don't really have the functions yet, but we will need to explain how to obtain confidence intervals, and how to plot them. We might want to mention the simulation function as an assessing tool.

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.

Patterson T.A., Basson M., Bravington M.V., Gunn J.S. (2012), "Classifying movement behaviour in relation to environmental conditions using hidden Markov models" *Journal of Animal Ecology*, 78 (6), 1113-1123.