`moveHMM`

An R package for the analysis of animal movement data

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

June 3, 2018

Contents

1 Introduction

The analysis of animal movement data has become increasingly important in terrestrial and marine ecology. Improvements in telemetry technology have resulted in an explosion in the volume of high precision data being collected. As a result, there are two challenges which researchers collecting these data regularly face: (1) data volume and (2) employing statistical methods which can accommodate some of the specific features of movement data (Patterson et al., 2009).

A substantial part of the literature on statistical modelling of animal movement data has focused on the intuitive approach of decomposing movement time series into distinct behavioural modes (a.k.a. bouts, states), via the use of so-called state-switching models. These approaches typically involve assuming movements of animals to be driven by stochastically evolving states, such as a slow moving state, which may be indicative of resting or foraging, versus faster movement states which might indicate transits between foraging patches. Associated with changes in movement speeds are changes to the distribution of directional changes in the movement (known as the turning angle — see further description below).

Bayesian methods which employ MCMC approaches have become very popular tools for the analysis of movement data using state-switching models (e.g. Jonsen et al., 2005; Morales et al., 2004). Typically, these have been implemented using WinBUGS (although see McClintock et al., 2012). While these models are relatively straightforward to build and hence fit in WinBUGS, the estimation can be painfully slow due to slow mixing of the MCMC samplers.

However, for an important subset of movement data, namely highly accurate position data (e.g. from GPS) — and more generally all time series of locations where the measurement error is negligible relative to the scale of the movement — the task of statistical classification of behaviour can be done much more efficiently using hidden Markov models (HMMs) and associated frequentist inferential tools. HMMs are increasingly popular in this field, due to their flexibility and to the associated very efficient recursive algorithms available for conducting statistical inference (Patterson et al., 2009; Langrock et al., 2012). The crucial requirements on movement data in order for HMMs to be suitable are that measurement error in positions is negligible and that there is a regular sampling unit (e.g. one positional observation per hour, or per dive, or any other meaningful unit).

`moveHMM` is an R package which implements HMMs and associated tools for state decoding, model selection etc. specifically tailored to animal movement modelling. Particular attention was paid to computational efficiency with the fitting algorithm implemented in C++. The high computational speed makes it feasible to analyze very large data sets — e.g. tens of thousands of positions collected for each of a dozen individual animals — on standard desktop PCs. The package also allows users to incorporate covariate data into their models, which is particularly useful when inferring the drivers of changes in behaviour.

Our hope is that the `moveHMM` package will provide users who collect movement data with an interface to sophisticated and adequate methods for a statistical analysis of their data. The package is structured so as to allow the users to prepare their data for analysis, fit a variety of HMMs to their data and perform diagnostics on these fitted models.

The package is presented in Michelot et al. (2016), where its use is illustrated on simulated movement data of wild haggises.

In this vignette, we briefly introduce HMMs in the context of animal movement. We then provide a detailed example of a typical use of the package (preprocessing movement data, fitting an HMM to the data, and analyzing the fitted model). Finally, we describe more technically the structure of the package and its main functions.

2 A quick overview of the HMM approach to modelling animal movement

For full details of the HMM approach in the context of animal movement modelling, the user should refer to the relevant primary publications (see, e.g., Patterson et al., 2009, Langrock et al., 2012, Zucchini et al., 2016). Here, we briefly highlight the essential features of the HMM approach.

The standard HMM approach to model an individual animal's movement considers bivariate time series comprising the step length and the turning angle at each time point (see illustration in Figure 1). The associated locations need to be *sampled at equally spaced points in time* (though missing data on an otherwise regular grid can easily be handled) and are assumed to be *observed with no or only negligible error*. The `moveHMM` package is restricted to such discrete time data and involves the assumption that the locations are observed with zero, or at least negligible error.

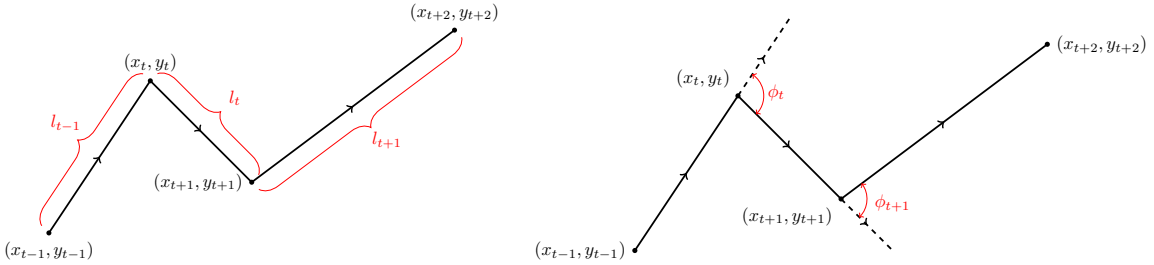

Figure 1: *Illustration of step lengths and turning angles*

At each time point, the parameters of the step length distribution (e.g., a gamma distribution) and the parameters of the turning angle distribution (e.g., a von Mises distribution) are determined by an underlying unobserved state. There are finitely many states which provide rough classifications of the movement (e.g. more active vs. less active), often interpreted as proxies for the animal's behavioural states (e.g. transiting vs. foraging). The sequence of states is assumed to be generated by a Markov chain, usually with a tendency of remaining in a state for some time before switching to another state. The corresponding state transition probabilities, as well as the parameters characterising the state-dependent distributions, are model parameters to be estimated. The number of states (movement modes) is unknown and has to be specified by the user. Typically with movement data one assumes a low number of states (say ≤ 4). For example, an animal may be foraging (low speed, high rates of turning) and transiting (high speeds, low rates of turning). Biologically interesting inference often involves modelling the state transition probabilities as functions of environmental covariates.

For a given set of model parameters, the likelihood of the data can be calculated using a recursive algorithm (the forward algorithm, cf. Patterson et al., 2016), which in a very effective way considers all possible state sequences that might have given rise to the observed time series. This makes numerical maximization of the (log-)likelihood, and hence maximum likelihood estimation, feasible in most cases. Having estimated the model parameters (and examined useful diagnostics of model fit), the user can estimate the most likely sequence of behavioural states. We encourage the user to consult a good primary text such as Zucchini et al. (2016) to familiarize themselves with the technical details of hidden Markov models.

3 Example application

Before we provide a detailed description of the various features of the `moveHMM` package in the subsequent section, we illustrate a typical HMM-based analysis of movement data using the main functions of the package, via an example. We use the data from Morales et al. (2004), collected on four elk in Canada.

3.1 Movement data

The input data need to have the correct format for subsequent processing and analysis. The data need to be provided as 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 belong to a single animal. Additional columns are considered as covariates. Note that, within this package, covariates need to have numerical values (rather than e.g. character values).

3.1.1 Load and format the tracking data

The elk data considered in Morales et al. (2004) are loaded with the package, as the data frame `elk_data`.

The data frame has four columns: "ID", "Easting", "Northing", and "dist_water". The last one is the distance of the animal to water, which for illustration purposes we want to include in the model as a covariate.

```
head(elk_data)
```

```
##        ID Easting Northing dist_water
## 1 elk-115  769928  4992847     200.00
## 2 elk-115  766875  4997444     600.52
## 3 elk-115  765949  4998516     561.81
## 4 elk-115  765938  4998276     550.00
## 5 elk-115  766275  4998005     302.08
## 6 elk-115  766368  4998051     213.60
```

The easting and northing values are expressed in meters in the data, and we decide that we want to deal with distances in kilometers for the step lengths. To achieve this, we transform the coordinates into meters.

```
elk_data$Easting <- elk_data$Easting/1000
elk_data$Northing <- elk_data$Northing/1000
```

As a result, this is what the data look like:

```
head(elk_data)
```

```
##        ID Easting Northing dist_water
## 1 elk-115 769.928 4992.847     200.00
## 2 elk-115 766.875 4997.444     600.52
## 3 elk-115 765.949 4998.516     561.81
## 4 elk-115 765.938 4998.276     550.00
## 5 elk-115 766.275 4998.005     302.08
## 6 elk-115 766.368 4998.051     213.60
```

3.1.2 Use `prepData`

The data are in the proper format, and can be processed using `prepData` to compute step lengths and angles. We choose the arguments carefully:

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

The call to the function for these data is,

```
data <- prepData(elk_data,type="UTM",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 dist_water
## 1 elk-115  5.5184434         NA 769.928 4992.847     200.00
## 2 elk-115  1.4165663  0.1262112 766.875 4997.444     600.52
## 3 elk-115  0.2397525  2.3832412 765.949 4998.516     561.81
## 4 elk-115  0.4327600  0.9385238 765.938 4998.276     550.00
## 5 elk-115  0.1037545  1.1375066 766.275 4998.005     302.08
## 6 elk-115 12.4164659 -0.9687435 766.368 4998.051     213.60
```

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

If the data set contains covariates which have missing values, then those are imputed using the closest non-missing value, by default the previous one if it is available. This is arbitrary and might not be appropriate in all situations.

It is also possible to print summary information about the data, using the function `summary`, e.g.

```
summary(data)
```

```
## Movement data for 4 tracks:
## elk-115 -- 194 observations
## elk-163 -- 159 observations
## elk-287 -- 164 observations
## elk-363 -- 218 observations
##
## Covariate(s):
##  dist_water
##       Min.       25%    Median      Mean       75%      Max.
##     0.0000  213.6000  477.6200  773.6457 1169.4950 3781.0400
```

3.1.3 Use `plot.moveData`

Once the data have been preprocessed, they can be plotted using the generic function `plot`. This 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. These are described in the documentation. To plot all animals' tracks on a single map, we call:

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

The resulting map, and the steps and angles graphs for the first animal, are displayed in Figure 2 (we omit the graphs for the three other animals, which are also displayed when using the above command).

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

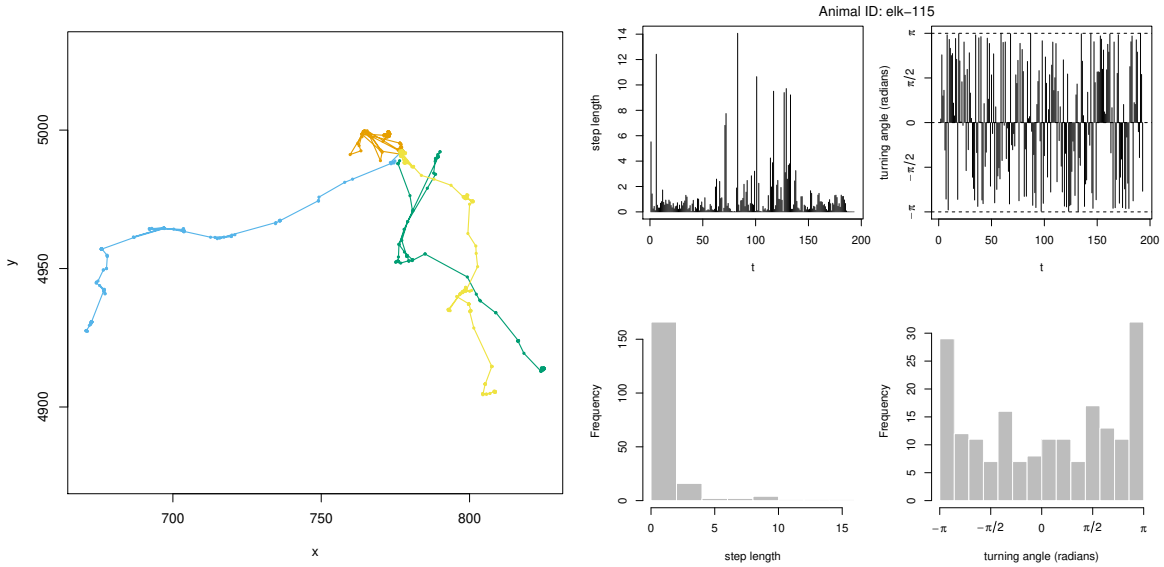

Figure 2: *Map of the animals' tracks (left) – each color represents an animal. Time series and histograms of the step lengths and turning angles for one individual, "elk-115" (right).*

3.2 Fitting the model

3.2.1 fitHMM

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. we use the default values for the initial values `beta0` and `delta0`;
- `formula=~dist_water`, i.e. the transition probabilities are functions of the covariate "dist_water";
- `stepDist="gamma"`, to model the step lengths with the gamma distribution (note that it is the default, so we do not need to explicitely specify it);
- `angleDist="vm"`, to model the turning angles with the von Mises distribution (default);
- `angleMean=NULL`, because we want to estimate the mean of the angle distribution (default);
- `stationary=FALSE`, as due to the covariates the process is not stationary (default).

We also need to specify initial values for the parameters of the state-dependent distributions, to be 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**. The initial parameters should be specified in two vectors, `stepPar0` (for the step distribution) and `anglePar0` (for the angle distribution). The necessary parameters of each distribution are detailed in Section 4.1.1.

Zero-inflation (as described in Section 4.1.2) must be included in the step length distribution if some steps are of length exactly zero (which is the case for the elk data). To do so, another parameter is added to the step distribution: its mass on zero.

Here, the initial values are chosen such that they correspond to the commonly observed pattern in two-state HMMs for animal movement data, with state 1 involving relatively short steps and many

turnings (hence the choice of a small initial value for the mean of the gamma step length distribution and an initial value of pi for the mean turning angle) and state 2 involving longer steps and fewer turnings (hence the choice of a larger initial value for the mean of the gamma step length distribution and an initial value of 0 for the mean turning angle).

For numerical stability, we decide to standardize the covariate values before fitting the model (explanations in Section 3.2.2).

```
## standardize covariate values
data$dist_water <-
    (data$dist_water-mean(data$dist_water))/sd(data$dist_water)

## initial parameters for gamma and von Mises distributions
mu0 <- c(0.1,1) # step mean (two parameters: one for each state)
sigma0 <- c(0.1,1) # step SD
zeromass0 <- c(0.1,0.05) # step zero-mass
stepPar0 <- c(mu0,sigma0,zeromass0)
angleMean0 <- c(pi,0) # angle mean
kappa0 <- c(1,1) # angle concentration
anglePar0 <- c(angleMean0,kappa0)

## call to fitting function
m <- fitHMM(data=data,nbStates=2,stepPar0=stepPar0,
            anglePar0=anglePar0,formula=~dist_water)
```

The returned object, `m`, is of the class `moveHMM`. It can be printed in order to obtain the maximum likelihood estimates of all model parameters.

```
m

## Value of the maximum log-likelihood: -1892.95
##
## Step length parameters:
## ----------------------
##                state 1      state 2
## mean       0.355004560 3.363565e+00
## sd         0.377971682 4.328807e+00
## zero-mass 0.001975657 6.433220e-09
##
## Turning angle parameters:
## ------------------------
##                   state 1   state 2
## mean          -2.9956557 0.1251057
## concentration  0.5997421 0.2284823
##
## Regression coeffs for the transition probabilities:
## --------------------------------------------------
##                1 -> 2     2 -> 1
## intercept  -2.0505161 -0.6902015
## dist_water -0.3598238  1.0778177
##
## Initial distribution:
## --------------------
## [1] 0.2483832 0.7516168
```

The argument `knownStates` of the function `fitHMM` makes it possible to set some values of the state process to fixed values, prior to fitting the model. This can be useful e.g. when the animal's behaviour

is known for some time points, but we discourage users to take advantage of this option to make the states match their expectations (instead of letting the data speak for themselves).

3.2.2 Dealing with numerical instability

As mentioned above, the numerical maximization routine might not identify the global maximum of the likelihood function, or even fail to converge altogether, for poorly chosen initial values of the parameters. In such a case, the optimization routine `nlm` might produce an error such as:

```
Error in nlm(nLogLike, wpar, nbStates, bounds, parSize, data, stepDist,  :
non-finite value supplied by 'nlm'
```

The best way to deal with such numerical problems is to test different sets of initial values, possibly chosen randomly. By comparing the resulting estimates for the different initial values used, one usually obtains a good feeling for any potential sensitivity of the numerical search to its chosen starting point. Note, however, that in any case there will usually be no certainty that the global maximum of the likelihood, i.e. the maximum likelihood estimate, has been identified.

During preliminary tests on the elk data, we noticed that, in this example, the numerical search is highly sensitive to the choice of the initial parameters `beta0`. This is due to the high values of the covariate: a small change in the associated regression coefficients can make a big difference in the likelihood function. In such cases, it is advisable to standardize the covariate values before fitting the model, for example by calling:

```
data$dist_water <-
    (data$dist_water-mean(data$dist_water))/sd(data$dist_water)
```

This allows for greater numerical stability, with the convergence of the fitting function depending less on the choice of initial values. The value of the maximum log-likelihood is not affected by the standardization of the covariate values, only the maximum likelihood estimate of `beta` is.

3.2.3 Confidence intervals

Confidence intervals for the model parameters can be computed with the function `CI`, passing as an argument the object created by `fitHMM`. It is possible to give the significance level of the desired confidence interval as an argument, e.g. 0.99 for 99% confidence intervals. By default, 95% confidence intervals are returned.

Below we show the 95% confidence intervals for the parameters of the 2-state model fitted to the elk data. `CI(m)$stepPar` corresponds to the bounds of the confidence intervals for the step parameters (`CI(m)$stepPar$lower` and `CI(m)$stepPar$upper`). `CI(m)$anglePar` and `CI(m)$beta` are respectively the bounds of the confidence intervals for the angle parameters and the regression coefficients of the transition probabilities.

```
CI(m)

## $stepPar
## $stepPar$lower
##                 state 1  state 2
## mean      0.2968064964 2.626198
## sd        0.3056841408 3.453159
## zero-mass 0.0002775592       NA
##
## $stepPar$upper
##               state 1  state 2
## mean      0.42461415 4.307965
## sd        0.46735363 5.426500
```

```
## zero-mass 0.01391803       NA
##
##
## $anglePar
## $anglePar$lower
##                   state 1     state 2
## mean          -3.2352342 -1.26496091
## concentration  0.4524568  0.05736118
##
## $anglePar$upper
##                   state 1   state 2
## mean          -2.7564276 1.7571470
## concentration  0.7559205 0.5157648
##
##
## $beta
## $beta$lower
##                1 -> 2     2 -> 1
## intercept  -2.6269076 -1.5625543
## dist_water -0.8582664  0.1687541
##
## $beta$upper
##                1 -> 2    2 -> 1
## intercept  -1.4741247 0.1821512
## dist_water  0.1386188 1.9868813
```

Note that a warning message is also output:

```
  Warning message:
  In CI(m) :
  Some of the parameter estimates seem to lie close to
  the boundaries of their parameter space. The associated
  CIs are probably unreliable (or might not be computable).
```

It here refers to the zero-inflation parameter in the second state. Its estimate is very close to zero, the inferior boundary of its range (this parameter is in the interval [0,1]), and this causes the corresponding confidence interval to be unreliable. The function can sometimes fail to compute such a confidence interval, and returns `NA` instead, as in this example.

3.3 Further inference tools and visualization of the model

Various options are available for the class `moveHMM`, and here we explain how to use them in the elk example.

3.3.1 Plot the model

The fitted model can be plotted, using the generic function `plot`. A few graphical options are available and listed in the documentation. Here, we call:

```
plot(m, plotCI=TRUE)
```

This outputs:

- an histogram of step lengths of all animals, with the fitted state-dependent densities,
- an histogram of turning angles of all animals, with the fitted state-dependent densities,
- plots of the transition probabilities as functions of the covariate considered,

- a map of each animal's track, colored by states.

Figure 3 displays those plots, but showing only one of the plotted maps, namely the one corresponding to the first animal, "elk-115". The state-dependent densities are weighted by the relative frequency of each state in the most probable state sequence (decoded with the Viterbi algorithm, see Section 3.3.2). For example, if according to the most probable state sequence, one third of the observations is allocated to the first state, and two thirds to the second state, the plots of the densities in the first state are weighted with a factor 1/3, and in the second state with a factor 2/3.

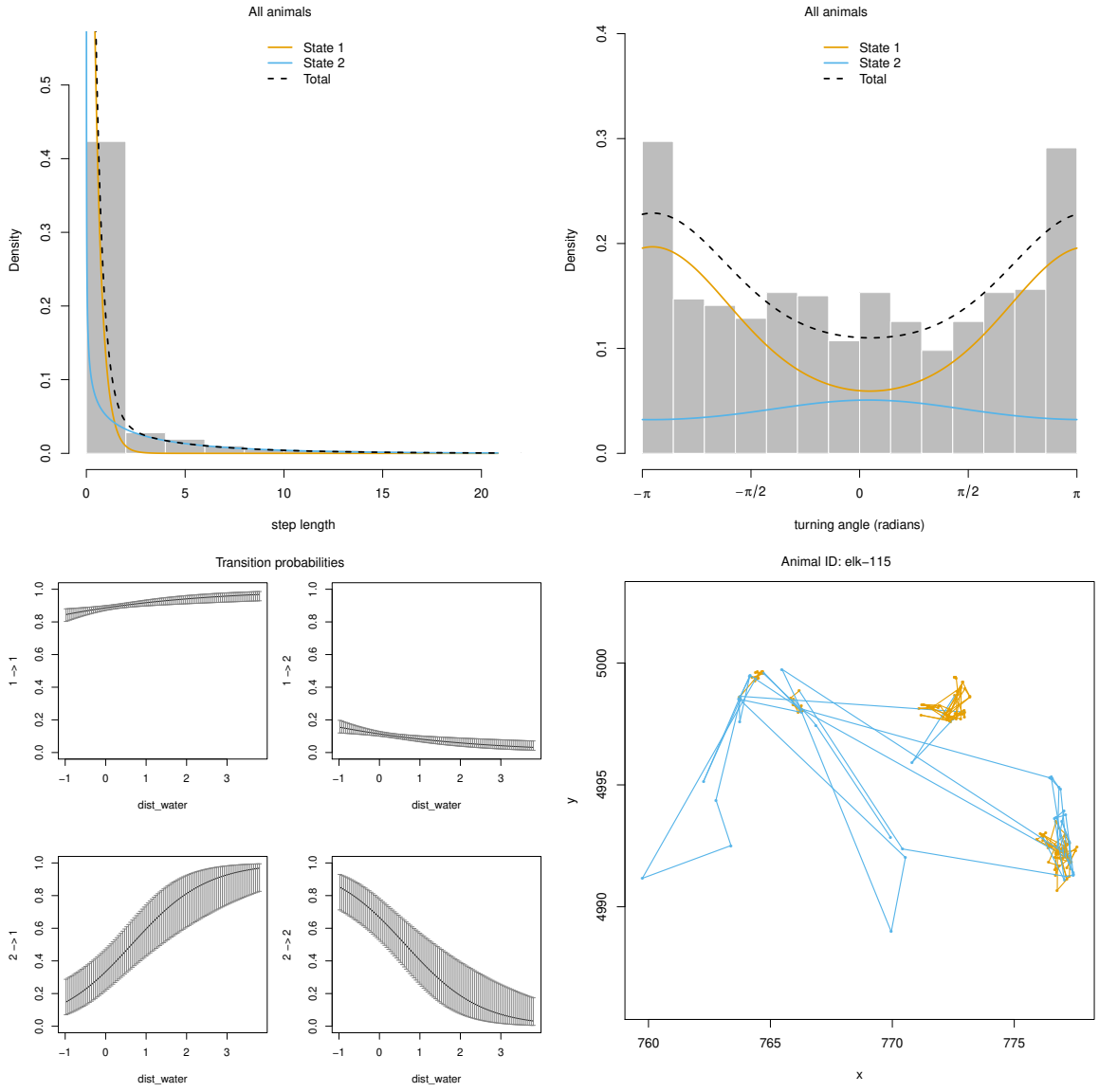

Figure 3: *Output of* `plot.moveHMM`*. Histogram of step lengths with fitted distributions (top-left), histogram of turning angles with fitted distributions (top-right), transition probabilities as functions of "dist_water" with 95% confidence intervals (bottom-left), and map of decoded track for the first animal (bottom-right).*

The first state (in orange on the plots) corresponds to short steps, and angles centered around π, and the second state (in blue on the plots) corresponds to longer steps, and angles centered around 0.

The plots of the transition probabilities as function of the covariate indicate that animals tend to switch from the second state to the first state when they are far from water, whereas they stay in the second state when closer to water.

3.3.2 State decoding

Two functions can be used to decode the state process.

Viterbi algorithm To globally decode the state process, the Viterbi algorithm is implemented in the function `viterbi`. This function outputs the most likely sequence of states to have generated the observation, under the fitted model. Below are the most probable states for the first 25 observations of the first individual:

```
states <- viterbi(m)
states[1:25]

##  [1] 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

State probabilities To get more accurate information on the state process, it is possible to compute the state probabilities for each observation, using `stateProbs`. This returns a matrix with as many columns as there are states in the model, and as many rows as there are observations (stacking all animals' observations). The elements of the matrix are defined as

$$\texttt{stateProbs(m)[t,j]} = \Pr(S_t = j)$$

where $\{S_t\}$ is the state process.

For example:

```
sp <- stateProbs(m)
head(sp)

##              [,1]      [,2]
## [1,] 6.485778e-06 0.9999935
## [2,] 1.045166e-01 0.8954834
## [3,] 6.443718e-01 0.3556282
## [4,] 6.066088e-01 0.3933912
## [5,] 4.606630e-01 0.5393370
## [6,] 3.177910e-11 1.0000000
```

The state with highest probability according to `stateProbs` might not be the same as the state in the most probable sequence returned by the Viterbi algorithm. This is because the Viterbi algorithm performs "global decoding", whereas the state probabilities are "local decoding". For more details, see Zucchini et al. (2016).

The function `plotStates` can be used to visualize the results of `viterbi` and `stateProbs`. Figure 4 shows the plots of the most likely state sequence decoded by the Viterbi algorithm, as well as both columns of the matrix of state probabilities, for one individual, "elk-115". It was obtained with the following command:

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

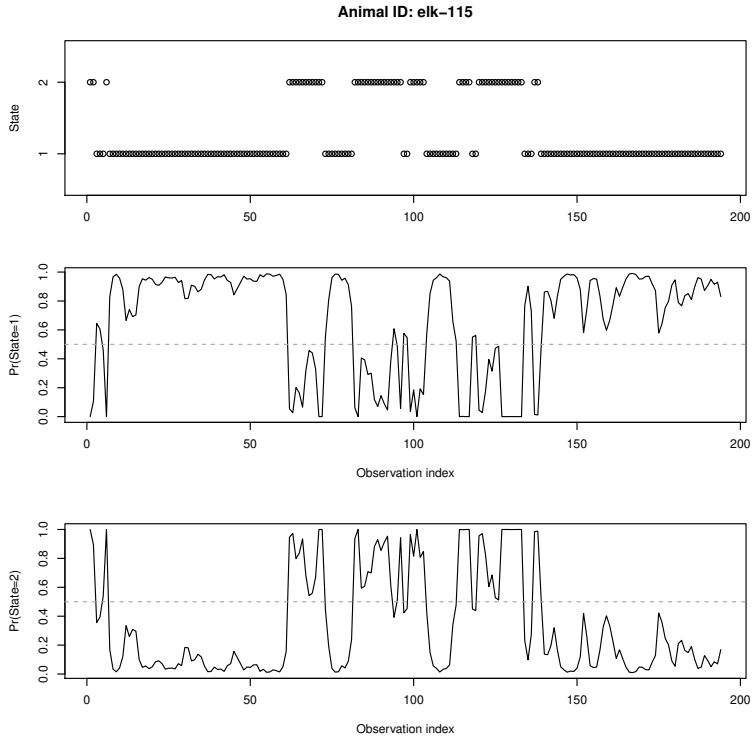

Figure 4: *Decoded states sequence (top row), and state probabilities of observations (middle and bottom rows) for elk-115*

3.3.3 Stationary state probabilities

For a transition probability matrix $\mathbf{\Gamma}$, the stationary distribution is the vector $\boldsymbol{\delta}$ that solves the equation $\boldsymbol{\delta} = \boldsymbol{\delta}\mathbf{\Gamma}$, subject to $\sum_{i=1}^{N} \delta_i = 1$ (see Section 4.1.4 for more information). It reflects the long-term proportion of time the model spends in each state.

When the transition probabilities are time-varying (i.e. functions of covariates), the stationary distribution does not exist. However, for fixed values of the covariates, we can obtain one transition probability matrix, and thus one stationary distribution. The function `plotStationary` does this over a grid of values of each covariate, and plots the resulting stationary state probabilites. They can be interpreted as the long-term probabilities of being in each state at different values of the covariate.

```
plotStationary(m, plotCI=TRUE)
```

3.3.4 Model selection with AIC

The generic method `AIC` is available to compare `moveHMM` models. For example, we now fit a 3-state HMM to the data, and want to compare the AICs of the 2-state and 3-state models.

```
# initial parameters
mu0 <- c(0.1,0.5,3)
sigma0 <- c(0.05,0.5,1)
zeromass0 <- c(0.05,0.0001,0.0001)
stepPar0 <- c(mu0,sigma0,zeromass0)
angleMean0 <- c(pi,pi,0)
```

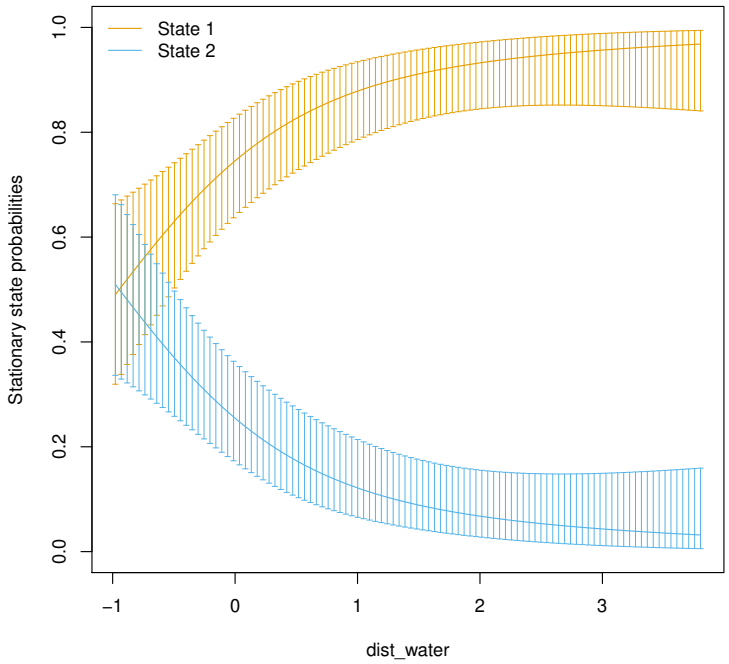

Figure 5: *Output of* `plotStationary`*. Stationary state probabilities, as functions of the distance to water, with 95% confidence intervals.*

```
kappa0 <- c(1,1,1)
anglePar0 <- c(angleMean0,kappa0)

# fit the 3-state model
m3 <- fitHMM(data=data,nbStates=3,stepPar0=stepPar0,
             anglePar0=anglePar0,formula=~dist_water)
```

And, to compare them:

```
AIC(m,m3)

##   Model      AIC
## 1    m3 3672.520
## 2     m 3815.899
```

In terms of AIC, the 3-state model is favoured over the 2-state model in this example.

3.3.5 Model checking

The pseudo-residuals (a.k.a. quantile residuals) of the model can be computed with `pseudoRes`. These follow a standard normal distribution if the fitted model is the true data-generating process. In other words, a deviation from normality indicates a lack of fit. For more theoretical background on pseudo-residuals, see Zucchini et al. (2016). The pseudo-residuals of the 2-state model fitted to the elk data are displayed in Figure 6. They can be computed and plotted with the following commands.

```
# compute the pseudo-residuals
pr <- pseudoRes(m)

# time series, qq-plots, and ACF of the pseudo-residuals
plotPR(m)
```

If some steps are of length zero (i.e. if the step distribution is zero-inflated), the corresponding pseudo-residuals are plotted as segments on the qq-plot. It is the case for the smallest step pseudo-residual located in the bottom-left corner of the step qq-plot in Figure 6. The pseudo-residuals of discrete data are defined as segments, and in this case, the segments start in $-\infty$.

3.4 Use `plotSat`

The function `plotSat` plots the tracking data on a satellite image, with the help of the package ggmap (Kahle and Wickham, 2013). Il only works with longitude and latitude values, so we need to convert the UTM coordinates first. We can do this e.g. with the help of the packages sp (Pebesma and Bivand, 2005) and rgdal (Bivand et al., 2016) (more information can be found in the documentations of those packages).

```
library(rgdal)
utmcoord <- SpatialPoints(cbind(data$x*1000,data$y*1000),
                          proj4string=CRS("+proj=utm +zone=17"))
llcoord <- spTransform(utmcoord,CRS("+proj=longlat"))
lldata <- data.frame(ID=data$ID,x=attr(llcoord,"coords")[,1],
                     y=attr(llcoord,"coords")[,2])
```

In the code above, we need to multiply the UTM coordinates by 1000, as we had divided them by 1000 earlier to work with distances in kilometres. In the function `SpatialPoints`, we indicate `+zone=17`, because the data come from the UTM zone 17.

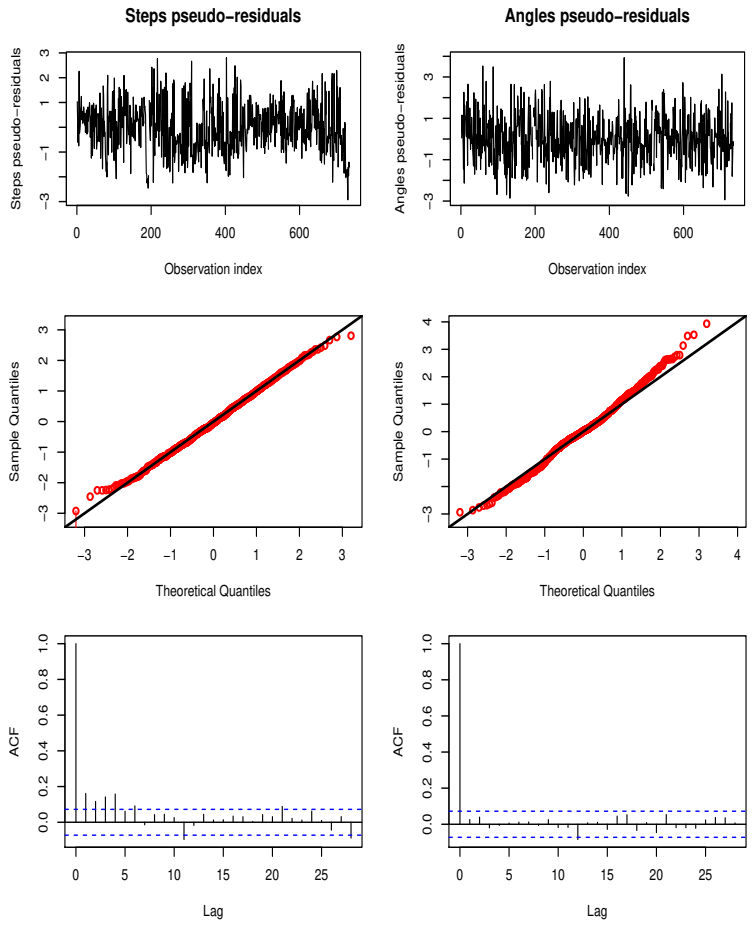

Figure 6: *Time series, qq-plots, and autocorrelation functions of the pseudo-residuals of the 2-state model.*

The data frame `lldata` contains the converted longitude and latitude coordinates of the observations. We can plot it with `plotSat`; the result is shown in Figure 7. As the satellite image is fetched from Google, an Internet connection is required to use `plotSat`.

```
plotSat(lldata,zoom=8)
```

3.5 Dealing with one-dimensional data

It is sometimes of interest to model one-dimensional movement data, like e.g. dive data. This can be done in moveHMM by setting all values of the second coordinate to zero in the data, and use the option `angleDist="none"`.

In the case where one-dimensional data are provided, the plotting functions for the data and for the model will output plots of the first coordinate as a function of time, instead of a map of the track.

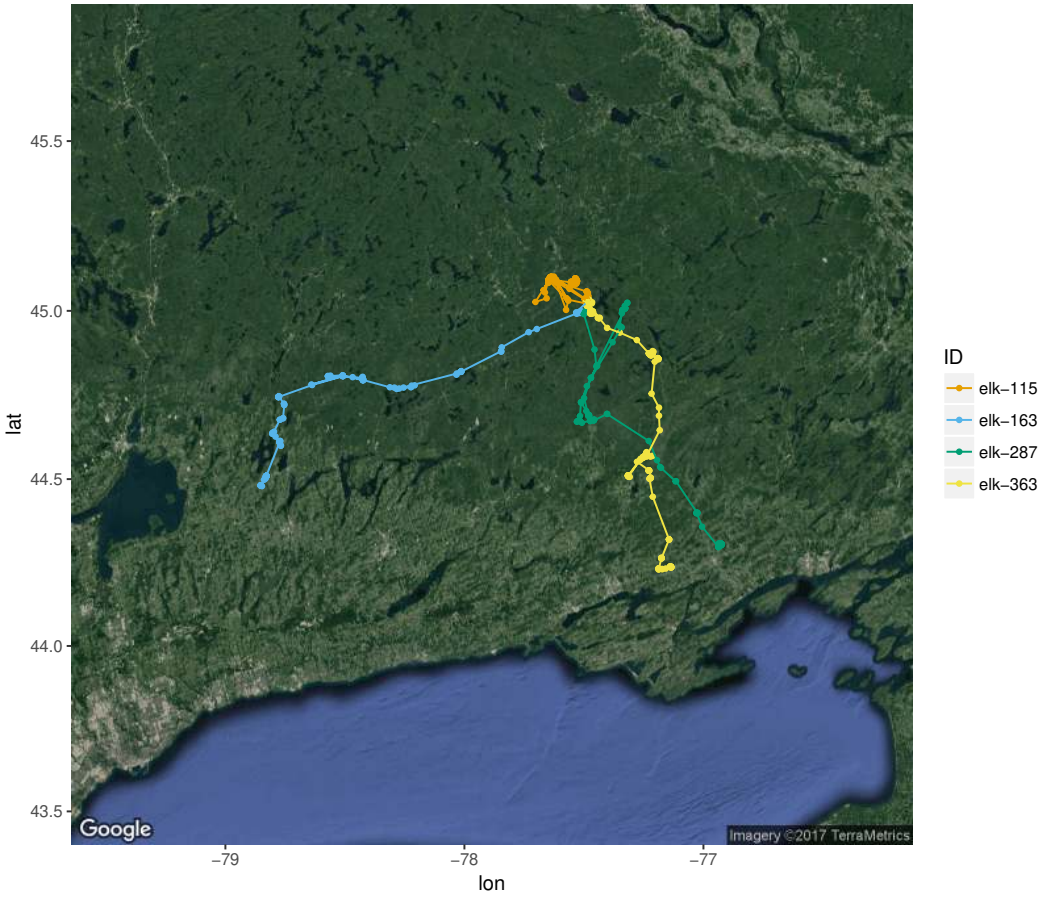

Figure 7: *Plot of the elk data, obtained with plotSat.*

4 Package features

In this section, we describe the global structure of the package, and then describe in more detail the main functions required to fit an HMM to movement data.

The package is articulated in terms of two S3 classes: `moveData` and `moveHMM`. The first extends the native R data frame, 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.

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`, `AIC.moveHMM`), and a variety of other functions can be called on `moveHMM` objects. All functions are described in more detail in Section 4.2 and their use is explained on an example in Section 3. Figure 8 illustrates the links between the main components of the package.

Note we will occasionally omit the class to which a method belongs, if the context makes it clear.

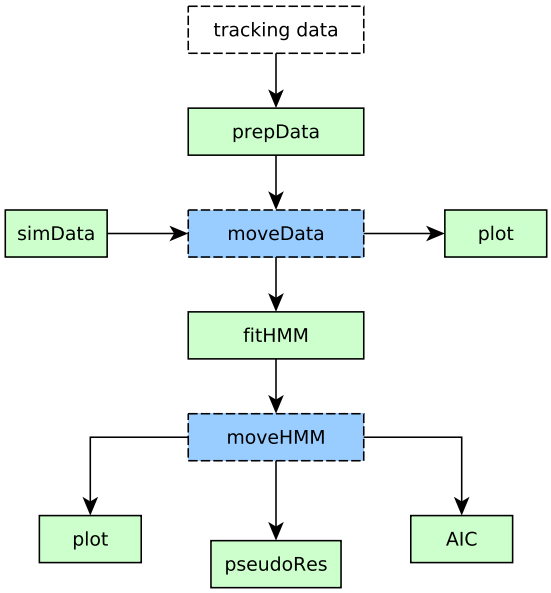

Figure 8: *Structure of the main components of the package. The blue boxes are S3 classes, and the green boxes are functions. The arrows indicate input and output of data.*

Besides, as illustrated in the example in Section 3, it is not necessary to specify the class when calling the R function (e.g. calling `plot` on a `moveHMM` object automatically refers to `plot.moveHMM`).

4.1 Model options

4.1.1 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 are not 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 $\in (0, +\infty)$	standard deviation $\in (0, +\infty)$
Weibull	shape $\in (0, +\infty)$	scale $\in (0, +\infty)$
log-normal	location $\in \mathbb{R}$	scale $\in (0, +\infty)$
exponential	rate $\in (0, +\infty)$	
von Mises	mean $\in (-\pi, \pi]$	concentration $\in (0, +\infty)$
wrapped Cauchy	mean $\in (-\pi, \pi]$	concentration $\in (0, 1)$

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

$$\text{shape} = \frac{\text{mean}^2}{\text{SD}^2}, \qquad \text{rate} = \frac{\text{mean}}{\text{SD}^2}$$

4.1.2 Zero-inflation

If some steps are exactly equal to zero, then strictly positive distributions such as the gamma are inadequate. In such cases, zero-inflated distributions can be considered. A zero-inflated step length distribution simply assumes that there is a probability z of observing a 0 and a probability of $1-z$ of observing a positive value distributed according to a standard positive distribution (e.g. a gamma). Within the package `moveHMM`, zero-inflation will automatically be included if there are zero steps. In that case the (state-dependent) values z will be estimated, with the remaining positive distribution, weighted by $1-z$, specified as one of the available standard step length distributions, listed in Section 4.1.1.

4.1.3 Covariates

In practice it is often of interest to model the state transition probabilities as functions of time-varying covariates. This can be done by assuming the Markov chain to be time-varying, with transition probability matrix $\mathbf{\Gamma}^{(t)} = \left(\gamma_{ij}^{(t)}\right)$, linking the transition probabilities to the covariate(s) via the multinomial logit link. In the general case of N states,

$$\gamma_{ij}^{(t)} = \Pr\big(S_t = j | S_{t-1} = i\big) = \frac{\exp(\eta_{ij})}{\sum_{k=1}^{N} \exp(\eta_{ik})},$$

where

$$\eta_{ij} = \begin{cases} \beta_0^{(ij)} + \sum_{l=1}^{p} \beta_l^{(ij)} w_{lt} & \text{if } i \neq j, \\ 0 & \text{otherwise,} \end{cases}$$

for $i, j = 1, \ldots, N$. Here $\{S_t\}$ is the state process, w_{lt} is the l-th covariate at time t and p is the number of covariates considered. The β parameters directly affect the off-diagonal elements in $\mathbf{\Gamma}^{(t)}$ — with an increase in the linear predictor η_{ij} resulting in an increase in $\gamma_{ij}^{(t)}$ — and hence also the diagonal entries due to the row constraints (with the entries in each row summing to one). Note in particular that we have to fix $\eta_{ii} = 0$ for all i since otherwise the model would be overparameterized (not identifiable).

Within `moveHMM`, the β coefficients for the off-diagonal transition probabilities are stored in an $(p+1) \times (N \cdot (N-1))$ matrix. For example, for a 3-state HMM with two covariates, the matrix `beta` is

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

Here the first row corresponds to the intercept terms and the other two rows to the slope coefficients associated with the two covariates. There are as many columns as there are off-diagonal entries in the 3×3 transition probability matrix, and that matrix is filled row-wise (i.e. column 1 in `beta` is linked to $\gamma_{12}^{(t)}$, column 2 is linked to $\gamma_{13}^{(t)}$, column 3 is linked to $\gamma_{21}^{(t)}$, etc.).

In practice, many movement models involve only two states, in which case the above equations boil

down to

$$\mathbf{\Gamma}^{(t)} = \begin{pmatrix} \frac{1}{1+\exp\left(\beta_0^{(12)}+\sum_{l=1}^{p}\beta_l^{(12)}w_{lt}\right)} & \frac{\exp\left(\beta_0^{(12)}+\sum_{l=1}^{p}\beta_l^{(12)}w_{lt}\right)}{1+\exp\left(\beta_0^{(12)}+\sum_{l=1}^{p}\beta_l^{(12)}w_{lt}\right)} \\ \frac{\exp\left(\beta_0^{(21)}+\sum_{l=1}^{p}\beta_l^{(21)}w_{lt}\right)}{1+\exp\left(\beta_0^{(21)}+\sum_{l=1}^{p}\beta_l^{(21)}w_{lt}\right)} & \frac{1}{1+\exp\left(\beta_0^{(21)}+\sum_{l=1}^{p}\beta_l^{(21)}w_{lt}\right)} \end{pmatrix}$$

$$= \begin{pmatrix} 1-\text{logit}^{-1}\left(\beta_0^{(12)}+\sum_{l=1}^{p}\beta_l^{(12)}w_{lt}\right) & \text{logit}^{-1}\left(\beta_0^{(12)}+\sum_{l=1}^{p}\beta_l^{(12)}w_{lt}\right) \\ \text{logit}^{-1}\left(\beta_0^{(21)}+\sum_{l=1}^{p}\beta_l^{(21)}w_{lt}\right) & 1-\text{logit}^{-1}\left(\beta_0^{(21)}+\sum_{l=1}^{p}\beta_l^{(21)}w_{lt}\right) \end{pmatrix}$$

The inverse logit link function is applied in order to map the real-valued predictor onto the interval $[0, 1]$ (with the above multinomial logit link representing a generalization of this approach to the case of $N > 2$ states). In the case of two states, the matrix β in `moveHMM` is structured as follows:

$$\begin{pmatrix} \beta_0^{(12)} & \beta_0^{(21)} \\ \beta_1^{(12)} & \beta_1^{(21)} \\ \vdots & \vdots \\ \beta_p^{(12)} & \beta_p^{(21)} \end{pmatrix}.$$

4.1.4 Stationarity

The function `fitHMM` includes the option of fitting a stationary model (using the option `stationary=TRUE`, with the default being `stationary=FALSE`). This is only possible if no covariates are incorporated into the model. (Otherwise the transition probabilities will be time-dependent, such that the Markov chain is non-homogeneous and in particular cannot be stationary.) When no covariates are considered and the option `stationary=TRUE` is selected, then the initial state distribution of the Markov chain will automatically be chosen as the stationary distribution (a.k.a. steady-state distribution) implied by the estimated transition probability matrix (as opposed to being estimated when `stationary=FALSE`). This stationary distribution is the vector $\boldsymbol{\delta}$ that solves the equation $\boldsymbol{\delta} = \boldsymbol{\delta}\mathbf{\Gamma}$ subject to $\sum_{i=1}^{N}\delta_i = 1$. In practice, this solution almost always exists.

4.2 Main functions

4.2.1 prepData

Tracking data usually consist of time series of either easting-northing coordinates or longitude-latitude values. However, with the HMM approach the derived quantities step lengths and turning angles are modelled.

The function `prepData` computes the steps and angles from the coordinates. As input, this function takes an R data frame with columns "x" (either easting or longitude) and "y" (either northing or latitude). If the names of the coordinates columns are not "x" and "y", then the argument `coordNames` should specify them. If several animals were observed, there should also be a column "ID" which identifies the animal being observed. If there is no "ID" column, all observations will be considered to be associated with 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 "LL" (longitude-latitude, the default) or "UTM". The former indicates that the coordinates are longitude-latitude values, and the latter that they are easting-northing values.

To compute the step lengths, `prepData` calls the function `spDistN1` from the package `sp`. The step lengths are in the unit 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 function `plot`.

4.2.2 fitHMM

Using the function `fitHMM`, an HMM can be fitted to an object of class `moveData`, via numerical maximum likelihood. The list of the arguments of `fitHMM` 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` functions (see Sections 4.2.3 and 4.2.4), 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.

4.2.3 Generic methods

Methods (i.e. class functions) are available for both `moveData` and `moveHMM` objects, to operate on them. For details on the options see the documentation, and for an example of their use, see Section 3.

- `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.
- `summary.moveData` outputs some summary information about a `moveData` object: the number of animals, the number of observations for each animal, and quantiles of the covariate values.
- `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.
- `print.moveHMM` prints the value of the maximum log-likelihood, and the maximum likelihood estimates of the parameters of the model.
- `AIC.moveHMM` returns the AIC of one or several fitted models.

4.2.4 Other operations on `moveHMM`

Other functions can be called on a `moveHMM` object, for further analysis.

- `CI` computes confidence intervals for the step length distribution parameters, for the turning angle distribution parameters, and for the regression coefficients of the transition probabilities.
- `pseudoRes` computes the pseudo-residuals of the model. These can be used to assess the goodness of fit. If the model is the true data-generating process, then the pseudo-residuals follow a standard normal distribution (Zucchini et al., 2016).
- `stateProbs` computes the probabilities of the underlying Markov chain being in the different states, at each observation, under the fitted model.

- `viterbi` computes the sequence of most probable states, under the fitted model, using the Viterbi algorithm (Zucchini et al., 2016).
- `plotStates` plots the most probable state sequence (as decoded with `viterbi`), and the state probabilities (as computed with `stateProbs`).
- `plotPR` plots time series, qq-plots, and the sample autocorrelation (ACF) functions of the pseudo-residuals of the fitted model (Zucchini et al., 2016). The qq-plots can be used to visually assess whether or not the pseudo-residuals are standard normally distributed. The points in the qq-plot will be close to the straight line if the model fits the data well. If the sample ACFs display a residual autocorrelation, then this is an indication that the model might not have captured all relevant correlation structure in the data.

4.2.5 simData

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

It is possible to call `simData` on a fitted model directly, to simulate data from it. This can be used to assess the fit, by checking that the simulated data display the same features as the real data.

4.2.6 plotSat

Based on functions provided by the package ggmap (Kahle and Wickham, 2013), `plotSat` plots tracking data on a satellite image retrieved from Google. Note that it only works with longitude and latitude values, and not with easting and northing coordinates. It can be used on any data frame with fields 'x' and 'y', e.g. typically a `moveData` object. It is necessary to specify manually the zoom level wanted, with the argument `zoom`, which can take values in $\{3, 4, 5, \ldots, 20, 21\}$. Other arguments are optional, and are described in the documentation.

References

Bivand, R., Keitt, T., and Rowlingson, B. (2016) "rgdal: Bindings for the Geospatial Data Abstraction Library", *R package version 1.1-8*, URL: `https://CRAN.R-project.org/package=rgdal`

Jonsen, I.D., Flemming, J.M., Myers, R.A. (2005), "Robust state-space modeling of animal movement data", *Ecology*, 86 (11), 2874–2880.

Kahle, D., Wickham, H. (2013) "ggmap: Spatial Visualization with ggplot2", *The R Journal*, 5 (1), 144–161. URL: `http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf`

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.

McClintock, B.T., King, R., Thomas, L., Matthiopoulos, J., McConnell, B.J., Morales, J.M. (2012), "A general discrete-time modeling framework for animal movement using multistate random walks", *Ecological Monographs*, 82 (3), 335–349.

Michelot, T., Langrock, R., Patterson, T.A. (2016), "moveHMM: an R package for analysing animal movement data using hidden Markov models", *Methods in Ecology and Evolution*, 7 (11), 1308–1315.

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.

Patterson T.A., Parton, A., Langrock, R., Blackwell, P.G., Thomas, L., King, R. (2016), "Statistical modelling of animal movement: a myopic review and a discussion of good practice", *arXiv preprint*, arXiv:1603.07511.

Pebesma, E.J., Bivand, R.S. (2005) "Classes and methods for spatial data in R", *R News*, 5 (2), 144–161. URL: `http://cran.r-project.org/doc/Rnews/`

Zucchini, W. MacDonald, I.L., Langrock, R. (2016). *Hidden Markov Models for Time Series: An Introduction Using R, Second Edition.* Chapman & Hall/CRC Press, Boca Raton, FL.