# Custom plots with moveHMM

Théo Michelot

June 23, 2025

We consider the elk data set from Morales et al. (2004), which is automatically loaded with the package moveHMM. We fit a 2-state model, and we explain how the parameter estimates can be extracted from the fitted model object. We showcase how they can be used to produce custom plots, to illustrate the results of an HMM analysis.

For more detail about preparing the data, and fitting HMMs in moveHMM, please refer to the package documentation or the main vignette. Here, we use the default distributions for the step lengths (gamma) and turning angles (von Mises), and include the effects of two covariates on the transition probabilities.

```r
set.seed(1)
library(moveHMM)

# Make up a fake covariate, to consider interactions later on
elk_data$temp <- rnorm(nrow(elk_data), 15, 8)

# Prepare data
data <- prepData(elk_data, type="UTM", coordNames = c("Easting", "Northing"))

# Number of states
nstate <- 2

# Fit model with covariates
m <- fitHMM(data, nbStates = nstate, stepPar0 = c(200, 1000, 200, 1000, 0.01, 0.01),
            anglePar0 = c(pi, 0, 1, 1), formula = ~ dist_water * temp)
```

## 1 Step lengths and turning angles

The estimates of the parameters of the step length and turning angle distributions are stored in `m$mle$stepPar` and `m$mle$stepPar`, respectively.

```r
# Estimated step length parameters
stepMean <- m$mle$stepPar["mean",]
stepSD <- m$mle$stepPar["sd",]

# Estimated turning angle parameters
angleMean <- m$mle$anglePar["mean",]
```

```r
angleCon <- m$mle$anglePar["concentration",]
```

In moveHMM, the gamma distribution of step lengths is parameterised with the mean and standard deviation, for convenience of interpretation. However, in R, the functions associated with the gamma distribution use the shape and rate parameters. We can derive the estimated shape and rate parameters as follows.

```r
stepShape <- stepMean^2/stepSD^2
stepRate <- stepMean/stepSD^2
```

We estimate the most likely state sequence using the Viterbi algorithm.

```r
# Most likely state sequence
states <- viterbi(m)
```

For each behavioural state, we plot a histogram of the step lengths that were classified in that state, and we overlay the estimated density function. This could for example be used to visually assess goodness-of-fit.

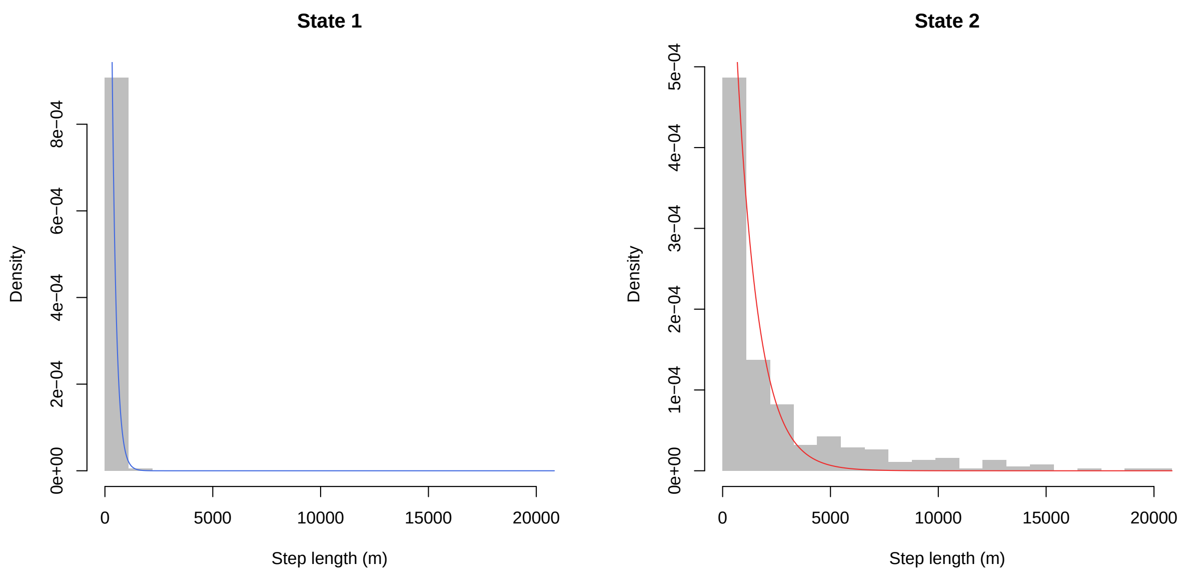
```r
# Colours for states
mycols <- c("royalblue", "firebrick2")

# Grid of step length values, to plot densities
stepgrid <- seq(min(data$step, na.rm = TRUE),
                max(data$step, na.rm = TRUE),
                length = 1000)

# Loop over states
for(s in 1:nstate) {
    # Indices of observations in state s (excluding steps of length 0)
    ind <- which(states == s & data$step != 0)

    # Histogram of step lengths in state s
    hist(data$step[ind], col = "grey", border = 0, main = paste("State", s),
         xlab = "Step length (m)", probability = TRUE,
         xlim = range(data$step, na.rm = TRUE),
         breaks = seq(0, max(data$step, na.rm = TRUE), length = 20))

    # Estimated gamma density for state s
    points(stepgrid,
           dgamma(stepgrid, shape = stepShape[s], rate = stepRate[s]),
           col = mycols[s], type = "l")
}
```



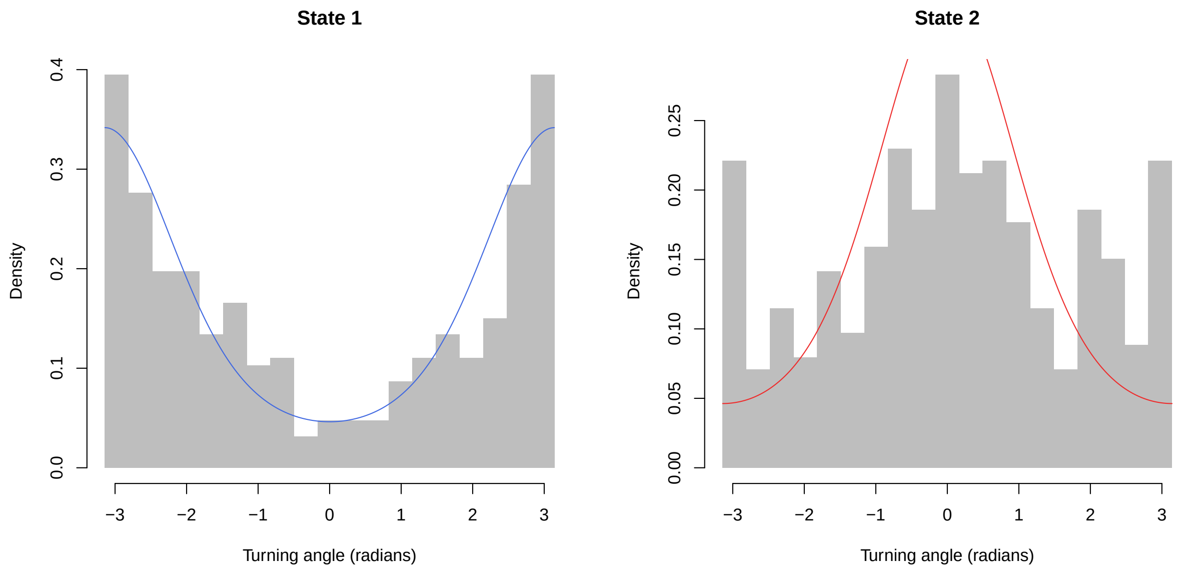
Similarly, we can obtain histograms and estimated density lines for the turning angle distributions.

```
# Grid of turning angle values
anglegrid <- seq(-pi, pi, length = 1000)

# Loop over states
for(s in 1:nstate) {
    # Indices of observations in state s
    ind <- which(states == s)

    # Histogram of turning angles in state s
    hist(data$angle[ind], col = "grey", border = 0, main = paste("State", s),
         xlab = "Turning angle (radians)", probability = TRUE,
         xlim = c(-pi, pi), breaks = seq(-pi, pi, length = 20))

    # Estimated von Mises density for state s
    points(anglegrid,
           moveHMM:::dvm(anglegrid, mu = angleMean[s], kappa = angleCon[s]),
           col = mycols[s], type = "l")
}
```



## 2 Transition probabilities

We now turn to plots of the transition probabilities as functions of covariates. The matrix of estimated regression coefficients can be extracted from the fitted model.

```
beta <- m$mle$beta
```

To plot the transition probability matrix as a function of a covariate (distance to water, say), we need to fix other covariates (temperature, say). By default, moveHMM fixes other covariates to their mean value, but this may not always be a sensible choice.

In the following, we show how the transition probabilities can be plotted as functions of the distance to water, for two different values of the temperature. This is useful to visualise the interaction between the two covariates. We define a grid of distance to water values, over which we will plot the transition probabilities of interest, and we choose two fixed values for the temperature (here, 10 degrees and 20 degrees). Then, we construct two design matrices based on those covariates, one for each value of the temperature.

```
# Distance values at which the transition probabilities should be plotted
dist_water_grid <- seq(0, max(data$dist_water), length = 1000)

# Fixed values for other covariates (temperature)
temp_fixed <- c(10, 20)

# Design matrix for temp = 10
newcovs1 <- cbind("intercept" = 1,
                  "dist_water" = dist_water_grid,
                  "temp" = temp_fixed[1],
                  "dist_water:temp" = dist_water_grid * temp_fixed[1])
```

```r
# Design matrix for temp = 20
newcovs2 <- cbind("intercept" = 1,
                  "dist_water" = dist_water_grid,
                  "temp" = temp_fixed[2],
                  "dist_water:temp" = dist_water_grid * temp_fixed[2])
```

We use the function `trMatrix_rcpp` to calculate the transition probabilities from the design matrices and the estimated regression parameters. Note that the function is not automatically exported by moveHMM, and so it needs to be prefixed by `moveHMM:::`, as shown below.

```r
# Transition probability matrices for temp = 10
tpm1 <- moveHMM:::trMatrix_rcpp(nbStates = nstate,
                                beta = beta, covs = newcovs1)

# Transition probability matrices for temp = 20
tpm2 <- moveHMM:::trMatrix_rcpp(nbStates = nstate,
                                beta = beta, covs = newcovs2)
```

The output objects, `tpm1` and `tpm2`, are three-dimensional arrays. Each slice of the array is a transition probability matrix corresponding to one row of the design matrix. For example, we can visualise the first three transition probability matrices of `tpm1`:

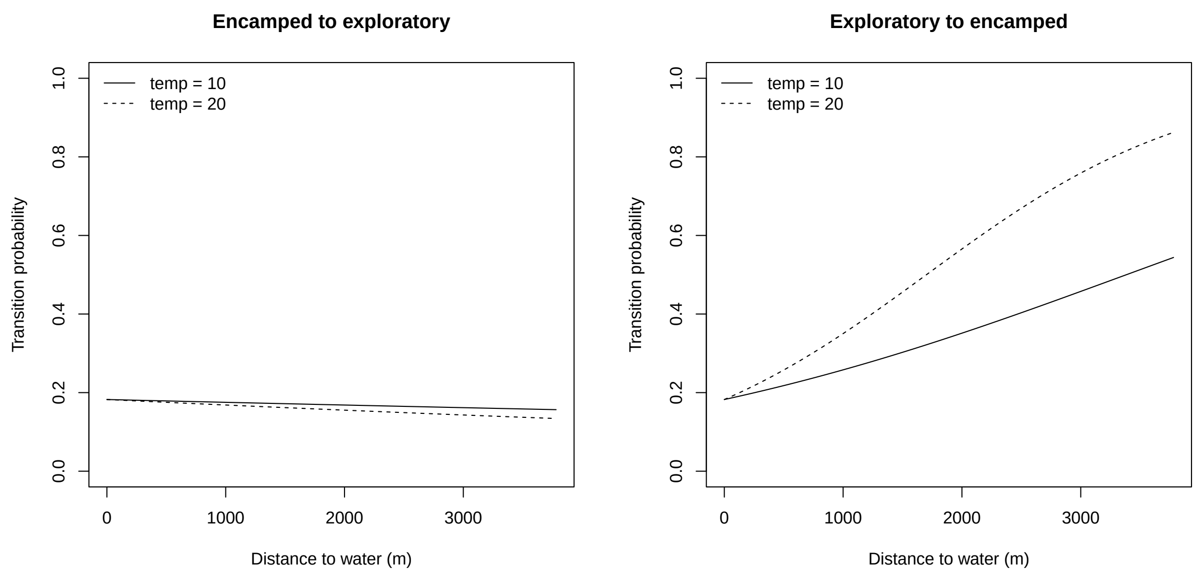
```r
tpm1[,,1:3]
```

```
## , , 1
##
##           [,1]      [,2]
## [1,] 0.8175742 0.1824258
## [2,] 0.1824255 0.8175745
##
## , , 2
##
##           [,1]      [,2]
## [1,] 0.8176016 0.1823984
## [2,] 0.1826759 0.8173241
##
## , , 3
##
##           [,1]      [,2]
## [1,] 0.8176290 0.1823710
## [2,] 0.1829265 0.8170735
```

We can now plot the transition probabilities on a grid of distance values. We display the transition probabilities obtained for the two temperature values (temp = 10 and temp = 20) on the same plots, to visualise the differences.

```
# Plot transition probability from state 1 to state 2 (for temp = 10)
plot(dist_water_grid, tpm1[1,2,], type = "l", ylim = c(0, 1),
     xlab = "Distance to water (m)", ylab = "Transition probability",
     main = "Encamped to exploratory")
# Plot transition probability for temp = 20
points(dist_water_grid, tpm2[1,2,], type = "l", lty = 2)
# Add legend
legend("topleft", legend = paste("temp =", temp_fixed), lty = 1:2, bty = "n")

# Plot transition probability from state 2 to state 1 (for temp = 10)
plot(dist_water_grid, tpm1[2,1,], type = "l", ylim = c(0, 1),
     xlab = "Distance to water (m)", ylab = "Transition probability",
     main = "Exploratory to encamped")
# Plot transition probability for temp = 20
points(dist_water_grid, tpm2[2,1,], type = "l", lty = 2)
# Add legend
legend("topleft", legend = paste("temp =", temp_fixed), lty = 1:2, bty = "n")
```



## 3 Stationary state probabilities

Similarly to the transition probabilities, we may want to plot the stationary state probabilities as functions of covariates. We use the function `stationary` to derive the stationary distributions from the design matrix defined above.

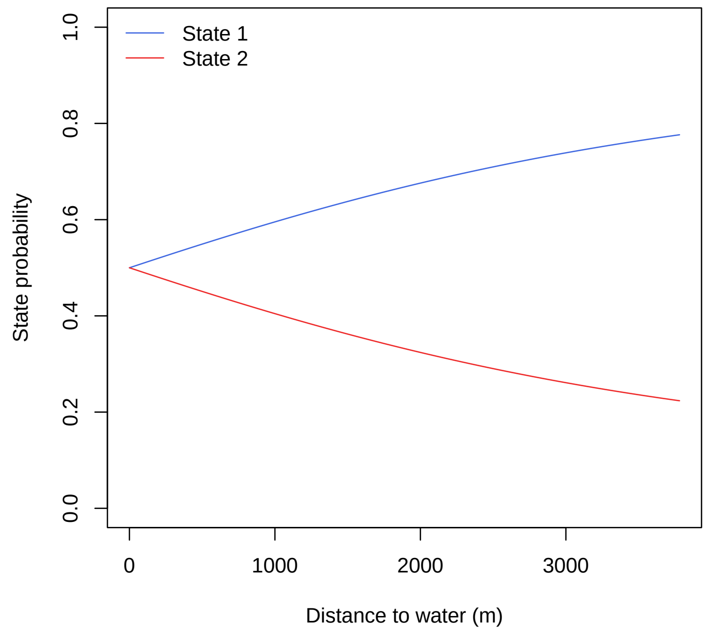
```
# Compute stationary distribution for each row of 'newcovs'
stat <- stationary(m = m, covs = newcovs1)
```

```r
# Visualise stationary distributions
head(stat)
```

```
##           [,1]      [,2]
## [1,] 0.4999995 0.5000005
## [2,] 0.5003800 0.4996200
## [3,] 0.5007603 0.4992397
## [4,] 0.5011405 0.4988595
## [5,] 0.5015207 0.4984793
## [6,] 0.5019007 0.4980993
```

```r
# Plot stationary probability of state 1 as function of distance to water
plot(dist_water_grid, stat[,1], type="l", col = mycols[1], ylim = c(0, 1),
     xlab = "Distance to water (m)", ylab = "State probability")
# Stationary probability of state 2
points(dist_water_grid, stat[,2], type="l", col = mycols[2])

# Add legend
legend("topleft", legend = paste("State", 1:2),
       col = mycols, lty = 1, bty = "n")
```



## References

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