# moveHMM demo

*Theoni Photopoulou*

*29 June 2017*

Let's go through some example code for processing track data and fitting a hidden Markov model to it using the `R` package `moveHMM` by Michelot et al. 2016. I borrow heavily from the paper's example code and the `moveHMM` vignette. I used the code in the paper to simulate the haggis data analysed below.

## Setup

First, set your working directory and load the `moveHMM` package together with the `tseries` package.

```
setwd("/Users")
require(moveHMM)
```

```
## Loading required package: moveHMM
## Loading required package: CircStats
## Loading required package: MASS
## Loading required package: boot
## Loading required package: sp
```

```
require(tseries)
```

```
## Loading required package: tseries
```

## Load the data

Read the data into `R` from a file and have a look at it.

```
rawHaggis <- read.csv("rawHaggises.csv")
head(rawHaggis)
```

```
##   ID         x         y     slope      temp
## 1  1  0.000000  0.000000 25.957002 10.344959
## 2  1 -1.068761 -0.194650 18.606632  8.352531
## 3  1 -6.152549  2.051343 16.524004 13.529650
## 4  1 -6.703983  3.338480  9.154917 10.951095
## 5  1 -6.541667  3.553843  5.547686 11.243328
## 6  1 -7.160298  1.960377  8.129402 13.187280
```

```
table(rawHaggis$ID)
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
## 400 400 400 400 400 400 400 400 400 400 400 400 400 400 400
```

## Prepare the data

You need to compute the steps and angles from the x and y locations to use in the HMM. The `moveHMM` function `prepData` does that for us! Notice that you need to specify what type of locations you have in terms of the coordinate system. Here you have UTM coordinates, but you can also also have latitude and longitude.

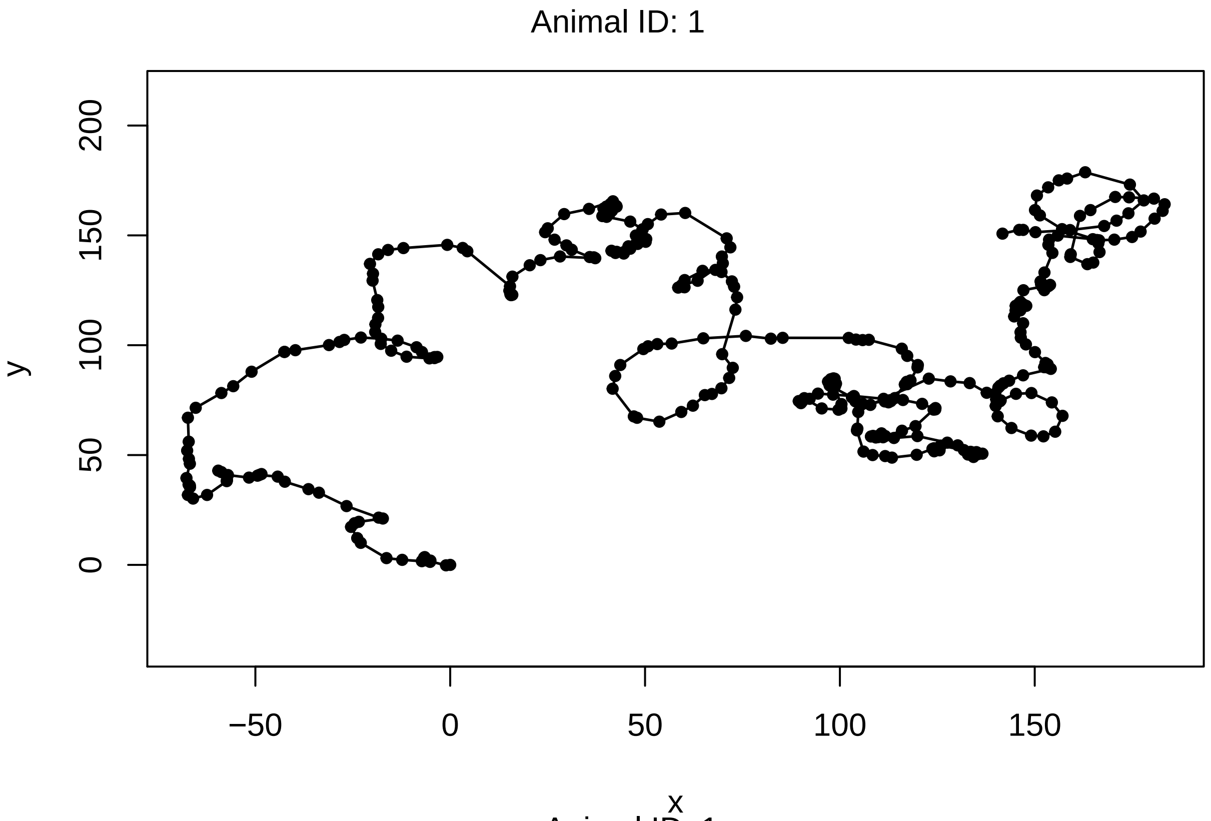
```
processedHaggis <- prepData(rawHaggis, type="UTM")
head(processedHaggis)
```

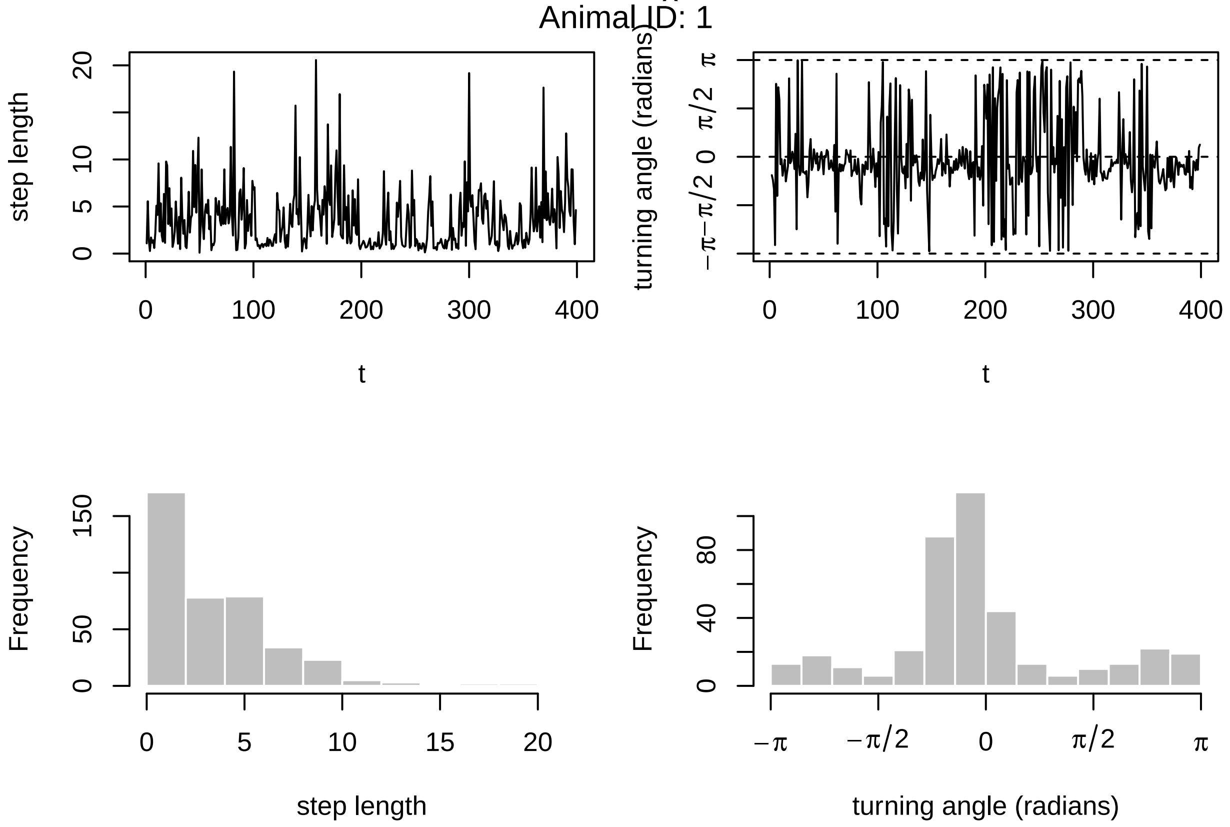
```
##   ID      step      angle         x         y     slope      temp
## 1  1 1.0863417         NA  0.000000  0.000000 25.957002 10.344959
## 2  1 5.5578218 -0.5961622 -1.068761 -0.194650 18.606632  8.352531
## 3  1 1.4002860 -0.7500230 -6.152549  2.051343 16.524004 13.529650
## 4  1 0.2696813 -1.0506197 -6.703983  3.338480  9.154917 10.951095
## 5  1 1.7093394 -2.8660552 -6.541667  3.553843  5.547686 11.243328
## 6  1 1.1529149  2.3676683 -7.160298  1.960377  8.129402 13.187280
```

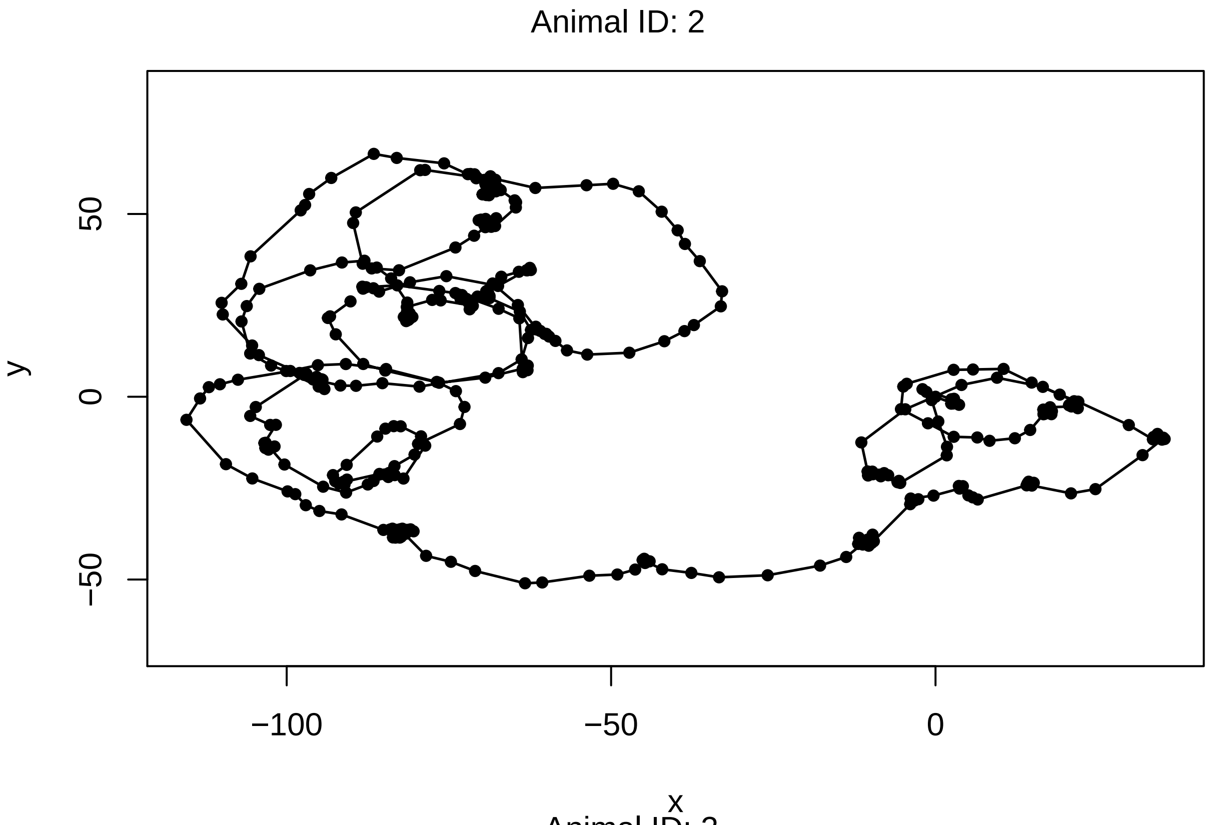
## Plot the data

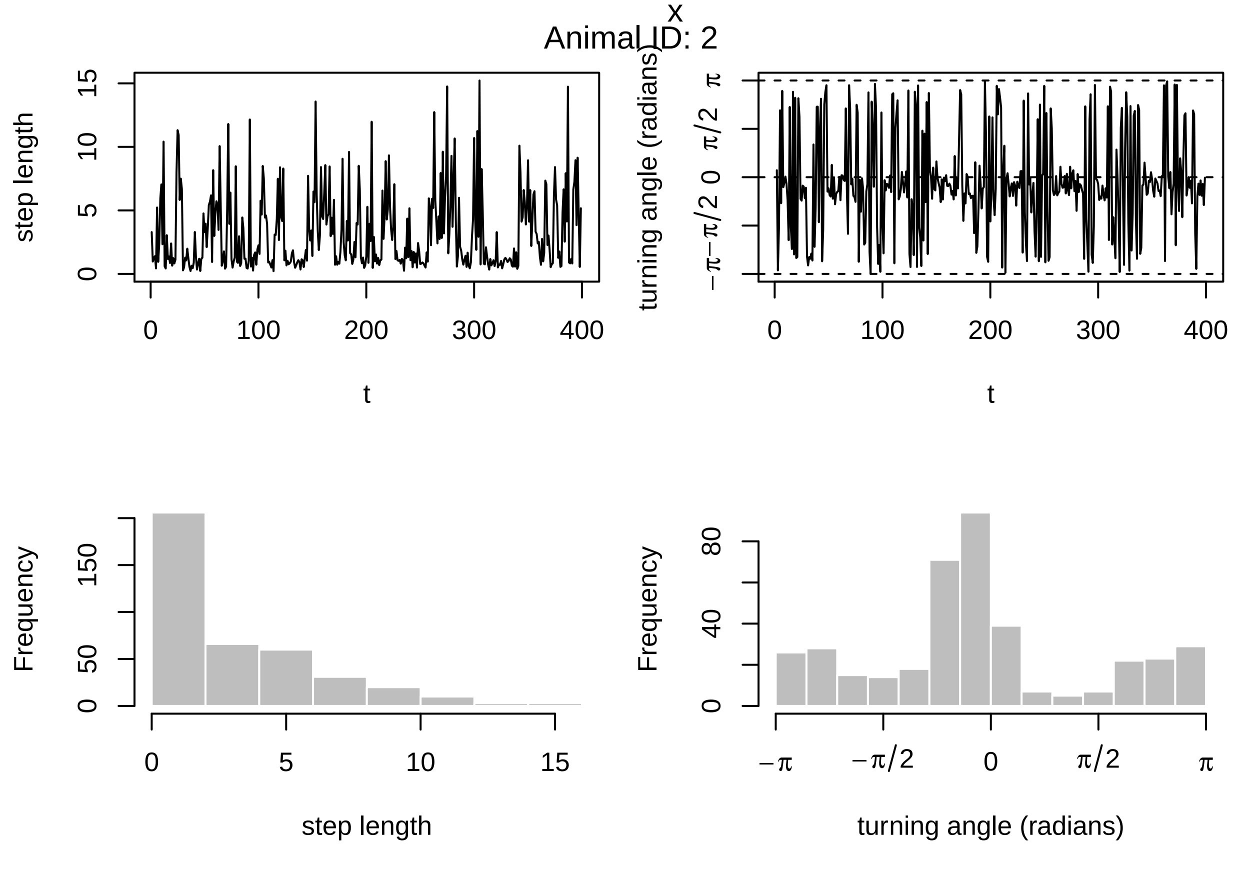
There is a really nice `moveHMM` function for plotting your track data. It shows you the track, as well as the time series and the distribution of your step lengths and your turning angles. Before it creates a new plot, it will ask you and you will have to press enter to show the next track. If you have more than one individual or track, under the `ID` column in the data, you will end up with a lot of plots.

```
plot(processedHaggis, animals=c(1,2), ask=F)
```


Animal ID: 1
y
x
200
150
100
50
0
−50
0
50
100
150


Animal ID: 1
step length
t
0
5
10
20
0
100
200
300
400
turning angle (radians)
−π
−π/2
0
π/2
π
Frequency
step length
0
50
150
0
5
10
15
20
Frequency
turning angle (radians)
0
40
80
−π
−π/2
0
π/2
π


Animal ID: 2
y
x
50
0
−50
−100
−50
0


Animal ID: 2
step length
t
turning angle (radians)
Frequency
step length
Frequency
turning angle (radians)
0
100
200
300
400
−π
−π/2
0
π/2
π

## Fit models

### How many states?

The first thing you need to do when fitting an HMM is decide how many states you think are reasonable for the data. This is a very important step and involves knowing your study system well. Make sure you can think of sound biological justification for the number of states you are estimating parameters for. You can read about the details of state selection in this paper by Pohle et al. 2017.

### Distributions

The second important thing you need to do is decide what distributions are appropriate for your step and turn distributions. If you don't specify distributions the `fitHMM` function will chose some by default (gamma distribution for step length and a von Mises distribution for turning angle), but you can also specify them yourself.

### Initial parameter values.

With HMMs you need to provide initial values for all the parameters you want to estimate, so, the mean and variance of the gamma distribution, and the mean and concentration of the von Mises distribution, along with any intercept and slope coefficients.

This is a key step and needs to be done carefully, because it can determine whether or not your model gives you sensible results.

Choose the means for step length (two parameters: one for each state)

```
mu0 <- c(1,4)
```

Choose the standard deviations for step length (two parameters: one for each state)

```
sigma0 <- c(0.5,1)
```

Group the means and standard deviations for step length together into an object

```
stepPar0 <- c(mu0,sigma0)
```

Choose the mean for angle (two parameters: one for each state)

```
angleMean0 <- c(pi,0)
```

Choose the angle concentration (two parameters: one for each state)

```
kappa0 <- c(0.7,1.5)
```

Group the means and concentrations for angle together into an object

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

Choose initial values for the intercept and slope coefficients for the covariates

```
beta0 <- matrix(c(-3,1,0.3,-0.5,-0.01,0.01), nrow=3, byrow=TRUE)
```

Now lets fit a model, starting with one without covariates

```
m1 <- fitHMM(data=processedHaggis,nbStates=2,stepPar0=stepPar0,anglePar0=anglePar0,
             formula=~1)
print(m1)
```

```
## Value of the maximum log-likelihood: -17396.68
##
## Step length parameters:
## ----------------------
##        state 1  state 2
## mean 0.9957921 4.957979
## sd   0.4963696 2.982310
##
## Turning angle parameters:
## ------------------------
##                 state 1    state 2
## mean          -3.139636 -0.2994294
## concentration  1.017413  8.1464085
##
## Regression coeffs for the transition probabilities:
## --------------------------------------------------
##              1 -> 2    2 -> 1
## intercept -1.766676 -2.020953
##
## Transition probability matrix:
## -----------------------------
##           [,1]      [,2]
## [1,] 0.8540438 0.1459562
## [2,] 0.1170205 0.8829795
##
## Initial distribution:
## --------------------
## [1] 0.4955978 0.5044022
```

### Chosing initial values

You should try several different sets of initial values for each model, as illustrated for 'm1' below, to increase the chance of finding the global maximum of the likelihood.

The maximum likelihood provides estimates of the parameters that best represent your data. If you only run the model using one set of initial values, you might get the wrong result.

### Notice that the global maximum is not always reached.

`moveHMM` will select initial values for you, but it is worth checking that you have found the global minimum in negative log likelihood. Note this next chunk might take a while to run.

```r
set.seed(1)
nruns <- 20
mles <- vector("numeric", length=nruns)
for (foo in 1:nruns){
    mu0 <- runif(2,0,4)
    sigma0 <- runif(2,0,2)
    stepPar0 <- c(mu0,sigma0)
    angleMean0 <- sample(c(0,pi),size=2,replace=TRUE)
    kappa0 <- runif(2,0,5)
    anglePar0 <- c(angleMean0,kappa0)
    m1 <- fitHMM(data=processedHaggis,nbStates=2,stepPar0=stepPar0,anglePar0=anglePar0,
                 formula=~1)
    mles[foo] <- -m1$mod$minimum
}
```

```
table(round(mles))
```

```
##
## -23150 -23108 -17397
##      2      1     17
```

Lets fit some more models with covariates and compare them. Model with covariate `slope`

```
m2 <- fitHMM(data=processedHaggis,nbStates=2,stepPar0=stepPar0,anglePar0=anglePar0,
             formula=~slope)
```

Model with covariates `slope` and `temp`.

```
m3 <- fitHMM(data=processedHaggis,nbStates=2,stepPar0=stepPar0,anglePar0=anglePar0,
             formula=~slope+temp)
```

Model with covariates `slope` and `slope^2`

```
m4 <- fitHMM(data=processedHaggis,nbStates=2,stepPar0=stepPar0,anglePar0=anglePar0,
             beta0=beta0,formula=~slope+I(slope^2))
```

The model 'm4' below is quite unstable numerically, i.e. the numerical maximization is sensitive to the choice of the initial values (which is mainly due to the use of a quadratic predictor). This is often the case with more complex models. You'll notice that this is the only model where we use our initial values `beta0` whereas in the other models we relied on `fitHMM` to find good initial values.

## Model selection

We can compare the models we have fitted using the AIC

```
print(AIC(m1,m2,m3,m4))
```

```
##   Model      AIC
## 1    m4 34104.21
## 2    m2 34675.28
## 3    m3 34679.01
## 4    m1 34815.36
```

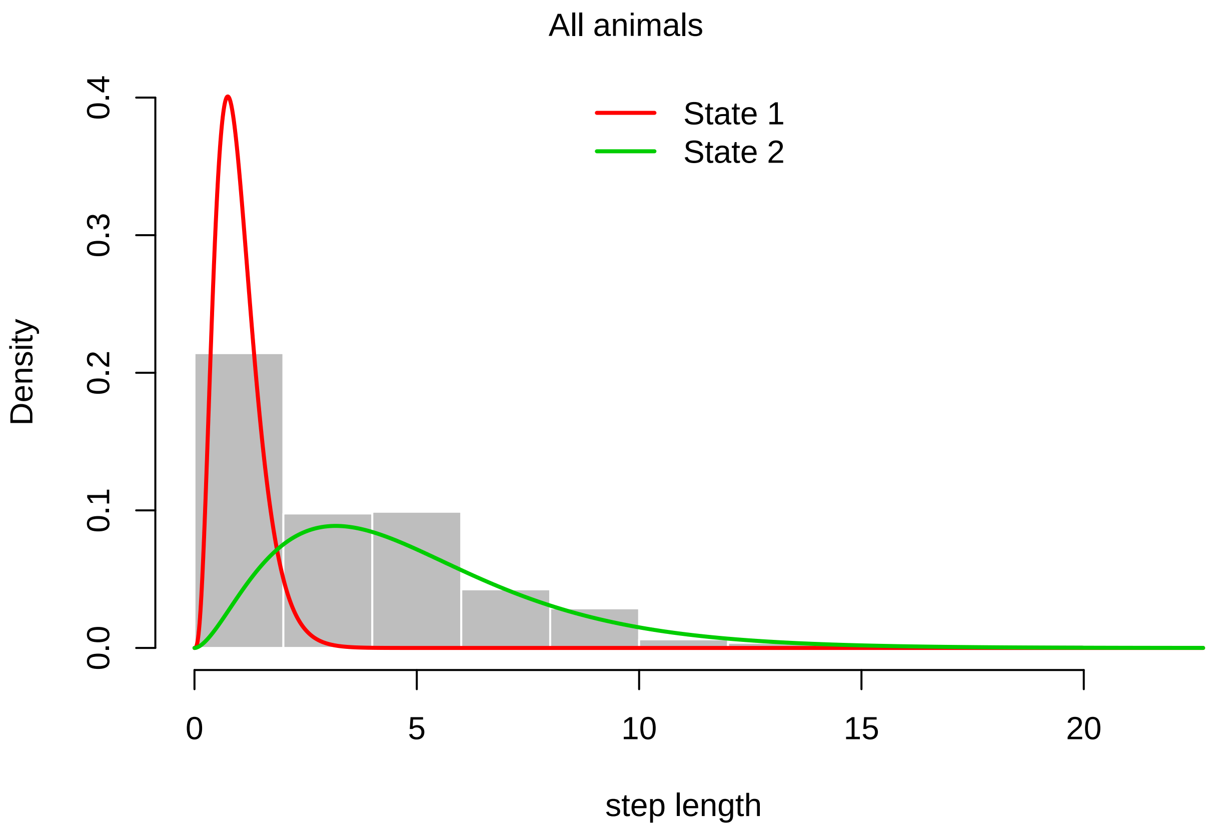
### Inspect and plot the fitted model 'm4'

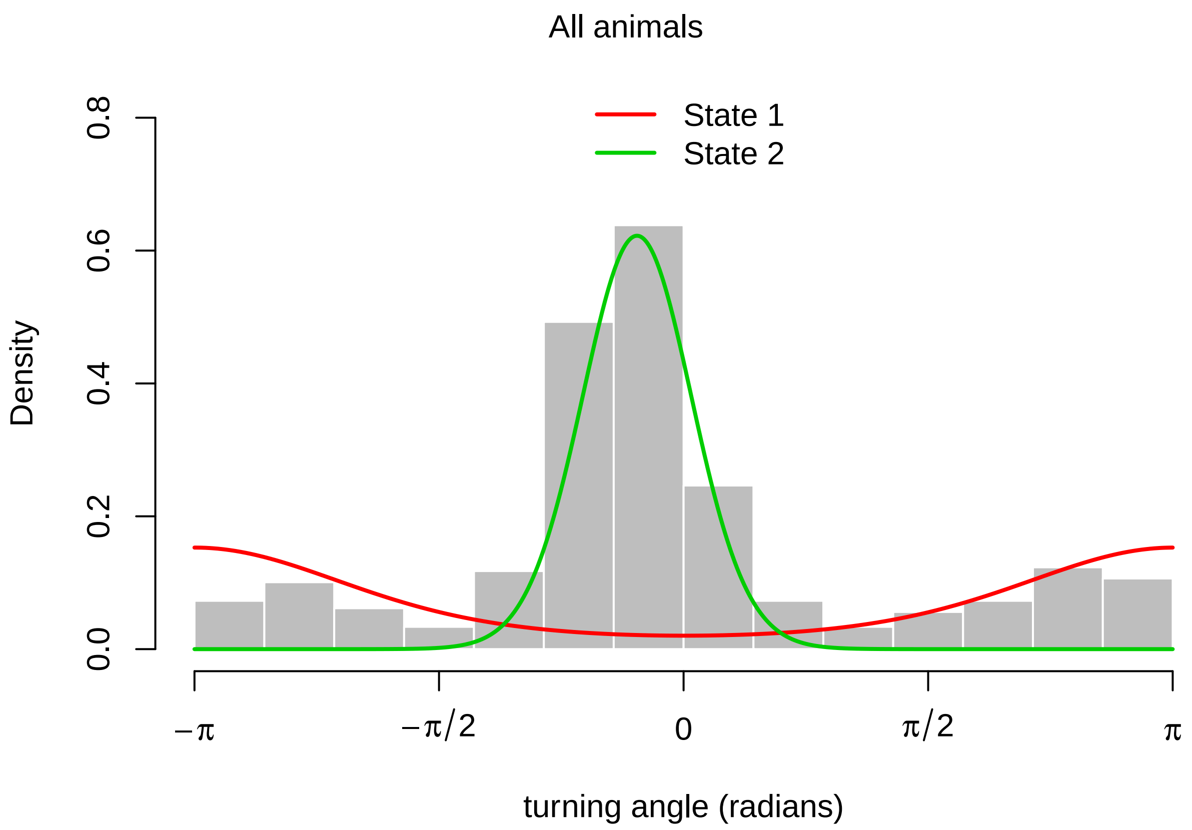
```
print(m4)
```

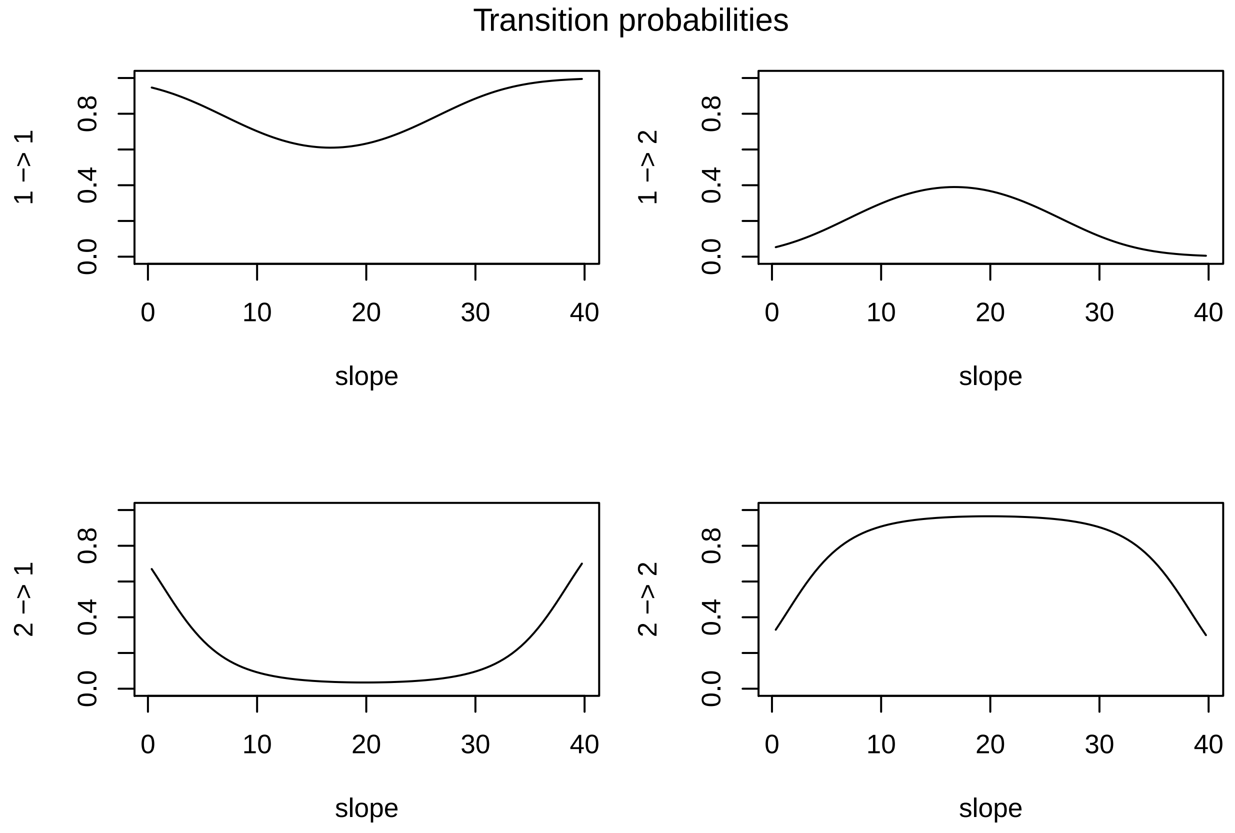
```
## Value of the maximum log-likelihood: -17037.11
##
## Step length parameters:
## ----------------------
##        state 1  state 2
## mean 0.9966482 4.964561
## sd   0.4972078 2.978937
##
## Turning angle parameters:
## ------------------------
##                  state 1    state 2
## mean           -3.137674 -0.2991378
## concentration  1.009752  8.1634222
##
## Regression coeffs for the transition probabilities:
## --------------------------------------------------
##                   1 -> 2      2 -> 1
## intercept  -2.978951388  0.85120838
## slope       0.302516794 -0.41958614
## I(slope^2) -0.009039766  0.01055137
##
## Initial distribution:
## --------------------
## [1] 0.502319 0.497681
```

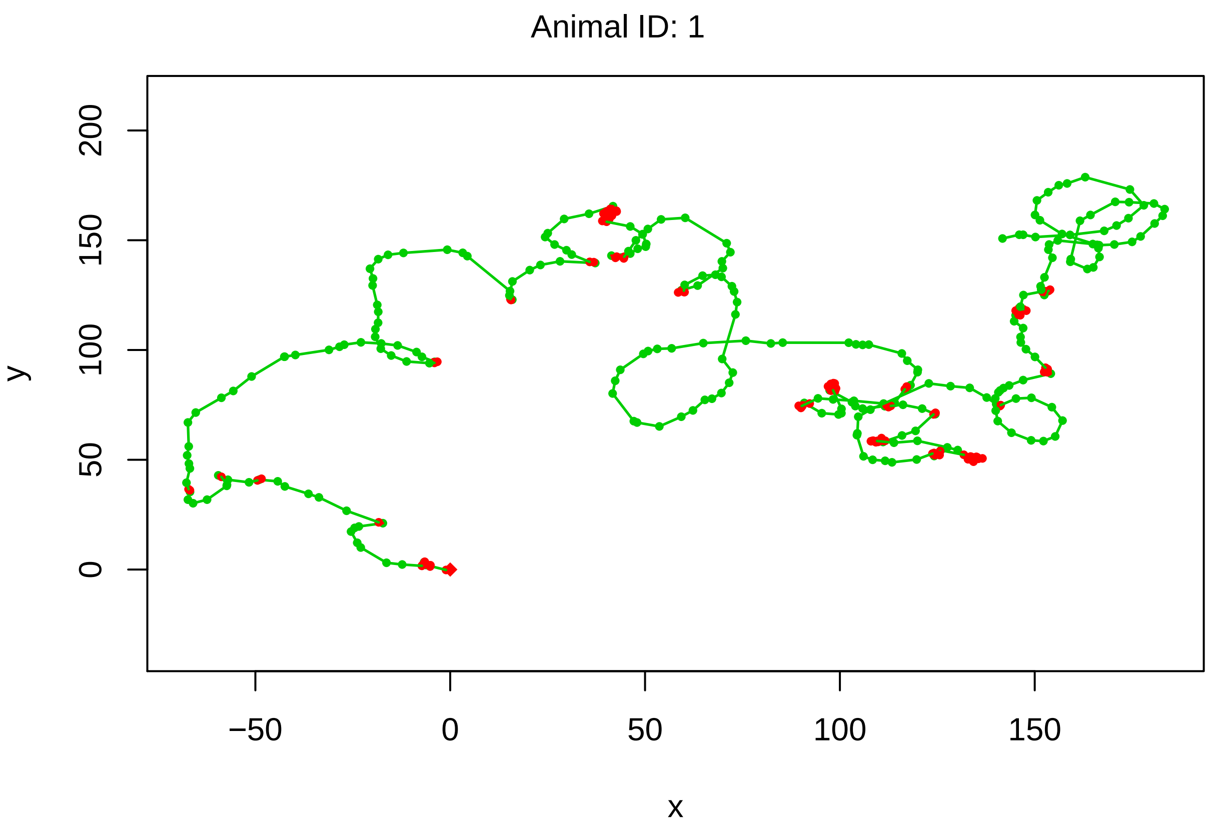
```
plot(m4, animals=1, ask=F)
```

```
## Decoding states sequence... DONE
```




All animals
State 1
State 2
Density
0.0
0.2
0.4
0.6
0.8
$-\pi$
$-\pi/2$
0
$\pi/2$
$\pi$
turning angle (radians)


Transition probabilities
1 -> 1
1 -> 2
2 -> 1
2 -> 2
0.0
0.4
0.8
0
10
20
30
40
slope



State 1 involves slow, undirected movement, whereas state 2 captures faster and more directed movement. If we think of the anatomy of the haggis with its long left leg, it makes sense that in flat terrain the haggises might have to resort to primitive crawling strategies, while in very steep terrain, they will likely fall over (Michelot et al. 2016). In terms of behaviour, we can imagine that state 1 would then capture the haggises' clumsy attempts to get back to more favourable terrains, while state 2 captures the haggises' running around hills (clockwise, when seen from above).

## Decode the most probable state sequence

```
states <- viterbi(m4)
length(states) == 15*400
```

```
## [1] TRUE
```

```
table(states)
```

```
## states
##    1    2
## 2670 3330
```

```
cat("Proportion of the time spent in state 2:",length(which(states==2))/length(states),"\n")
```

```
## Proportion of the time spent in state 2: 0.555
```

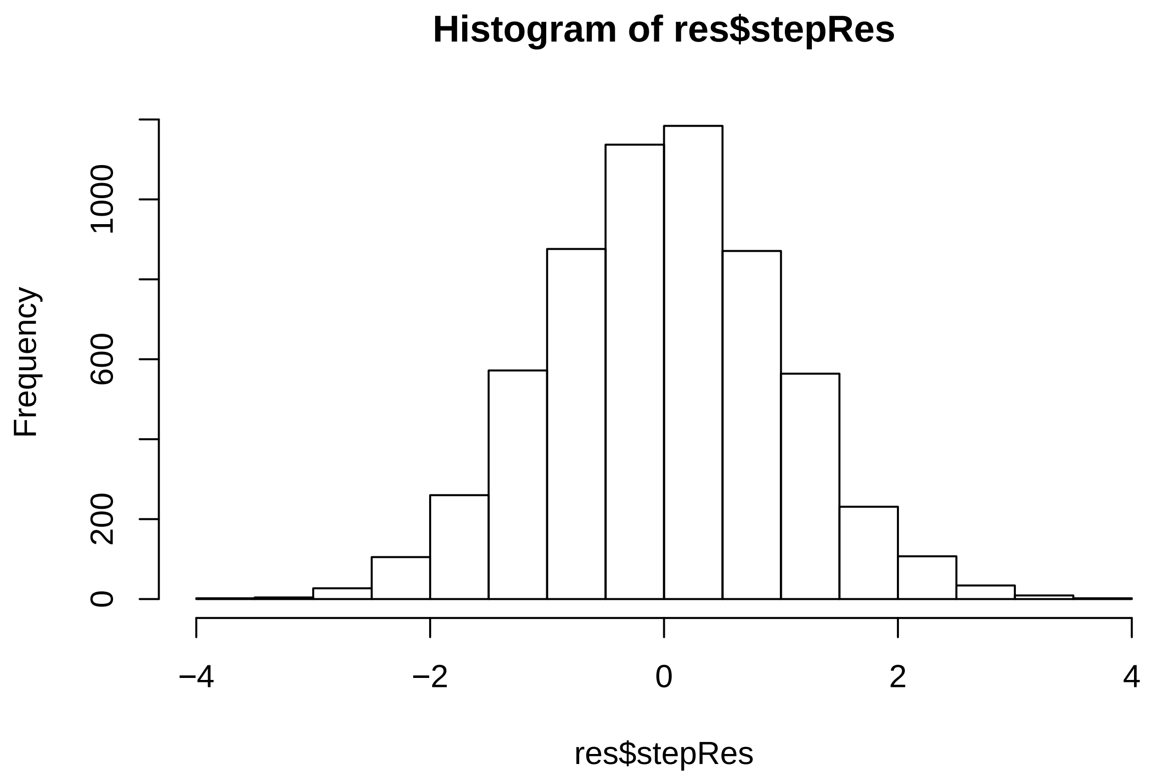
## Compute the pseudo-residuals.

If you want to know what they are read up in Zucchini, MacDonald and Langrock 2016. We use them to assess model fit.

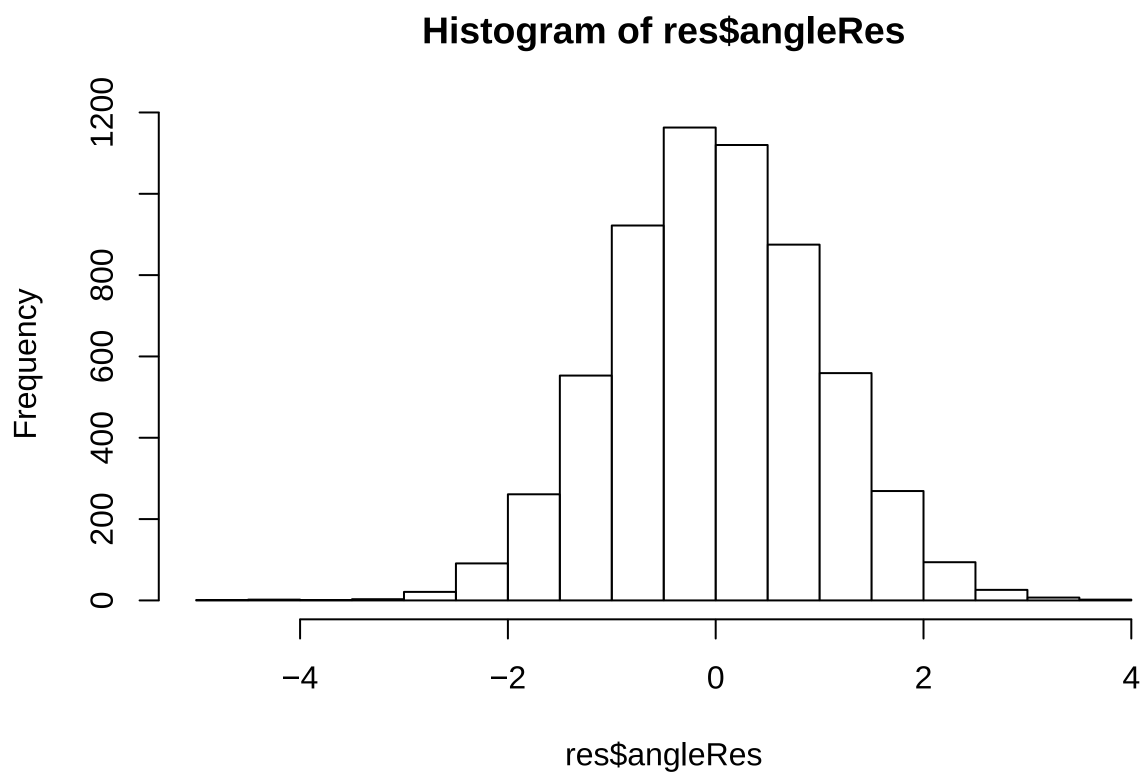
```r
res <- pseudoRes(m4)
names(res)
```

```
## [1] "stepRes"  "angleRes"
```

```r
hist(res$stepRes)
```



```r
hist(res$angleRes)
```



The histograms of pseudo-residuals look pretty good but we can also do a formal test for Normality called the Jarque Bera test. We can do this for step length pseudo-residuals

```
print(jarque.bera.test(res$step[which(!is.na(res$step))]))
```

```
##
##  Jarque Bera Test
##
## data:  res$step[which(!is.na(res$step))]
## X-squared = 0.58198, df = 2, p-value = 0.7475
```

And for turning angle pseudo-residuals

```
print(jarque.bera.test(res$angle[which(!is.na(res$angle))]))
```

```
##
##  Jarque Bera Test
##
## data:  res$angle[which(!is.na(res$angle))]
## X-squared = 0.86478, df = 2, p-value = 0.649
```

The large p-values from the Jarque Bera test suggest we have no reason to reject the null hypothesis of normality.

## Confidence intervals

Now that you are convinced you have a defensible HMM, you can also get confidence intervals for your parameter estimates

```
CI(m4)
```

```
## $stepPar
## $stepPar$lower
##        state 1  state 2
## mean 0.9776356 4.862783
## sd   0.4811722 2.888238
##
## $stepPar$upper
##        state 1  state 2
## mean 1.0160305 5.068469
## sd   0.5137779 3.072484
##
##
## $anglePar
## $anglePar$lower
##                 state 1    state 2
## mean          -3.194423 -0.3116579
## concentration  0.942860  7.7721310
##
## $anglePar$upper
##                 state 1    state 2
## mean          -3.080697 -0.2866217
## concentration  1.077508  8.5551385
##
##
## $beta
## $beta$lower
##                 1 -> 2       2 -> 1
## intercept  -3.44090217  0.416659834
## slope       0.25259528 -0.475702680
## I(slope^2) -0.01023881  0.009224972
##
## $beta$upper
##                  1 -> 2      2 -> 1
## intercept  -2.517000610  1.28575693
## slope       0.352438310 -0.36346960
## I(slope^2) -0.007840727  0.01187776
```