Hidden Markov models using R

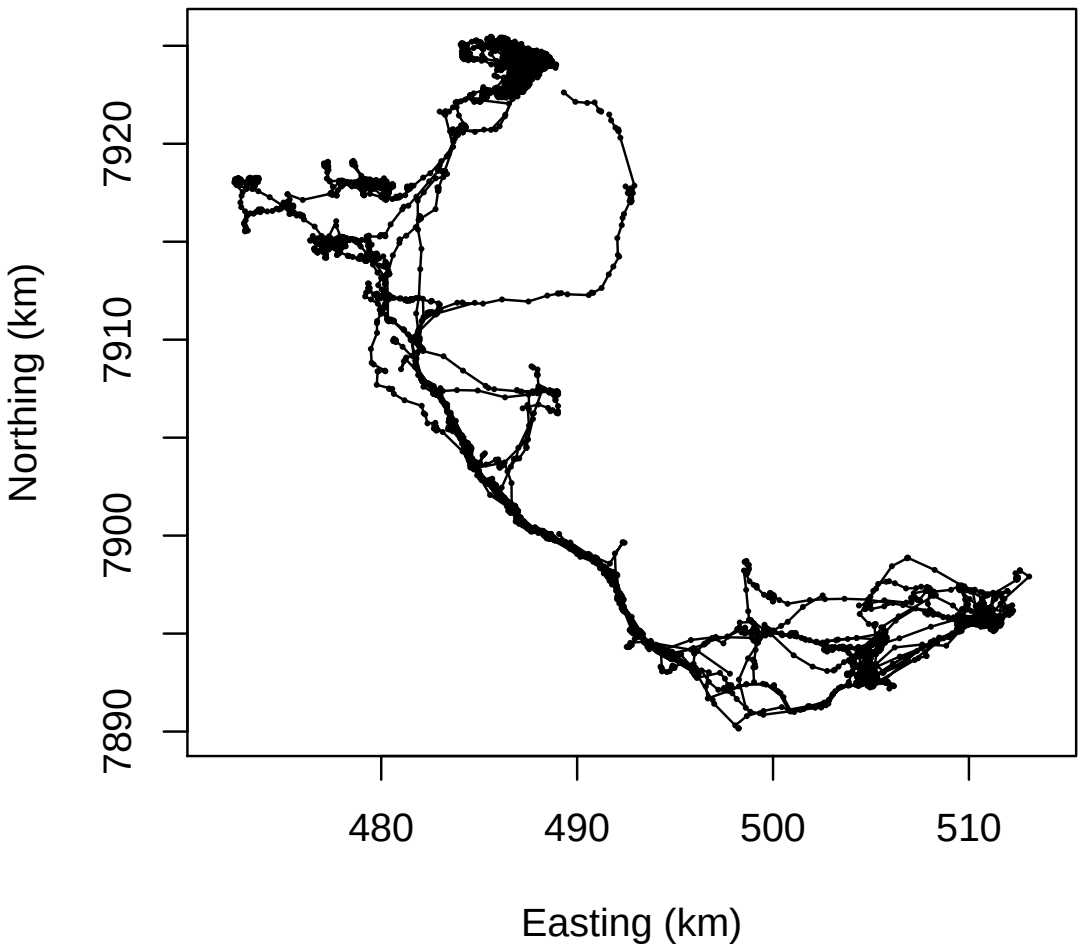
Théo Michelot

University of St Andrews

5 November 2019

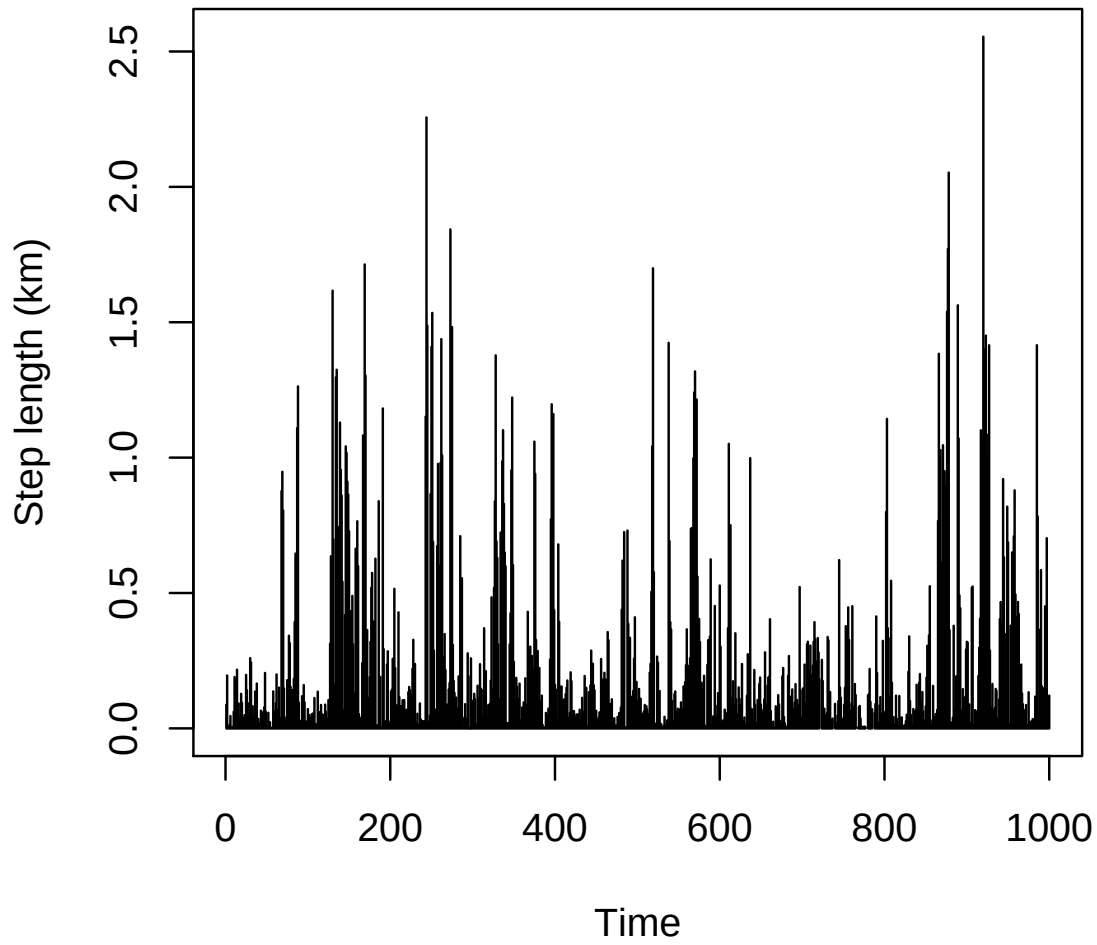
Introduction

Motivating example: zebra movement



Data from: Michelot et al. (in press). "Inference in MCMC step selection models". Biometrics.

Motivating example: zebra movement



Data from: Michelot et al. (in press). "Inference in MCMC step selection models". Biometrics.

Hidden Markov models in R

There are various R packages to fit hidden Markov models to time series data (e.g. depmixS4, HiddenMarkov...).

I will focus on **moveHMM**, a package tailored for the analysis of animal movement data.

Workflow in moveHMM

Workflow

1. Data preprocessing
2. Data visualisation
3. Model formulation
4. Model fitting
5. Model visualisation
6. Model checking

1. Prepare the data
2. Fit the model
3. Visualise the results

Input data

The input data for moveHMM is a data frame with columns:

- ID: track identifier
- x: x coordinate (Easting or longitude)
- y: y coordinate (Northing or latitude)
- Covariates

Input data: zebra example

```
track <- read.csv("code/zebra.csv")

head(track)

##       ID         x       y       d2w  tod
## 1 Apero 491830.3 7897887 3.929519 23.0
## 2 Apero 491745.5 7897867 3.947616 23.5
## 3 Apero 491804.5 7897680 3.761999  0.0
## 4 Apero 491804.0 7897676 3.761999  0.5
## 5 Apero 491802.2 7897682 3.761999  1.0
## 6 Apero 491804.4 7897696 3.787827  1.5
```

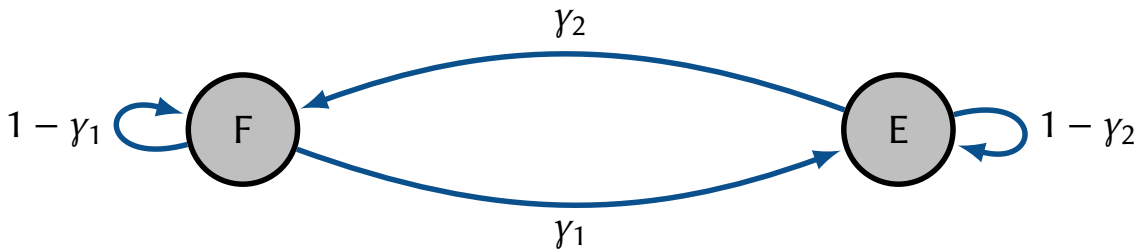
For each location we have two covariates,

- `d2w`: distance to water
- `tod`: time of day

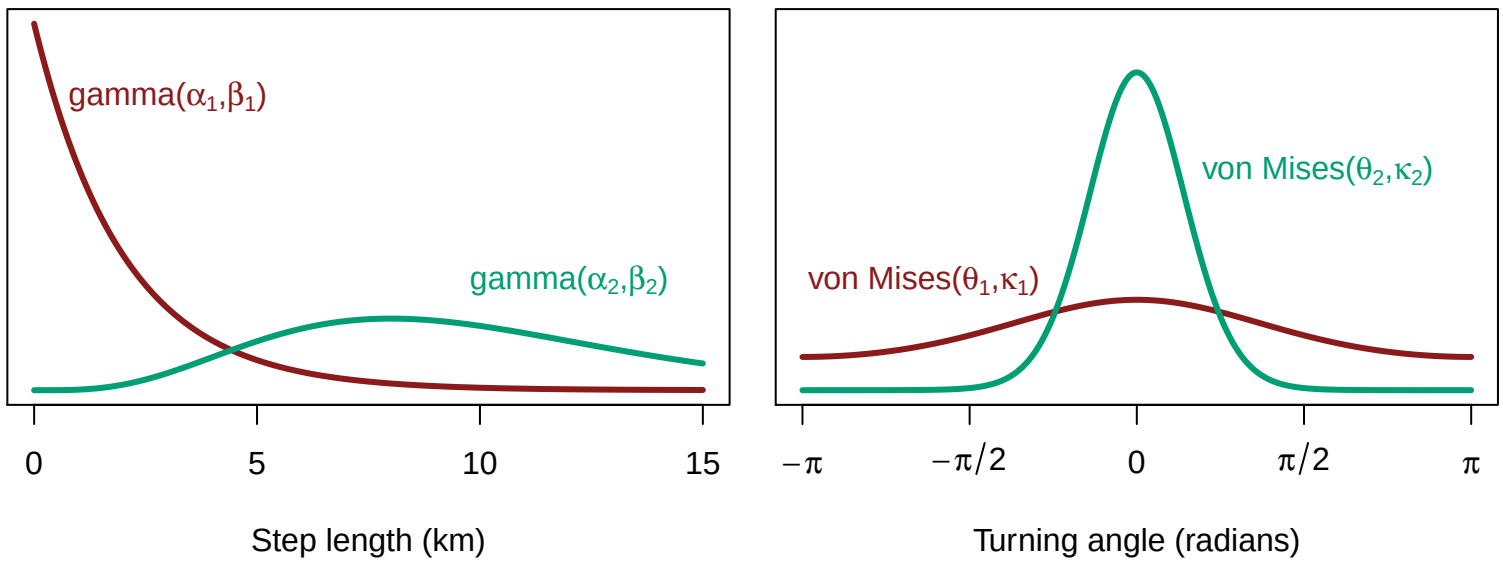
Regular sampling frequency

⟶ We need **regular time intervals** between data rows.

- Transition probabilities assume regular time intervals



- State-dependent distributions assume regular time intervals



Regular sampling frequency

- If there are missing locations, insert missing rows in data frame
- This ensures that data rows are at regular time intervals
- In R, NA indicates a missing value

```
track[720:725,]

##         ID        x       y       d2w         tod
## 720 Apero 504759.6 7893023 0.6866437 22.50000000
## 721 Apero 504965.8 7892833 0.4500312 23.00000000
## 722 Apero 504909.8 7892661 0.3001553 23.50000000
## 723 Apero       NA      NA        NA  0.01666667
## 724 Apero 504363.4 7892591 0.7626269  0.51666667
## 725 Apero 504225.9 7892379 0.8606651  1.00000000
```

Compute steps and angles

The function `prepData` calculates the step lengths and turning angles, from the locations.

It works for different types of coordinates:

- Longitude-latitude (`type="LL"`)
  ⟶ Step lengths computed with `spDistsN1` (package sp).
  ⟶ Turning angles computed with `bearing` (package geosphere).

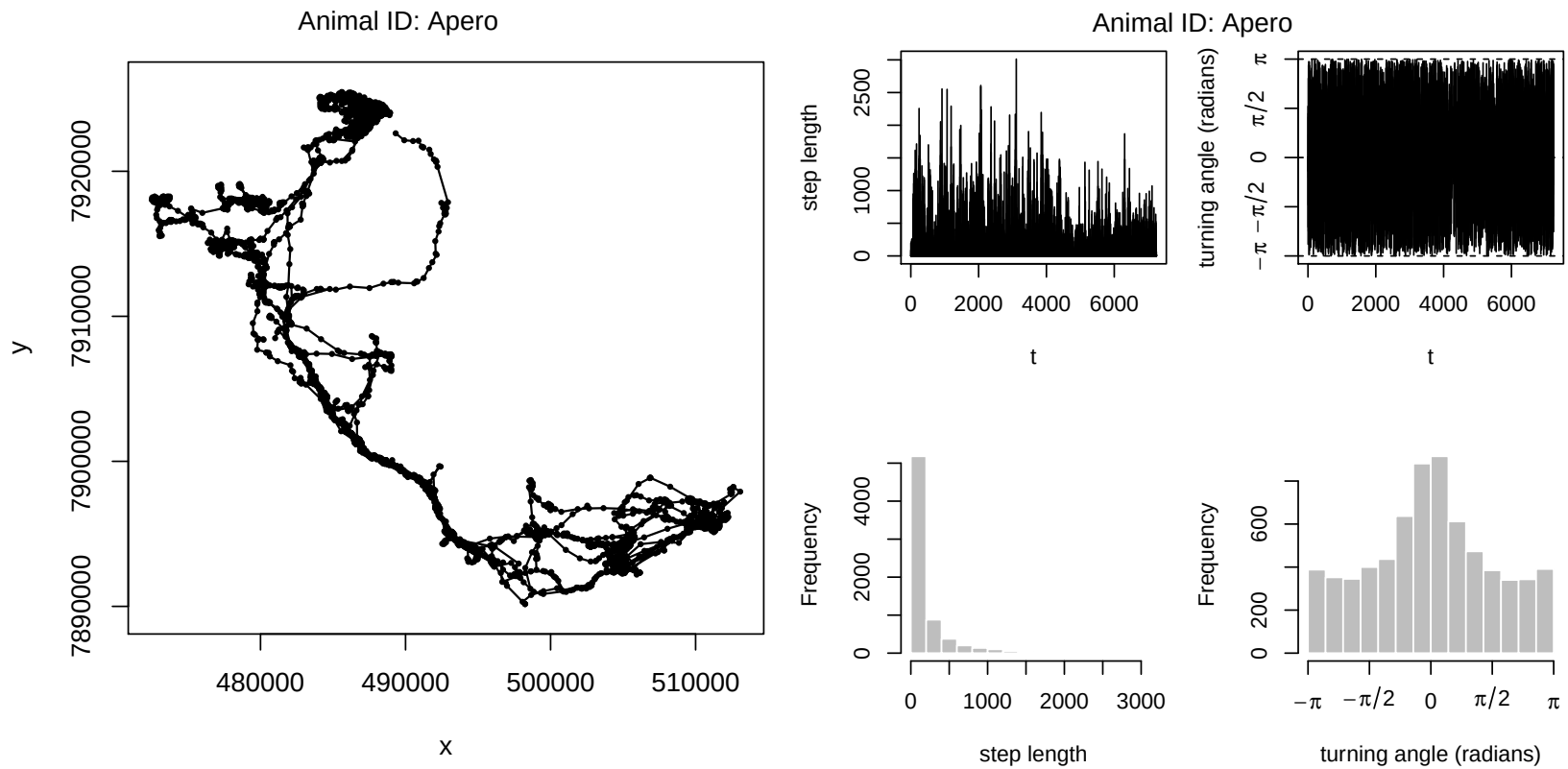
- Easting-Northing (`type="UTM"`)
  ⟶ Step lengths computed as Euclidean distance.
  ⟶ Turning angles computed with trigonometry.

prepData

```r
library(moveHMM)

data <- prepData(track, type = "UTM")

## Warning in prepData(track, type = "UTM"): There are 127
missing covariate values. Each will be replaced by the
closest available value.

head(data)

##      ID       step      angle        x       y      d2w  tod
## 1 Apero  87.364659         NA 491830.3 7897887 3.929519 23.0
## 2 Apero 195.236591  1.6367177 491745.5 7897867 3.947616 23.5
## 3 Apero   4.891494 -0.4035661 491804.5 7897680 3.761999  0.0
## 4 Apero   6.566435 -2.7686711 491804.0 7897676 3.761999  0.5
## 5 Apero  14.350180 -0.4328038 491802.2 7897682 3.761999  1.0
## 6 Apero  45.818968 -0.1076072 491804.4 7897696 3.787827  1.5
```

Visualise the data

```
plot(data, ask=FALSE)
```

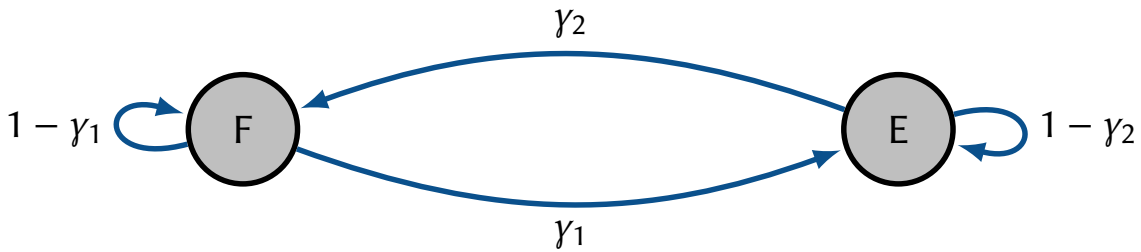


1. Prepare the data
2. Fit the model
3. Visualise the results

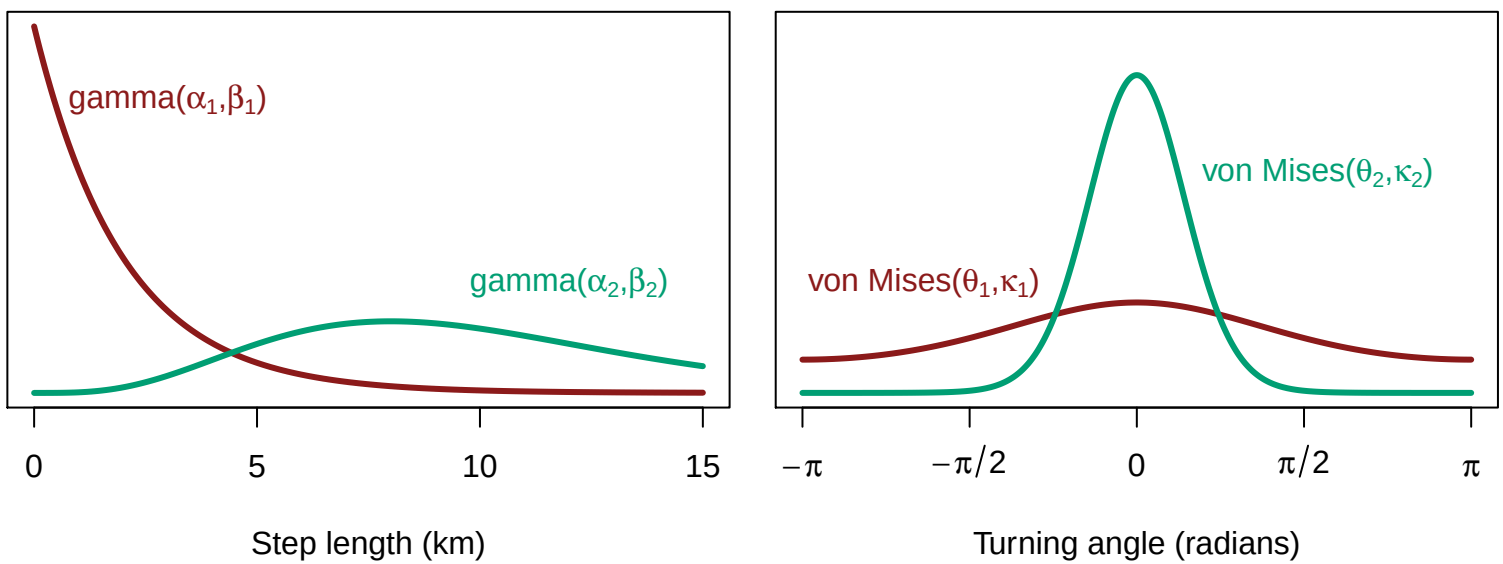
HMM parameters

There are two sets of parameters:

- The transition probabilities $\Pr(S_t = j | S_{t-1} = i)$, e.g.



- The state-dependent movement parameters, e.g.



Maximum likelihood estimation

The likelihood of the model gives a measure of how plausible the observed data are, for a given set of parameter values. We can maximise it with respect to the parameters, to obtain the **maximum likelihood estimates**.

The likelihood is

$$L(\Theta) = p(\boldsymbol{Z}_{1:n}|\Theta)$$

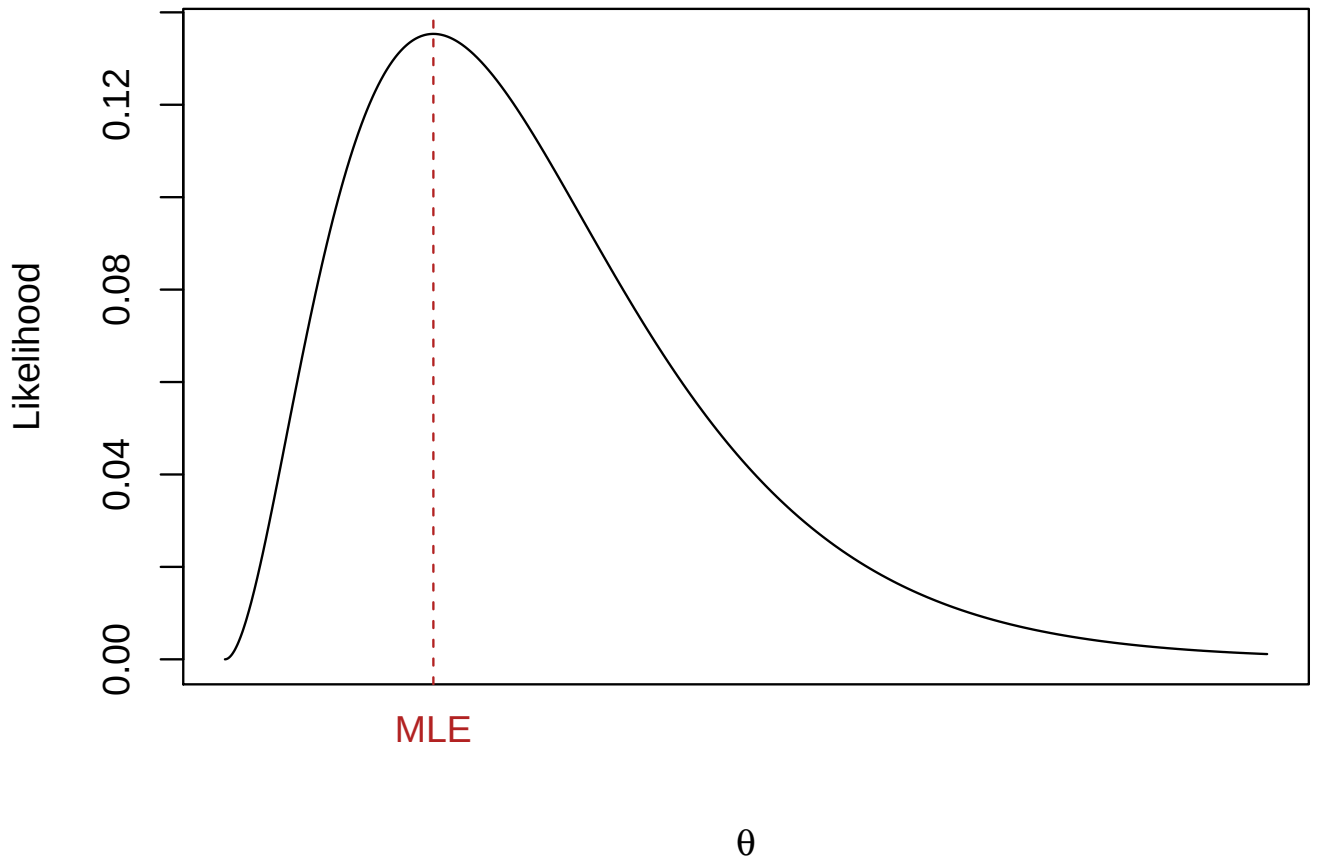
where

- $\boldsymbol{Z}_{1:n}$ is the set of all observed step lengths and turning angles
- Θ is the set of all parameters.

We want to find the value of Θ which maximises L.

Maximum likelihood estimation

Find the parameter value which maximises the likelihood:



In practice, this is usually done using numerical optimisers, like `optim` and `nlm` in R. They explore the parameter space to find the optimum.

`fitHMM`

In moveHMM, maximum likelihood estimation is implemented in the function `fitHMM`.

Inputs

- Data
- Number of states
- Initial parameter values

Outputs

- Estimated transition probabilities
- Estimated movement parameters

fitHMM

```
## define initial parameters
stepMean0 <- c(100, 500) # mean of step length distribu-
tion (one for each state)
stepSD0 <- c(100, 500) # SD of step length distribution
stepZM0 <- c(0.01, 0.01) # Zero mass of step length distri-
bution
angleMean0 <- c(0, 0) # mean of angle distribution
angleCon0 <- c(0.2, 2) # concentration of angle distribution

## fit 2-state model
m <- fitHMM(data = data, nbStates = 2,
            stepPar0 = c(stepMean0, stepSD0, stepZM0),
            anglePar0 = c(angleMean0, angleCon0))
```

- `data`: Data frame of locations, step lengths, turning angles...
- `nbStates`: Number of states
- `stepPar0`: Initial parameters for step length distribution
- `anglePar0`: Initial parameters for turning angle distribution

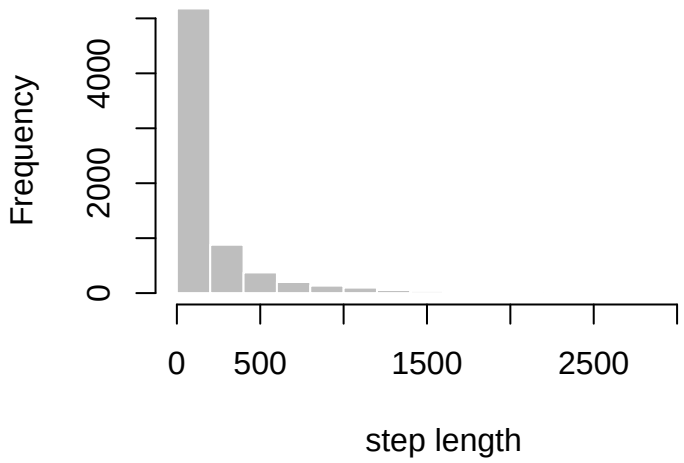
Fitted model

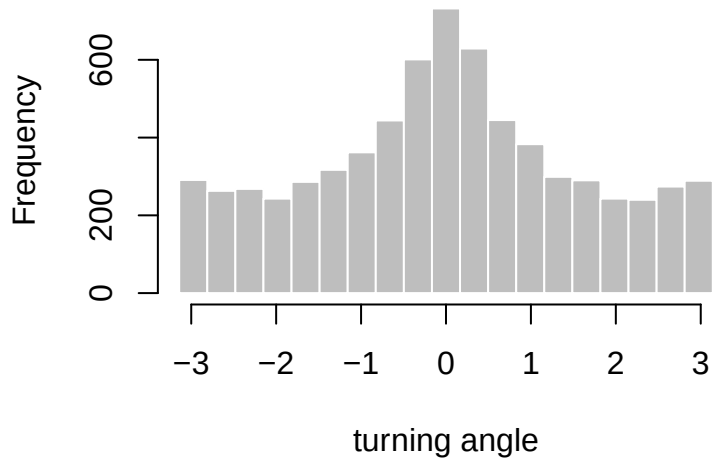
```
m
```

```
## Value of the maximum log-likelihood: -54005.64
##
## Step length parameters:
## ----------------------
##                 state 1      state 2
## mean      7.252747e+01 4.801008e+02
## sd        7.735607e+01 3.924185e+02
## zero-mass 3.997745e-04 3.505855e-11
##
## Turning angle parameters:
## ------------------------
##                   state 1     state 2
## mean          0.06356441 -0.01825904
## concentration 0.10078582  1.75848253
##
## Regression coeffs for the transition probabilities:
## --------------------------------------------------
##             1 -> 2   2 -> 1
## intercept -2.21887 -1.13441
##
## Transition probability matrix:
## -----------------------------
##           [,1]       [,2]
## [1,] 0.9019313 0.09806871
## [2,] 0.2433481 0.75665189
##
## Initial distribution:
## --------------------
## [1] 0.996807904 0.003192096
```

fitHMM: initial parameters

1. Plot histograms of step lengths and turning angles.





2. "What are some plausible values for the parameters?"

```
stepMean0 <- c(100, 500) # mean of step lengths
stepSD0 <- c(100, 500) # SD of step lengths
angleMean0 <- c(0, 0) # mean of turning angles
angleCon0 <- c(0.2, 2) # concentration of turning angles
```

3. Try many different initial parameters, possibly chosen at random.

fitHMM: number of states

There is no general method to select the "optimal" number of states.

1. Fit 2-state model, 3-state model, etc., and compare them:
   - Model comparison with AIC/BIC/...

```
AIC(mod2,mod3,mod4)
```

   - Model checking using pseudo-residuals.

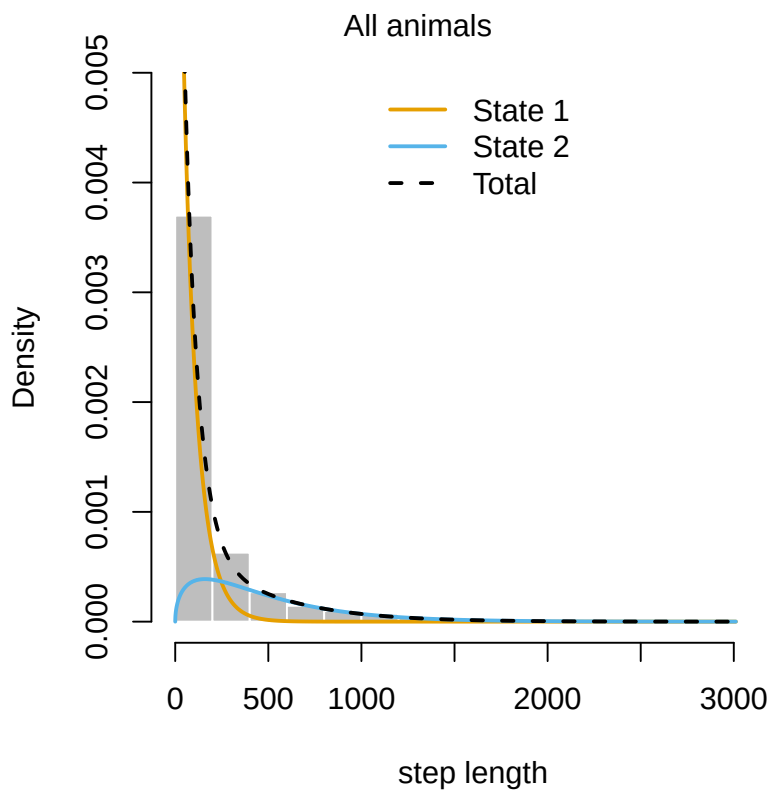
2. **Biological interpretation!**

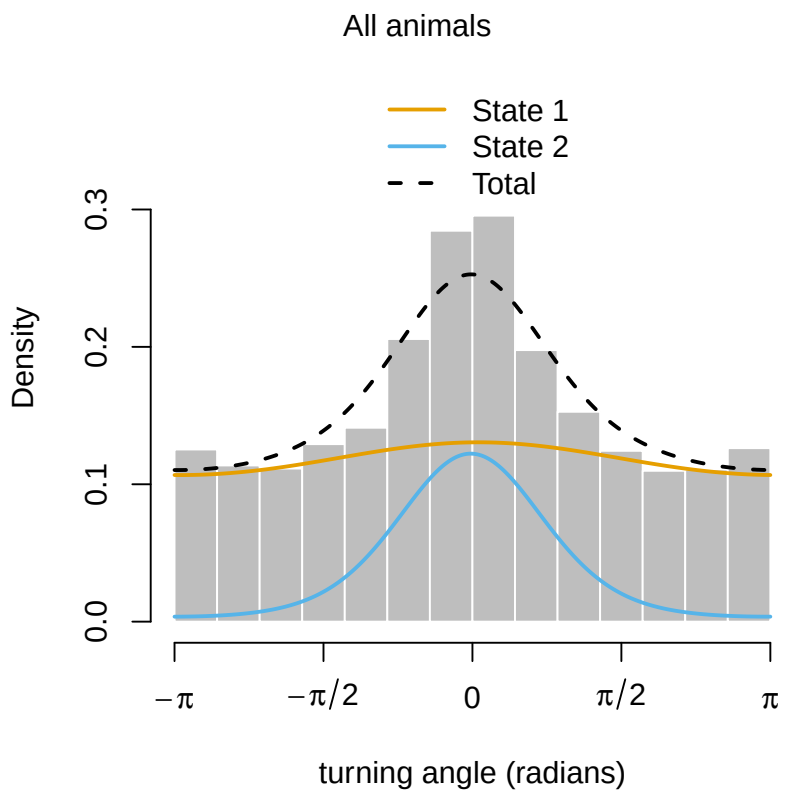
Pohle et al. (2017). Selecting the number of states in hidden Markov models: pragmatic solutions illustrated using animal movement, *JABES*.

1. Prepare the data
2. Fit the model
3. Visualise the results

Plot fitted model

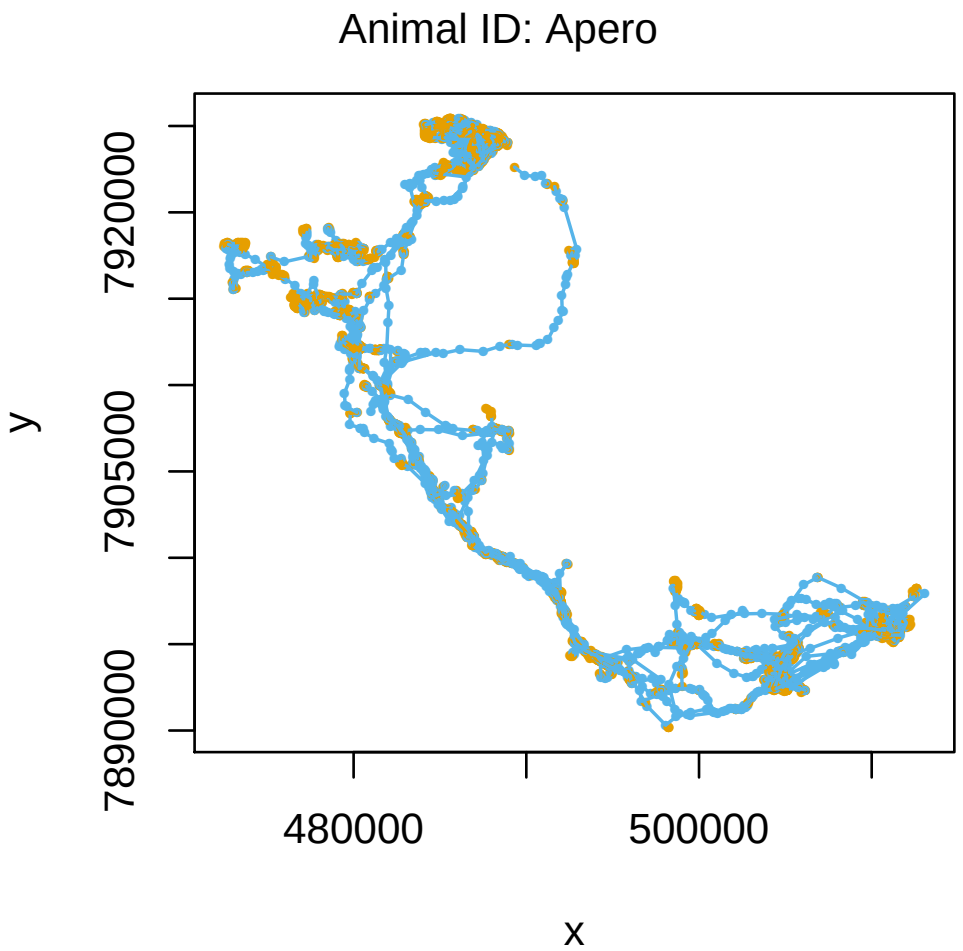
```
plot(m)
```





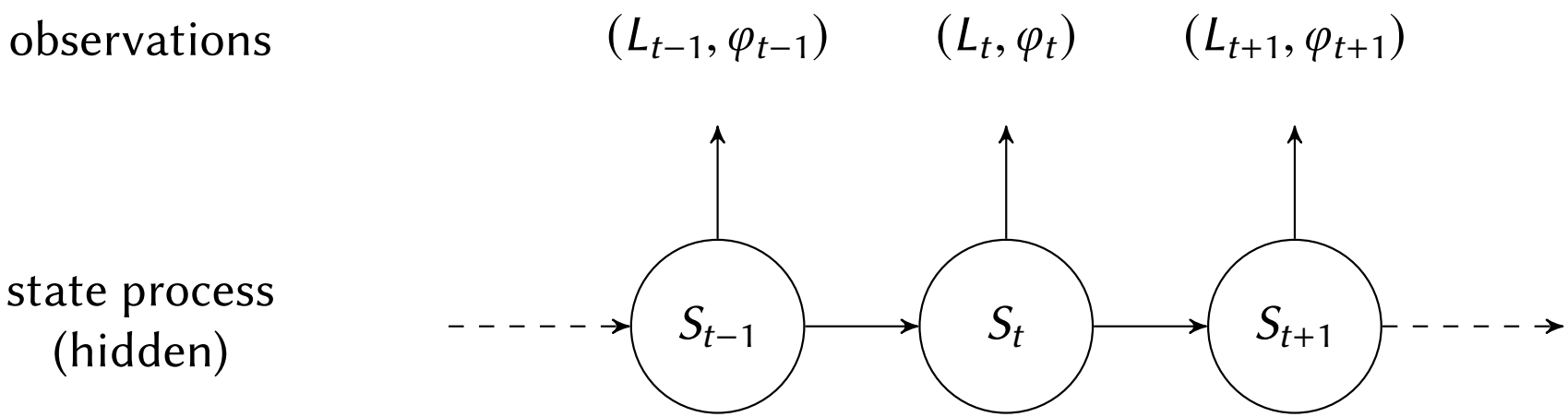
Plot fitted model

```
plot(m)
```



Estimate hidden states

From a fitted model, we can estimate the hidden states, i.e. the behavioural state of the animal at each time step.



Estimate hidden states

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

## [1] 1 1 1 1 1 1

## State probabilities
sp <- stateProbs(m)
head(sp)

##              [,1]          [,2]
## [1,] 0.9994970 0.0005029516
## [2,] 0.9783066 0.0216933632
## [3,] 0.9967452 0.0032547587
## [4,] 0.9998514 0.0001486322
## [5,] 0.9941226 0.0058773894
## [6,] 0.9852015 0.0147985372
```

Model checking

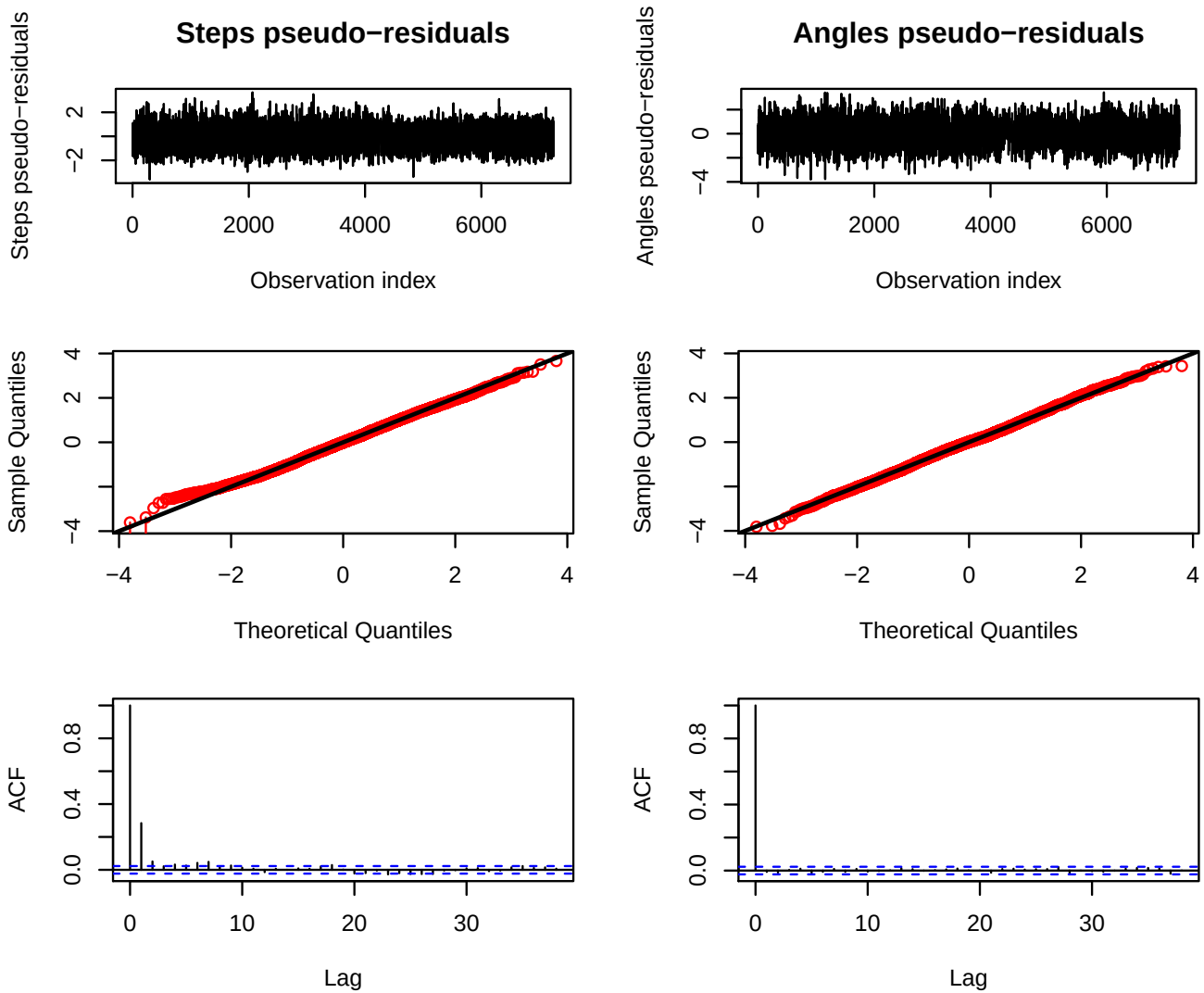
- To assess model fit in HMMs, we use **pseudo-residuals**.
- If the model is good, then the pseudo-residuals follow a normal distribution.
- Any deviation from normality indicates lack of fit.

In moveHMM, there are two relevant functions,

- `pseudoRes`: get pseudo-residuals
- `plotPR`: quantile-quantile plots of pseudo-residuals against normal distribution

Pseudo-residuals

```
plotPR(m)
```



Include covariates

Include covariates

In the function `fitHMM`, the argument `formula` specifies the covariate formula for the transition probabilities.

- **Distance to water:**

```
m.d2w <-
    fitHMM(data = data, nbStates = 2,
           stepPar0 = c(stepMean0, stepSD0, stepZM0),
           anglePar0 = c(angleMean0, angleCon0),
           formula = ~ d2w)
```

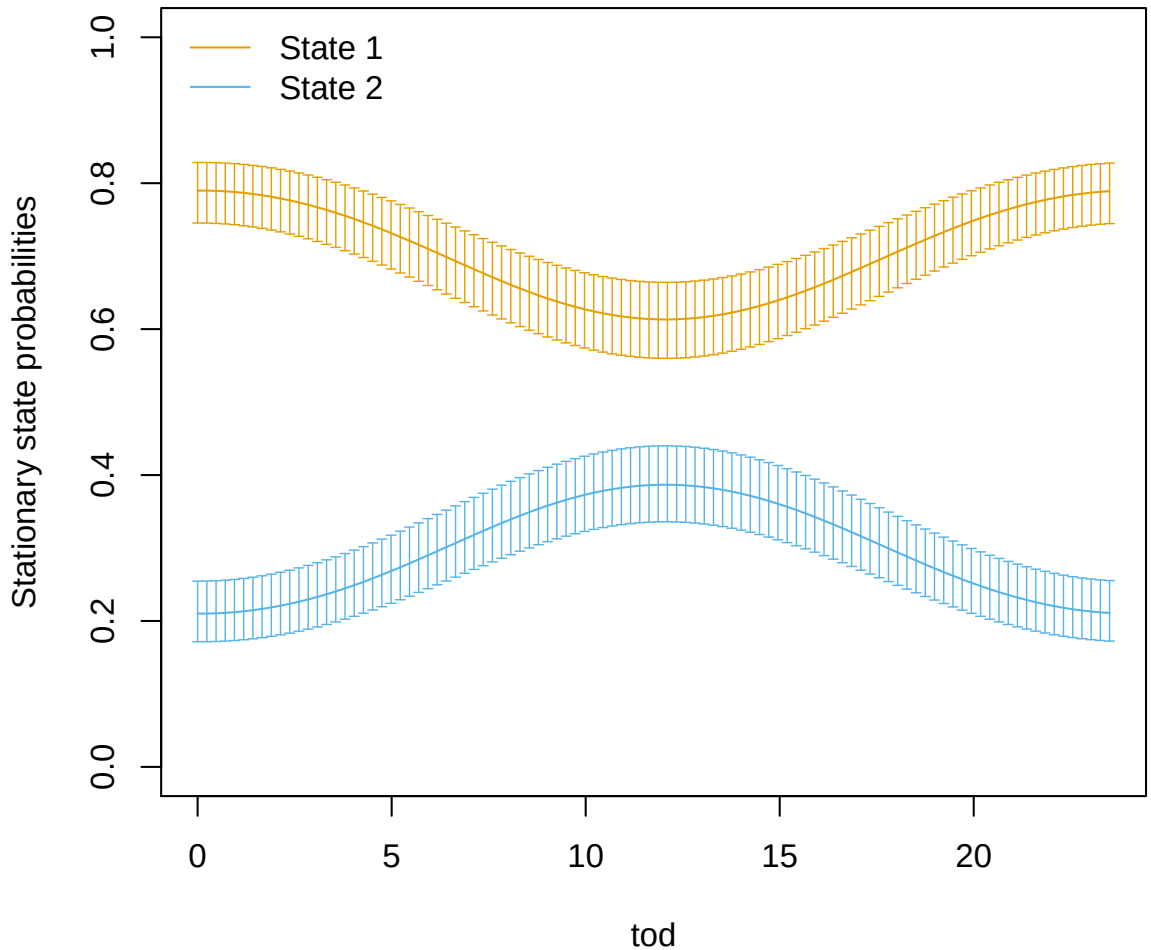
- **Time of day:**

```
m.tod <-
    fitHMM(data = data, nbStates = 2,
           stepPar0 = c(stepMean0, stepSD0, stepZM0),
           anglePar0 = c(angleMean0, angleCon0),
         formula = ~ cos(2*pi*tod/24) + sin(2*pi*tod/24))
```

Plot covariate effect

The stationary state probabilities give the probability of being in each state in the long run, for different values of the covariate.

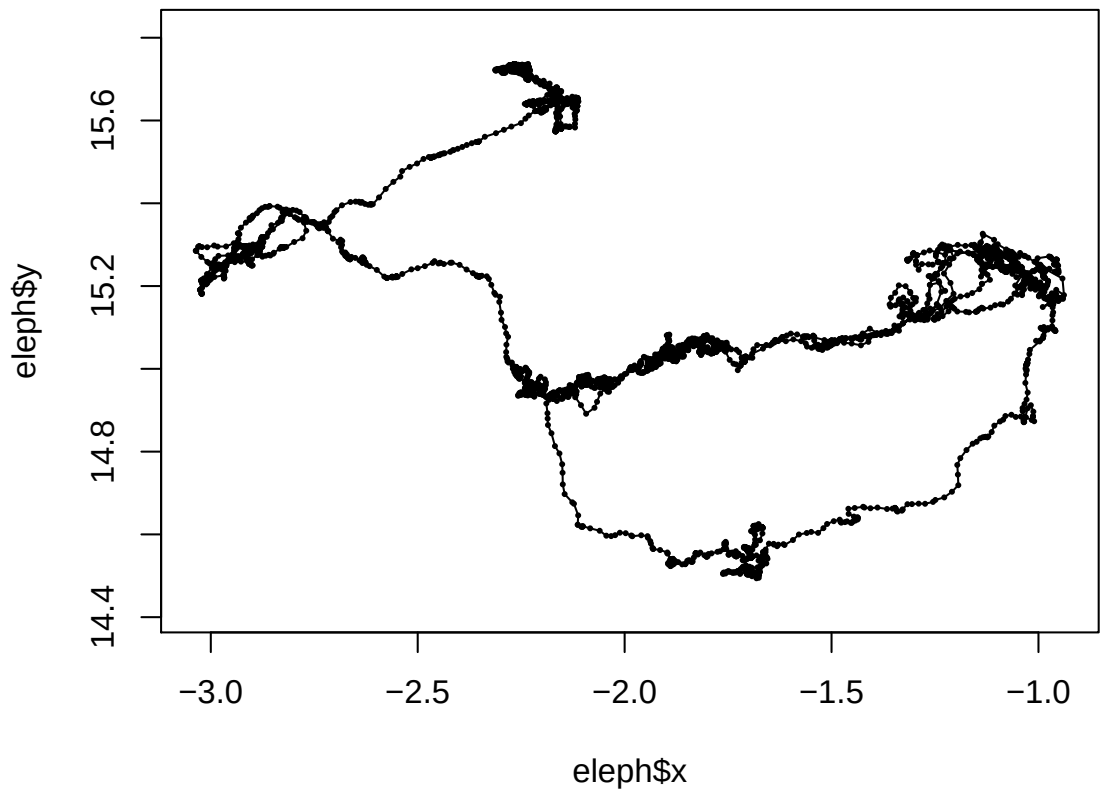
```
plotStationary(m.tod, plotCI = TRUE)
```



Practical session

Elephant data set

```
eleph <- read.csv("code/elephant.csv")
plot(eleph$x, eleph$y, type = "o", pch = 20, cex = 0.4, asp = 1)
```



⇝ 5000 hourly locations

Elephant data set

```
head(eleph)
```

```
##   ID          x         y temp tod
## 1  1 -2.160167 15.65350   38  17
## 2  1 -2.160075 15.65452   35  18
## 3  1 -2.159902 15.65451   32  19
## 4  1 -2.159435 15.65489   30  20
## 5  1 -2.158113 15.65512   29  21
## 6  1 -2.157848 15.65461   28  22
```

- ID
- Longitude-latitude locations
- Covariates: temperature and time of day