Analysing animal movement data in R

Théo Michelot

7 February 2022

This document presents the analysis of GPS movement tracks from African elephants at Kruger National Park, kindly made available on the Movebank Data Repository by Slotow, Thaker, and Vanak (2019) and previously analysed by Thaker et al. (2019).

```
# Plotting package
library(ggplot2)
theme_set(theme_bw())
# Movement modelling packages
library(momentuHMM)
library(foieGras)
library(adehabitatLT)
# GIS packages
library(sf)
library(sp)
# Load functions for later
source("utility_functions.R")
```

Data download and initial preparation

We first load the data directly from the URL (this requires an Internet connection). For the purposes of this demonstration, we only keep 2000 locations to reduce the computational cost of model fitting. We take the first 1000 locations from two tracks, to showcase that it is possible to use the analysis tools on data sets with multiple tracks (here identified by the variable `ID`).

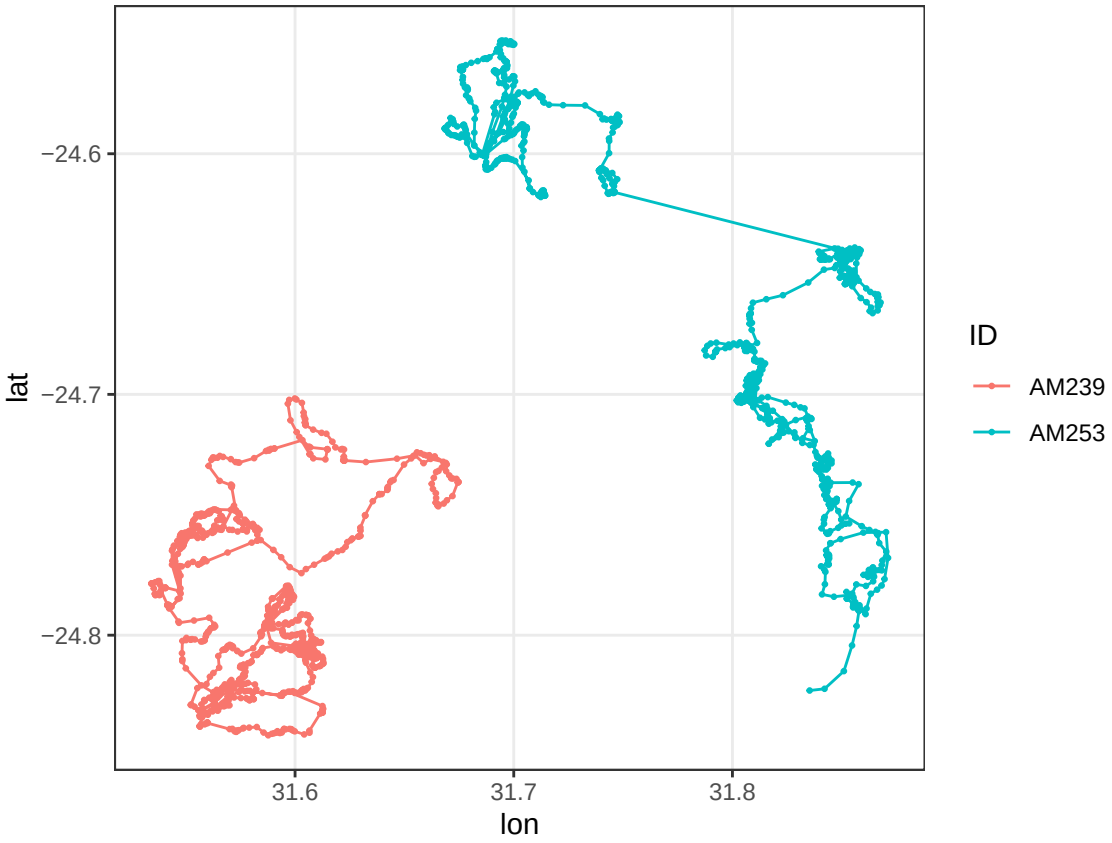
```
# Load data from Movebank URL
URL <- paste0("https://www.datarepository.movebank.org/bitstream/handle/10255/",
              "move.981/ThermochronTracking%20Elephants%20Kruger%202007.csv")
raw <- read.csv(url(URL))

# Keep relevant columns: ID, time, longitude, latitude, temperature
data_all <- raw[, c(9, 3, 4, 5, 6)]
colnames(data_all) <- c("ID", "time", "lon", "lat", "temp")
data_all$time <- as.POSIXct(data_all$time, format = "%Y-%m-%d %H:%M:%S", tz = "GMT")

# Just keep 2000 observations to save time with model fitting
data <- data_all[c(which(data_all$ID == unique(data_all$ID)[5])[1:1000],
                   which(data_all$ID == unique(data_all$ID)[6])[1:1000]),]

head(data)
```

```
         ID                time      lon       lat temp
68156 AM239 2007-08-13 00:30:00 31.60174 -24.80802   20
68157 AM239 2007-08-13 01:02:00 31.60172 -24.80802   19
68158 AM239 2007-08-13 01:30:00 31.60187 -24.80781   19
```

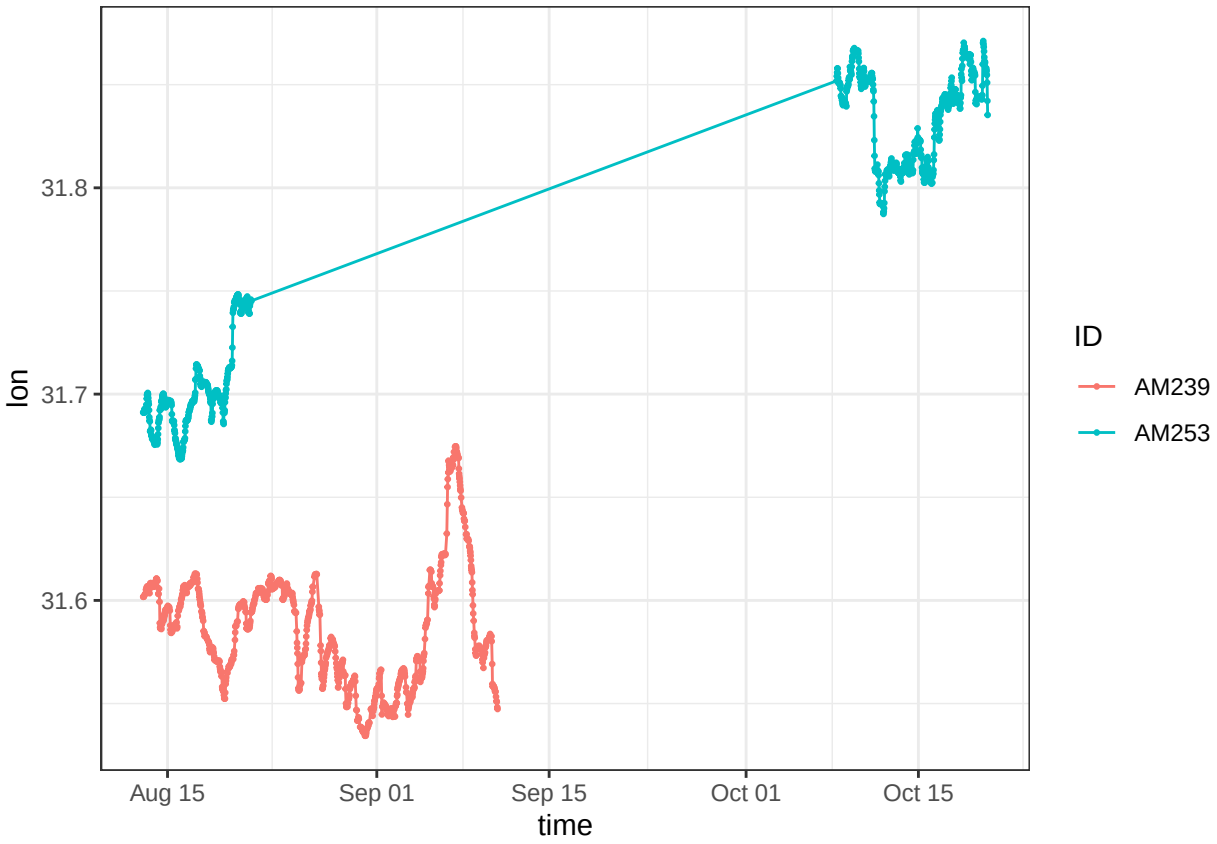
```
68159 AM239 2007-08-13 02:00:00 31.60188 -24.80788   19
68160 AM239 2007-08-13 03:30:00 31.60276 -24.80774   19
68161 AM239 2007-08-13 04:30:00 31.60319 -24.80783   18
```

To visualise the data, we can plot the 2-d trajectories (longitude vs latitude), or each coordinate against time to identify potential gaps. For example, there is one very long gap in the track of individual AM253.

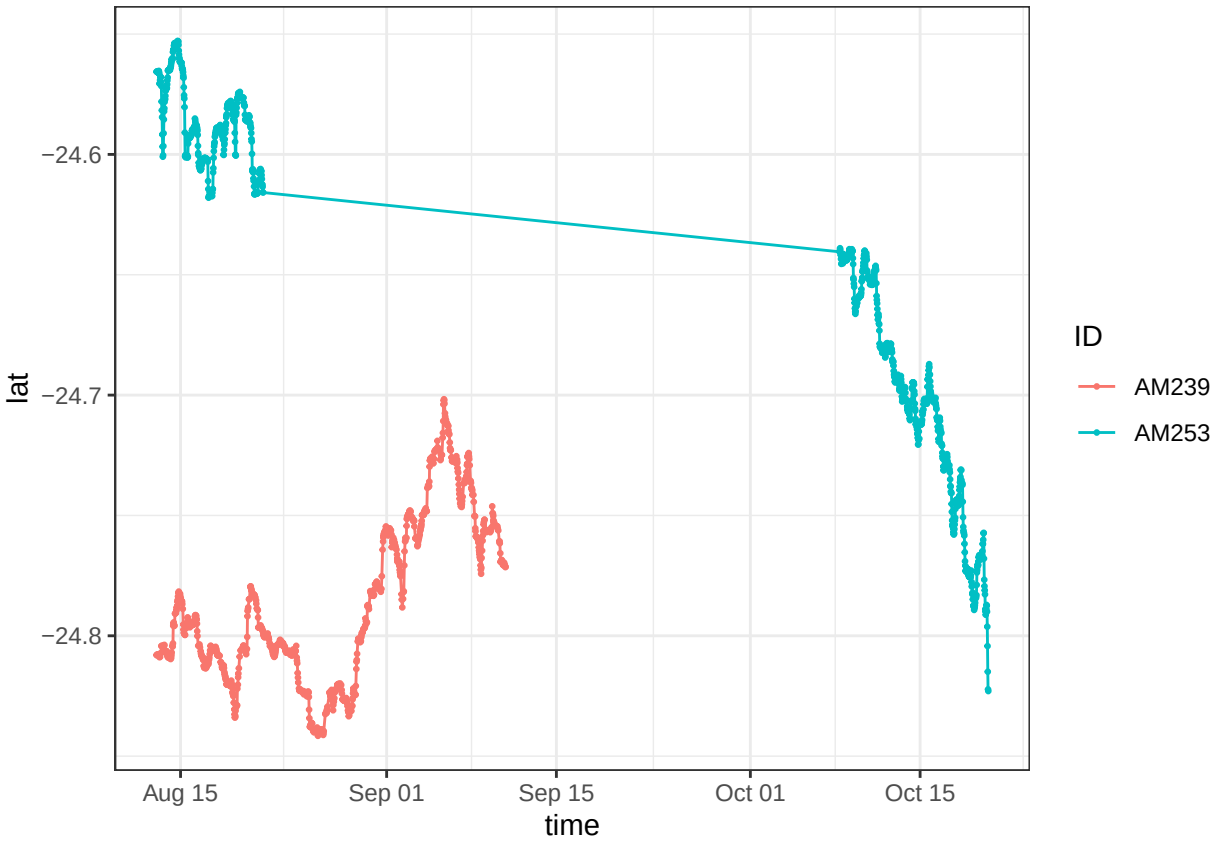
```
# Plot latitude vs longitude (Mercator projection)
ggplot(data, aes(lon, lat, col = ID)) +
    geom_point(size = 0.5) + geom_path() +
    coord_map("mercator")
```




```
# Longitude vs time
ggplot(data, aes(time, lon, col = ID)) +
    geom_point(size = 0.5) + geom_path()
```

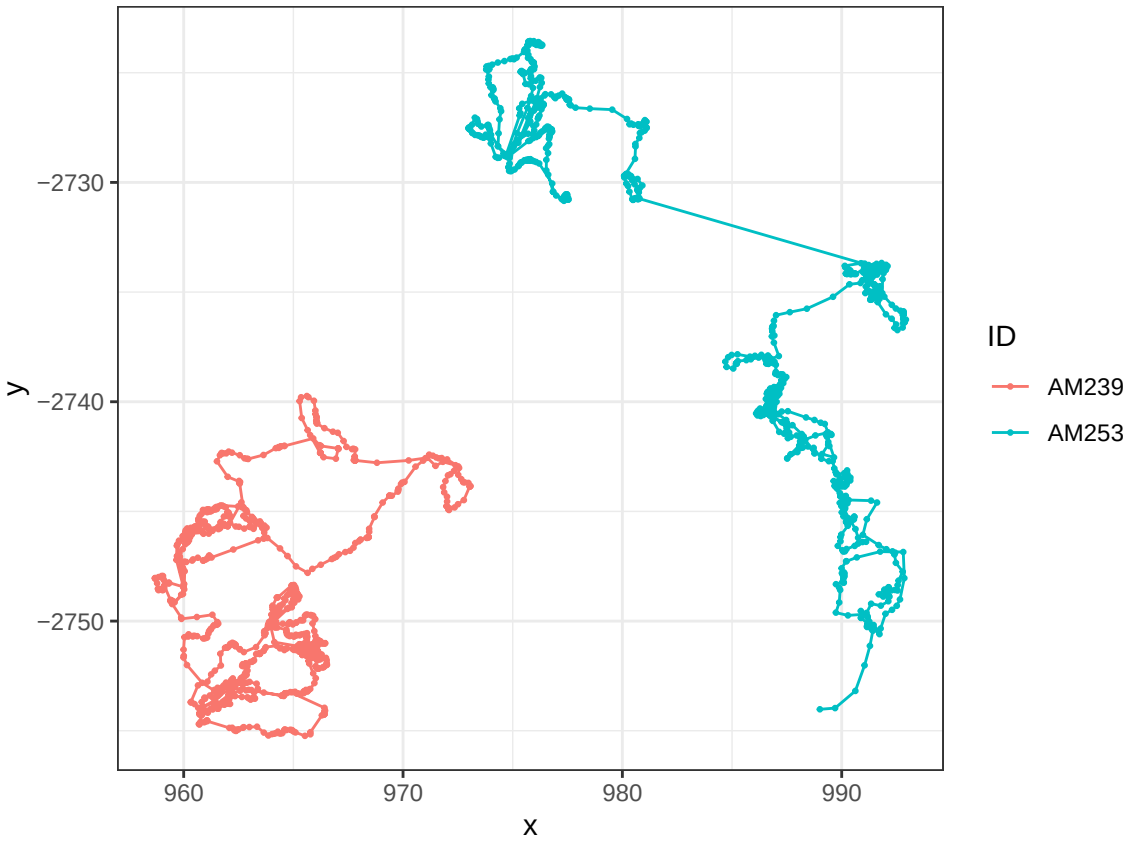



```
# Latitude vs time
ggplot(data, aes(time, lat, col = ID)) +
    geom_point(size = 0.5) + geom_path()
```



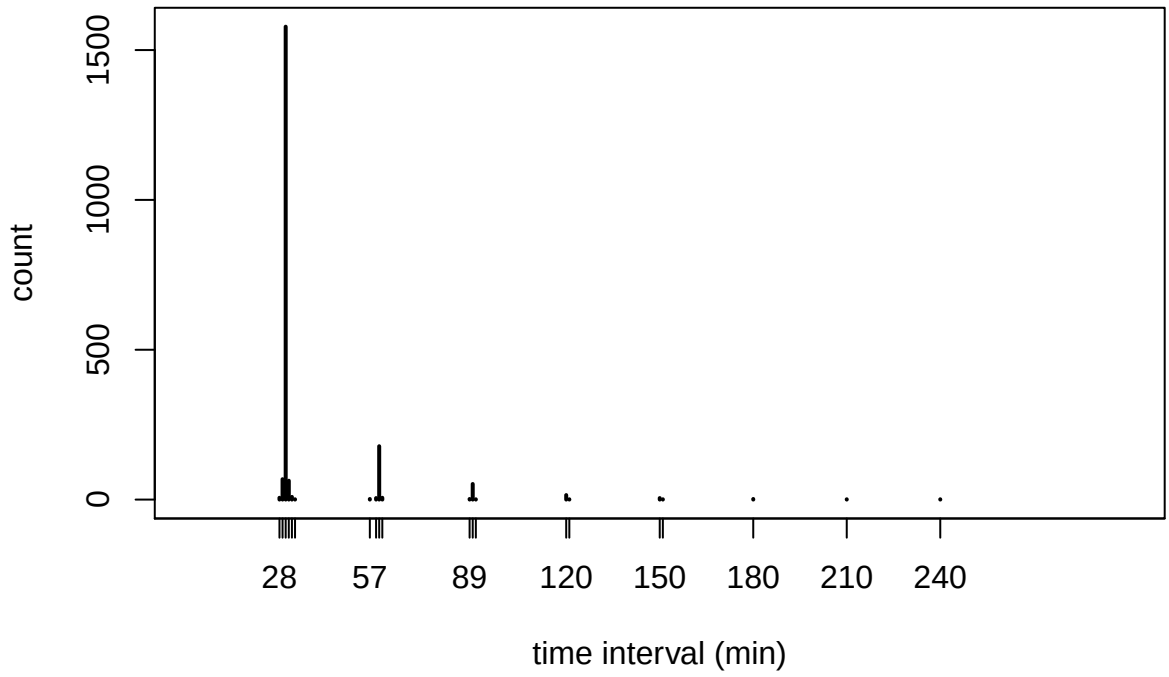
Many analyses require Easting-Northing locations rather than longitude-latitude. We can obtain projected (Easting-Northing) points using the packages sp and sf. (I know very little about GIS, there might well be a better way to do this!) Plotting the projected tracks is a good sanity check; here, they look reassuringly similar to the longitude/latitude plot.

```
# Project to UTM
llcoord <- st_as_sf(data[, c("lon", "lat")], coords = c("lon", "lat"),
                    crs = CRS("+proj=longlat +datum=WGS84"))
utmcoord <- st_transform(llcoord, crs = CRS("+proj=utm +zone=35 +datum=WGS84"))

# Add Easting-Northing to data (in km)
data[, c("x", "y")] <- st_coordinates(utmcoord)/1000

# Plot Northing vs Easting
ggplot(data, aes(x, y, col = ID)) +
    geom_point(size = 0.5) + geom_path() +
    coord_equal()
```



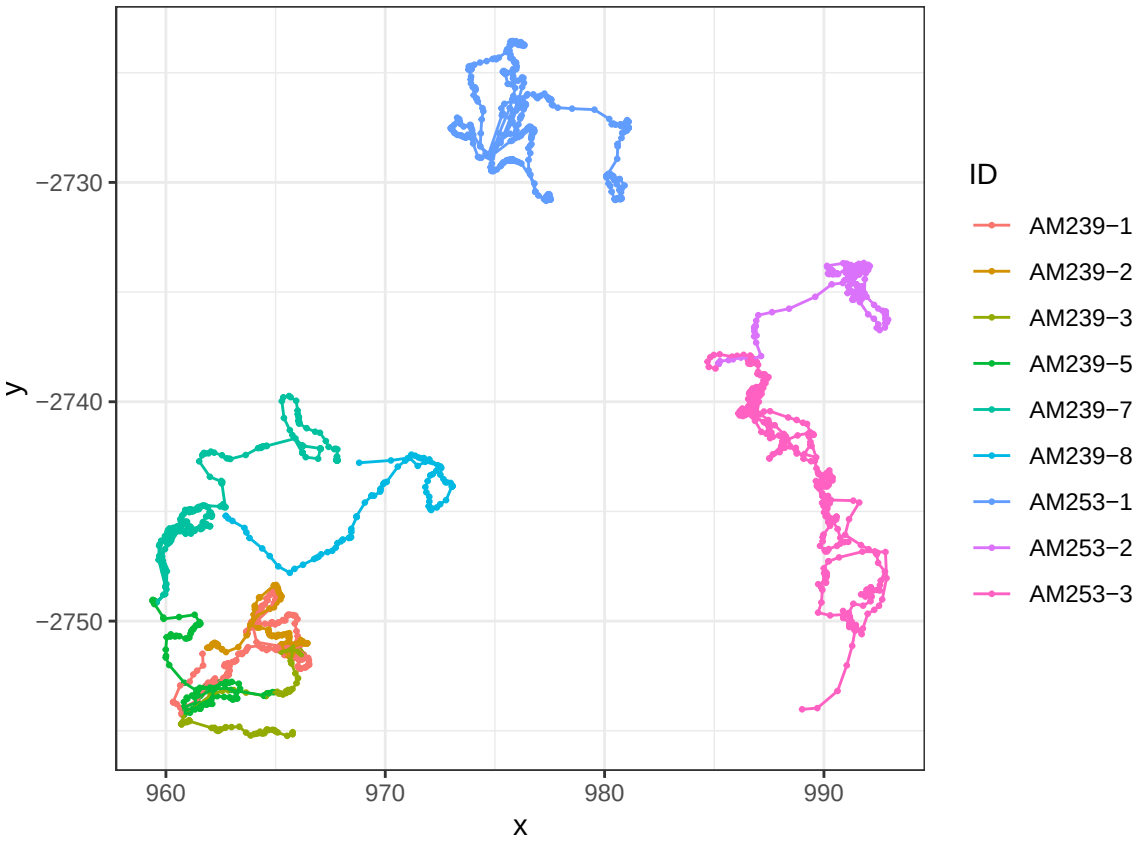
The base time interval between observations was 30 minutes in this study, but there were often missing observations, resulting in longer intervals (e.g., 1 hour, 1.5 hour, etc). There was one very long gap, lasting over a month.

```
# Table of time intervals in data
plot(table(diff(data$time)), xlim = c(0, 300),
     xlab = "time interval (min)", ylab = "count")
```



For many statistical approaches, long gaps in a track can be a problem, e.g., if we try to interpolate locations over a finer time grid (with high uncertainty during long gaps). The first step of data preparation is therefore often to split tracks where there are too many missing locations. The exact cutoff is very study-dependent; here, we decide that tracks should be split at gaps longer than 2 hours. In practice, this means re-defining the "ID" variable which identifies each track, to change at every long gap. We do this using the utility function `split_at_gap`, loaded from a separate file.

```
# Use function from utility_function.R to split data at gaps > 2 hours
data_split <- split_at_gap(data = data, max_gap = 2*60, shortest_track = 24*60)
ggplot(data_split, aes(x, y, col = ID)) +
    geom_point(size = 0.5) + geom_path() +
    coord_equal()
```



Regular time intervals using NAs

There is still some temporal irregularity in the data, as we kept gaps shorter than 2 hours. Hidden Markov models can be applied in cases where there are missing observations on an otherwise regular grid, like this. The package momentuHMM expects that such missing locations be including as rows of `NA` (the R value for missing data). To do this, we use the function `setNA` from the package adehabitatLT.

```
# Create adehabitat trajectory padded with NAs
data_ade <- setNA(ltraj = as.ltraj(xy = data_split[, c("x", "y")],
                                   date = data_split$time,
                                   id = data_split$ID),
                  date.ref = data_split$time[1],
                  dt = 30, tol = 5, units = "min")

# Transform back to dataframe
data_na <- ld(data_ade)[, c("id", "x", "y", "date")]
colnames(data_na) <- c("ID", "x", "y", "time")

# Add temperatures for non-missing locations
data_na$temp <- NA
data_na$temp[which(!is.na(data_na$x))] <- data_split$temp

head(data_na, 10)
```

```
        ID        x         y                time temp
1  AM239-1 965.4056 -2751.543 2007-08-13 00:30:00   20
2  AM239-1 965.4034 -2751.543 2007-08-13 01:02:00   19
3  AM239-1 965.4195 -2751.521 2007-08-13 01:30:00   19
```

```
4  AM239-1 965.4203 -2751.528 2007-08-13 02:00:00   19
5  AM239-1       NA        NA 2007-08-13 02:30:00   NA
6  AM239-1       NA        NA 2007-08-13 03:00:00   NA
7  AM239-1 965.5100 -2751.516 2007-08-13 03:30:00   19
8  AM239-1       NA        NA 2007-08-13 04:00:00   NA
9  AM239-1 965.5531 -2751.527 2007-08-13 04:30:00   18
10 AM239-1 965.5582 -2751.530 2007-08-13 05:00:00   18
```

The momentuHMM workflow

Data preparation

The data are now ready to be used in momentuHMM. The last step of data preparation is to derive step lengths and turning angles from the locations, using the function `prepData` from momentuHMM.

```
# Prepare data for HMM (compute step lengths and turning angles)
data_hmm1 <-prepData(data_na, type = "UTM", covNames = "temp")
```

```
Warning in prepData.default(data_na, type = "UTM", covNames = "temp"): There
are 316 missing covariate values. Each will be replaced by the closest available
value.
```

```
head(data_hmm1, 10)
```

```
        ID        step     angle                time        x         y temp
1  AM239-1 0.002128067        NA 2007-08-13 00:30:00 965.4056 -2751.543   20
2  AM239-1 0.027887615 -2.151601 2007-08-13 01:02:00 965.4034 -2751.543   19
3  AM239-1 0.007837343 -2.431126 2007-08-13 01:30:00 965.4195 -2751.521   19
4  AM239-1          NA        NA 2007-08-13 02:00:00 965.4203 -2751.528   19
5  AM239-1          NA        NA 2007-08-13 02:30:00       NA        NA   19
6  AM239-1          NA        NA 2007-08-13 03:00:00       NA        NA   19
7  AM239-1          NA        NA 2007-08-13 03:30:00 965.5100 -2751.516   19
8  AM239-1          NA        NA 2007-08-13 04:00:00       NA        NA   19
9  AM239-1 0.005541090        NA 2007-08-13 04:30:00 965.5531 -2751.527   18
10 AM239-1          NA        NA 2007-08-13 05:00:00 965.5582 -2751.530   18
```

Model fitting

We can now fit an HMM, which is done with the function `fitHMM`. Many different model formulations can be specified, but here we focus on just a few options (the most useful ones). For the purpose of the study, and based on previous knowledge about elephants, we would like to model their movement as a mixture of two different movement types; for this purpose, we will fit a 2-state HMM (i.e., the behavioural state process switches between 2 different values).

At a minimum, two things need to be specified before model fitting:

- Observation distributions, i.e., a model for each observed variable. Here, we assume that step lengths follow a gamma distribution in each state, and turning angles a von Mises distribution. (The von Mises distribution is a circular distribution, defined for angular variables.)
- Initial parameter values. These are needed because the model is fitted using a numerical optimisation procedure, which requires a starting point. The optimiser might fail to converge to the best model if the starting point is chosen badly, so this is often a difficult step. In practice, the general idea is to select "plausible" values, based on what we know of the data. Here, we need four initial parameters for the step length distribution: mean step length in state 1 (in km), mean step length in state 2, step length standard deviation in state 1, and step length standard deviation in state 2. We also need two turning angle parameters: the concentration of the von Mises distribution in each state, which measures how directed the movement is.

```r
# Observation distributions (step lengths and turning angles)
dist <- list(step = "gamma", angle = "vm")

# Initial parameters
# (step mean 1, step mean 2, step SD 1, step SD 2) and (angle concentration 1, angle concentration 2)
Par0_2s <- list(step = c(0.05, 0.2, 0.05, 0.2), angle = c(0.1, 3))

# Fit a 2-state HMM
hmm1 <- fitHMM(data_hmm1, nbStates = 2, dist = dist, Par0 = Par0_2s)
```

Some guidance for selecting initial parameter values is provided in a vignette for the R package moveHMM, on which momentuHMM is based: A short guide to choosing initial parameter values for the estimation in moveHMM. Although the exact syntax is slightly different between the two packages, the general advice can be applied similarly in a momentuHMM analysis.

Model visualisation

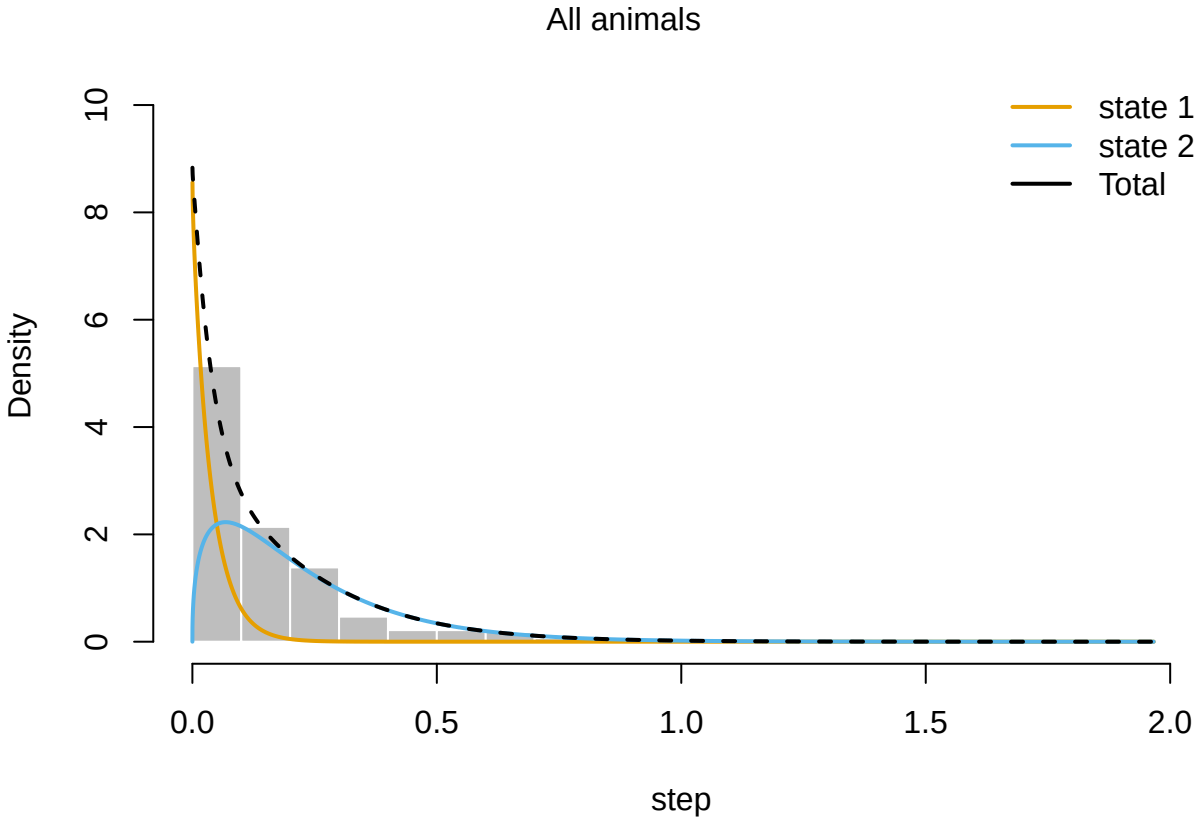
Once the model is fitted, we can print the parameter estimates, as well as plot some outputs.

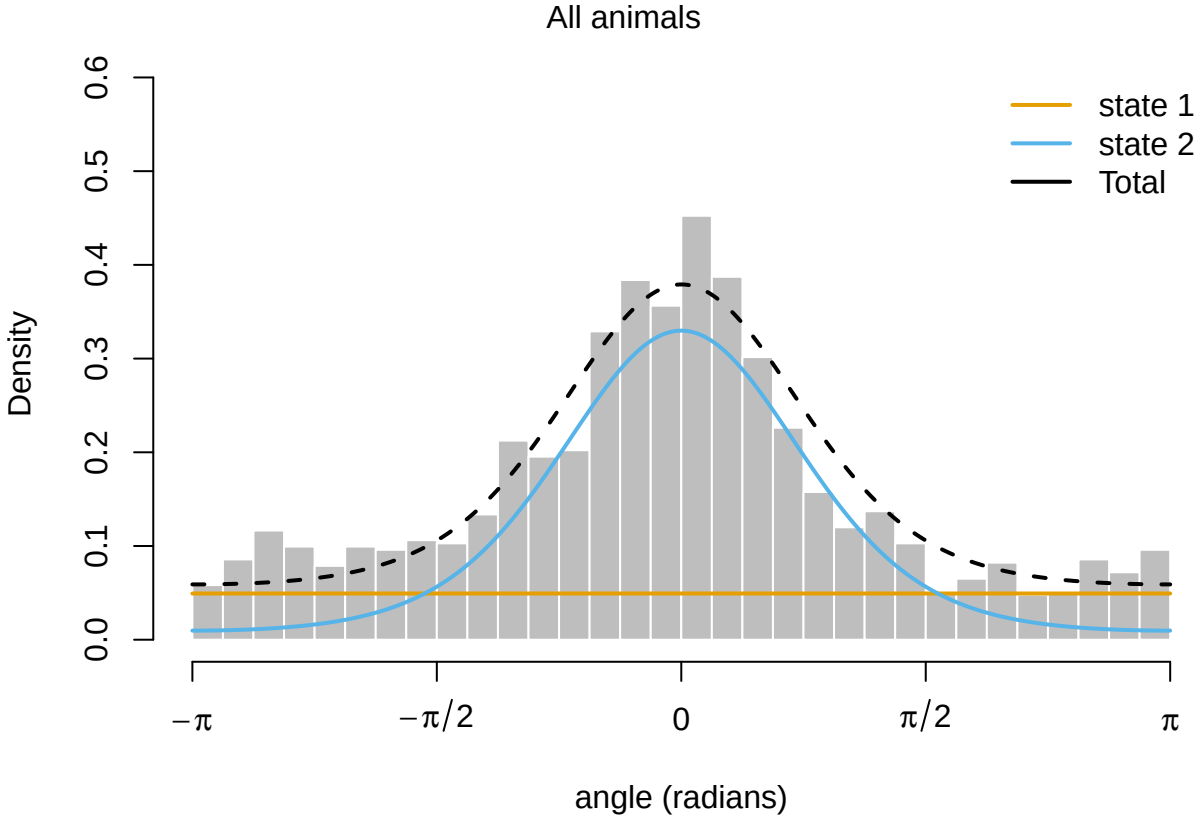
```r
# Print parameter estimates
hmm1
```

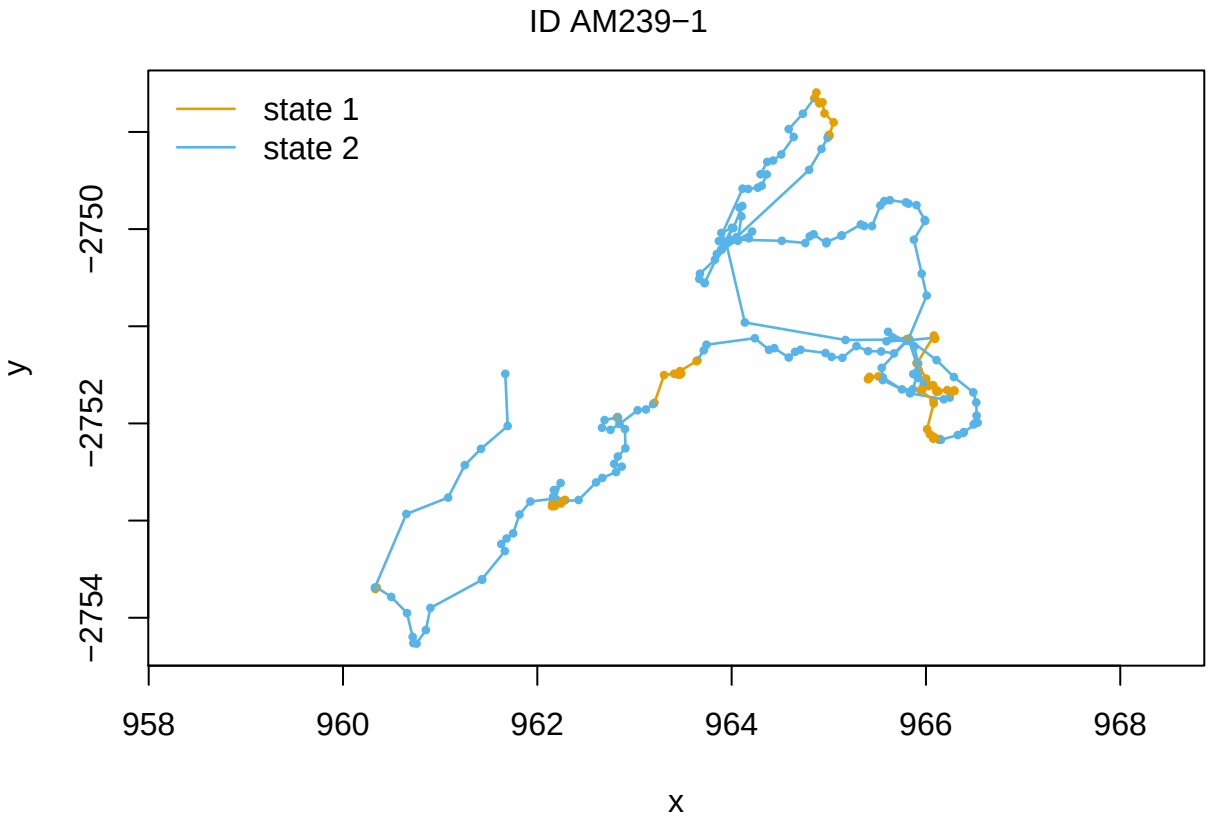
```
Value of the maximum log-likelihood: -740.1359


step parameters:
----------------
        state 1   state 2
mean 0.03917816 0.2262832
sd   0.03962630 0.1892416

angle parameters:
-----------------
                   state 1  state 2
mean          0.000000e+00 0.000000
concentration 1.868315e-08 1.763103

Regression coeffs for the transition probabilities:
---------------------------------------------------
               1 -> 2   2 -> 1
(Intercept) -1.595185 -2.21603

Transition probability matrix:
------------------------------
           state 1   state 2
state 1 0.83134441 0.1686556
state 2 0.09832017 0.9016798

Initial distribution:
---------------------
  state 1   state 2
0.5348213 0.4651787
```

```r
# Plot estimated distributions and state-coloured tracks
plot(hmm1, breaks = 25, ask = FALSE)
```

```
Decoding state sequence... DONE
```







The two states are very distinct: state 1 captures slow, undirected movement, whereas state 2 captures faster, more directed movement. The interpretation in terms of behaviour requires good knowledge of the study system, so I will not attempt an uneducated guess!

Mdel formulation: number of states

If we would like to fit a 3-state model, to describe movement driven by three different states, we just need to re-define the initial parameter values (because one value is required for each state).

```
# Initial parameters for 3-state model
Par0_3s <- list(step = c(0.02, 0.1, 0.3, 0.02, 0.1, 0.3),
                angle = c(0.01, 0.1, 3))

# Fit 3-state HMM
hmm2 <- fitHMM(data_hmm1, nbStates = 3, dist = dist, Par0 = Par0_3s)
```

Model fitting takes a little longer, because more parameters need to be estimated.

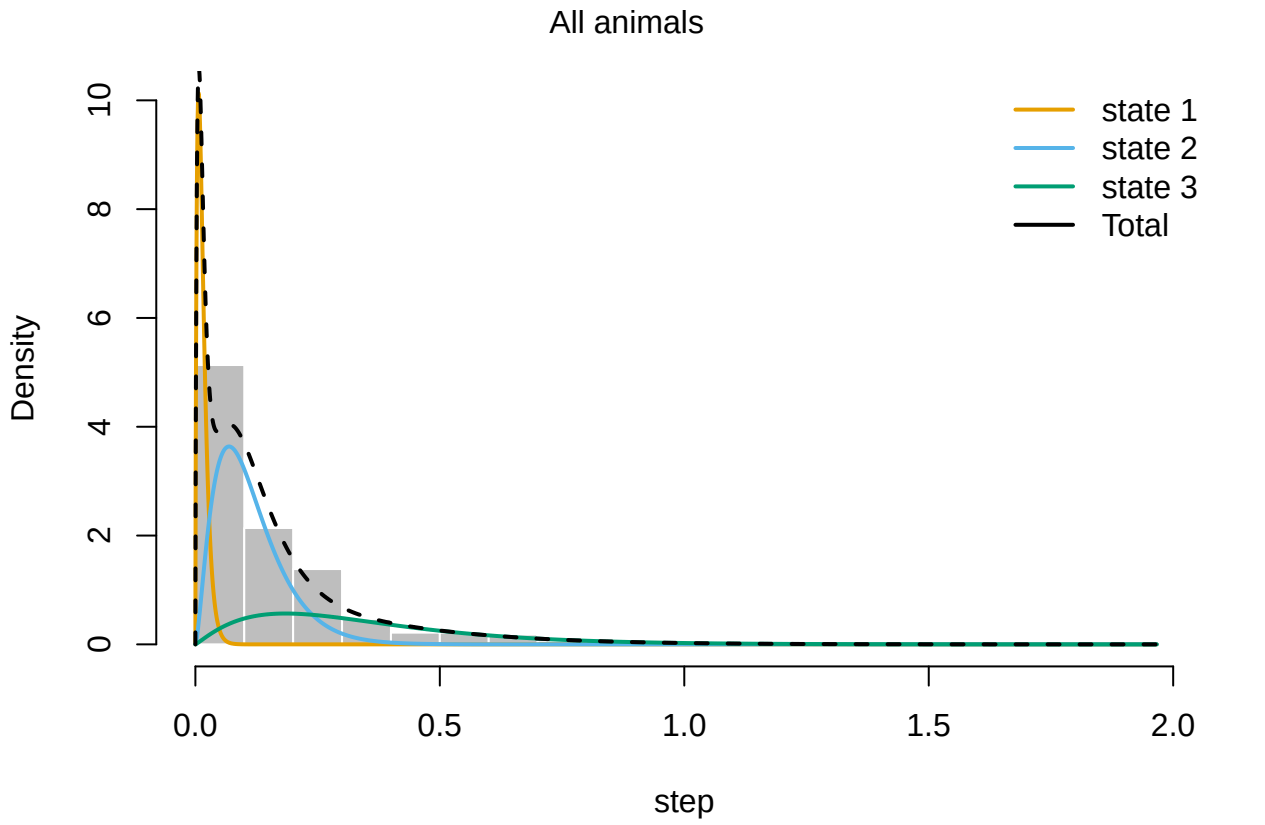
```
hmm2
```

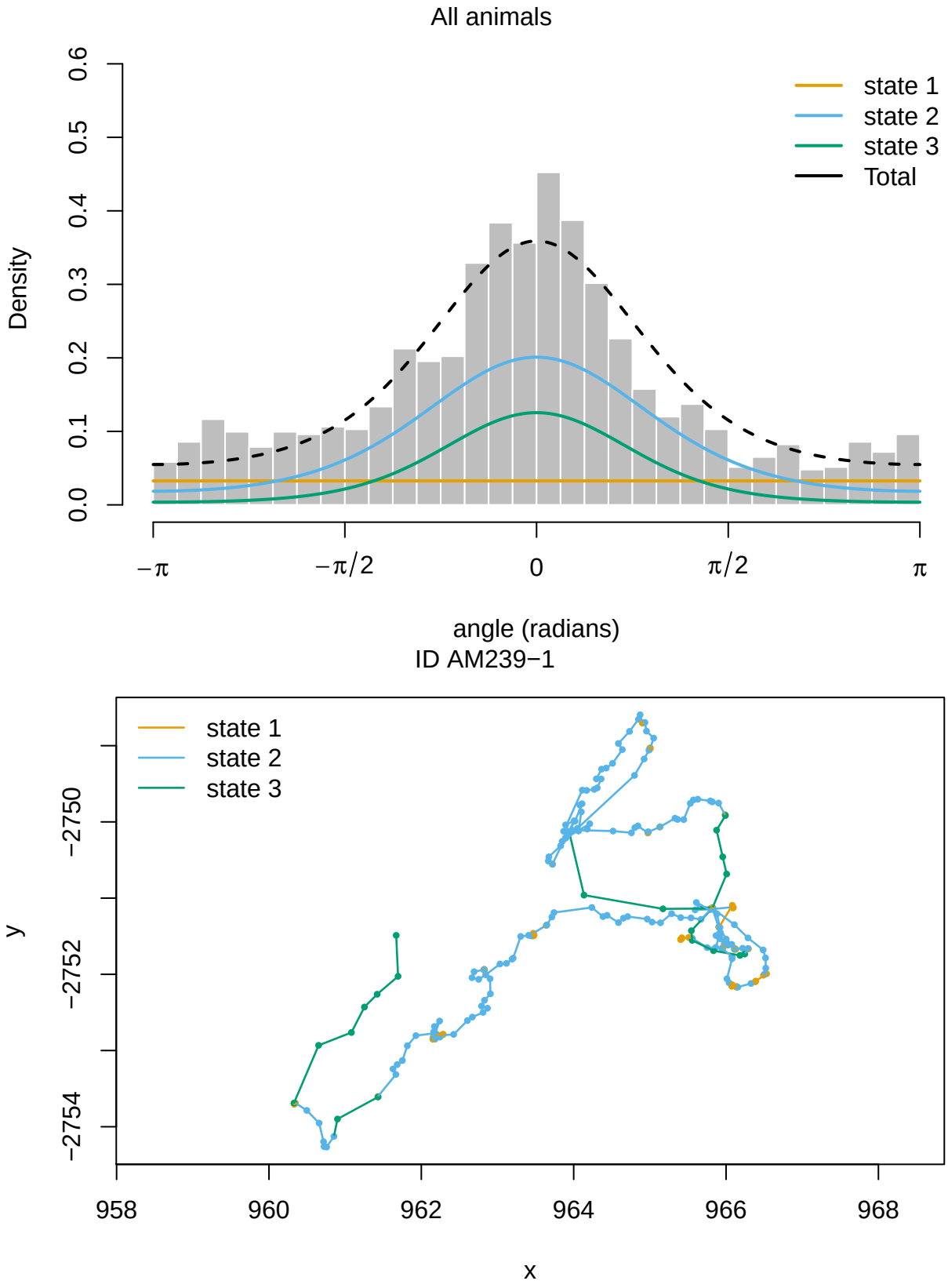
```
Value of the maximum log-likelihood: -610.882


step parameters:
----------------
        state 1    state 2   state 3
mean 0.01470994 0.11334480 0.3451753
sd   0.01090030 0.07105182 0.2356279

angle parameters:
```

```
-----------------
                   state 1  state 2  state 3
mean          0.000000e+00 0.000000 0.000000
concentration 9.613151e-10 1.190875 1.759863

Regression coeffs for the transition probabilities:
----------------------------------------------------
                1 -> 2   1 -> 3    2 -> 1    2 -> 3    3 -> 1    3 -> 2
(Intercept) -0.9407718 -1.96126 -1.592464 -2.686778 -3.277086 -1.764935

Transition probability matrix:
------------------------------
           state 1   state 2    state 3
state 1 0.65316468 0.2549475 0.09188787
state 2 0.15998425 0.7864579 0.05355780
state 3 0.03121593 0.1416105 0.82717362

Initial distribution:
---------------------
  state 1   state 2   state 3
0.4709703 0.1935056 0.3355241
```

```
plot(hmm2, breaks = 25, ask = FALSE)
```

```
Decoding state sequence... DONE
```





Although it is not directly an output of `fitHMM`, it is possible to estimate the most likely state sequence for a fitted model. This is done with the function `viterbi` (named after the Viterbi algorithm), which returns

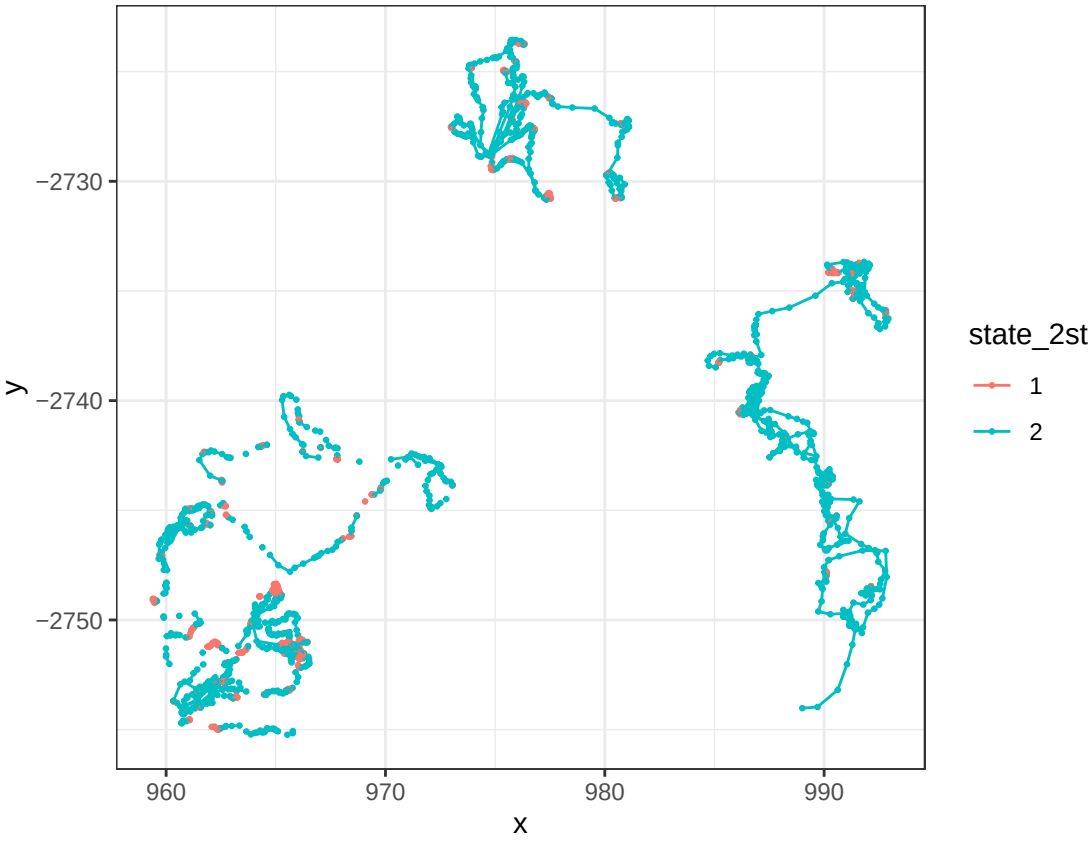
a sequence of estimated states for all times of observation. Here, we use it to create a custom plot of the trajectories, coloured by the estimated state sequences, for the 2-state and 3-state models.

```
# Get most likely sequence of states (Viterbi algorithm)
head(viterbi(hmm2))
```

```
[1] 1 1 1 1 1 1
```

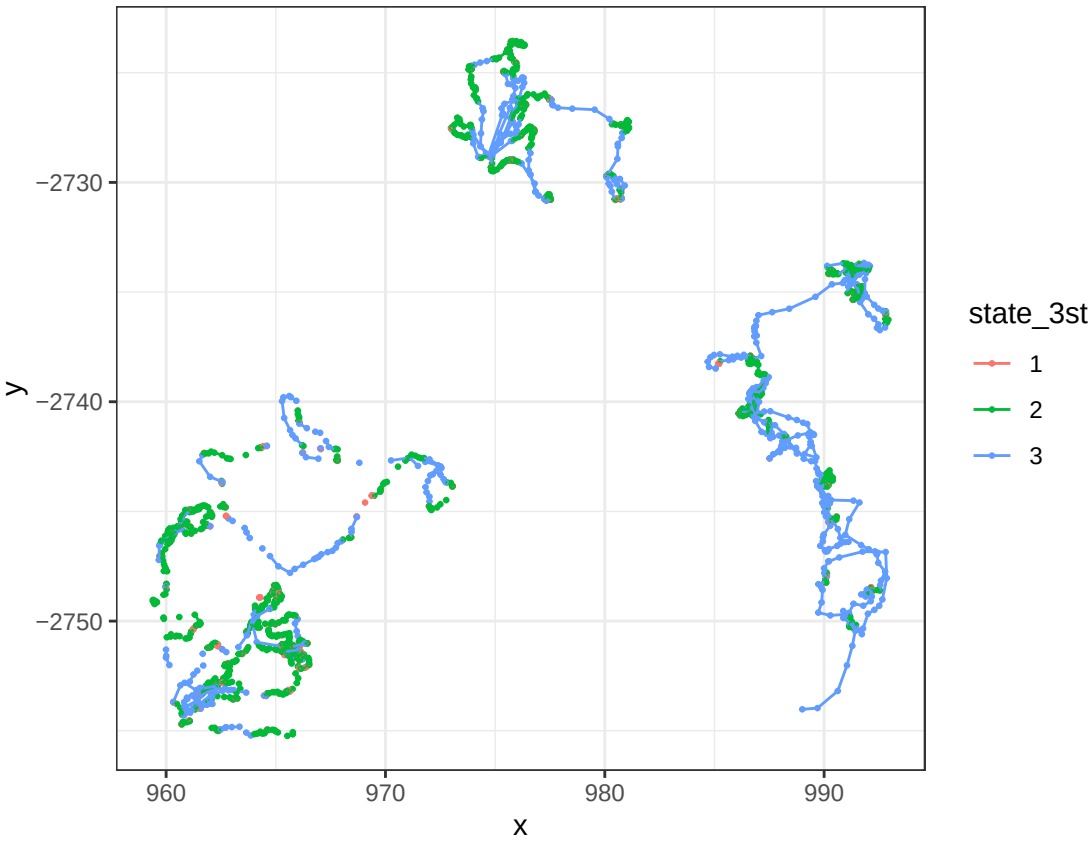
```
# Save most likely state sequences from 2-state and 3-state models
data_hmm1$state_2st <- factor(viterbi(hmm1))
data_hmm1$state_3st <- factor(viterbi(hmm2))

# Plot tracks, coloured by states
ggplot(data_hmm1, aes(x, y, col = state_2st, group = ID)) +
    geom_point(size = 0.5) + geom_path() +
    coord_equal()
```

```
Warning: Removed 316 rows containing missing values (geom_point).
```




```
ggplot(data_hmm1, aes(x, y, col = state_3st, group = ID)) +
    geom_point(size = 0.5) + geom_path() +
    coord_equal()
```

```
Warning: Removed 316 rows containing missing values (geom_point).
```



Model formulation: covariates

We might often be interested to investigate the drivers of animal behaviour. For example, in this data set, we have air temperature measurements taken by the tag at each time of observation. How does temperature tend to affect the elephants' behaviour? In HMMs, we can estimate the effect of covariates on the transition probabilities, i.e., the parameters that determine the dynamics of the unobserved state process.

We can specify the covariate model for the state transition probabilities using the `formula` argument in `fitHMM`. It uses the usual R formula syntax. We fit a model with a linear effect of temperature, and another model with a quadratic model. Model selection can be performed using the `AIC` function.

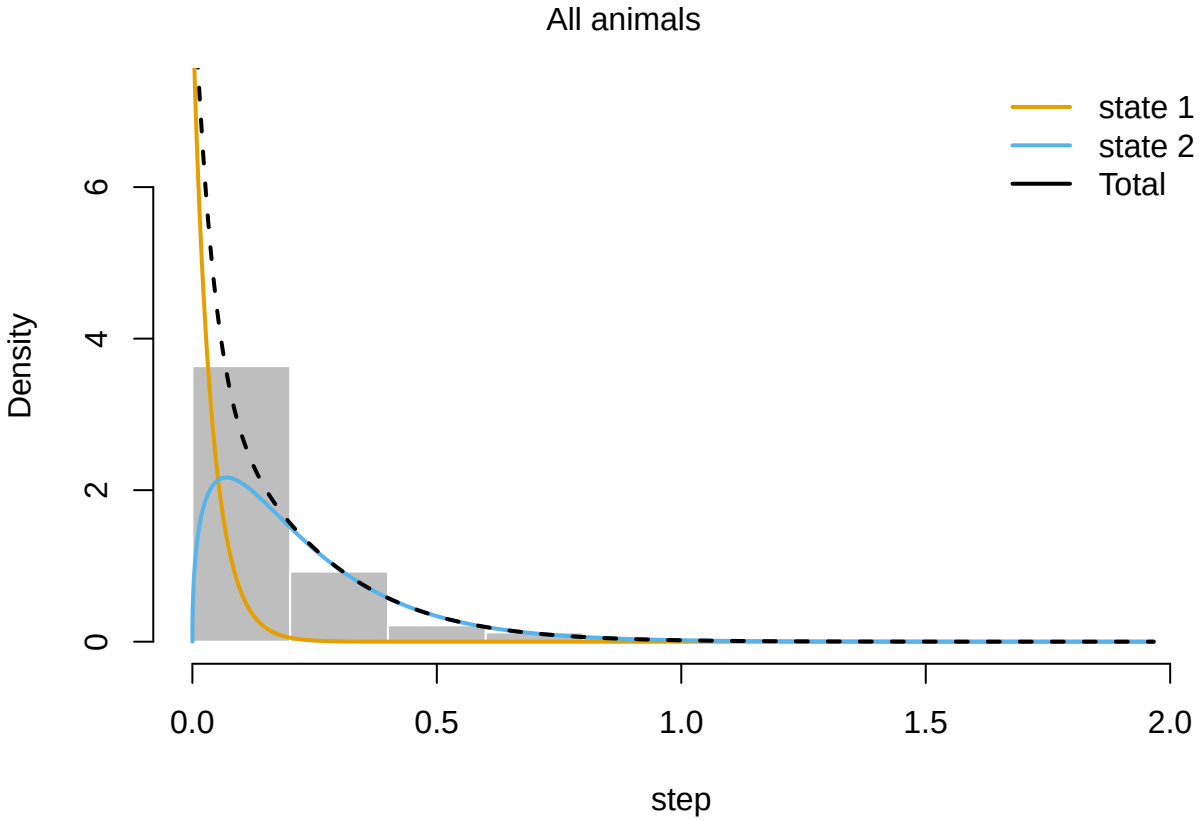
```
# Fit 2-state HMM with temperature covariate (linear or quadratic effect)
hmm3 <- fitHMM(data_hmm1, nbStates = 2, dist = dist,
               Par0 = Par0_2s, formula = ~temp)
hmm4 <- fitHMM(data_hmm1, nbStates = 2, dist = dist,
               Par0 = Par0_2s, formula = ~temp+I(temp^2))

# Compare models using AIC
AIC(hmm3, hmm4)
```

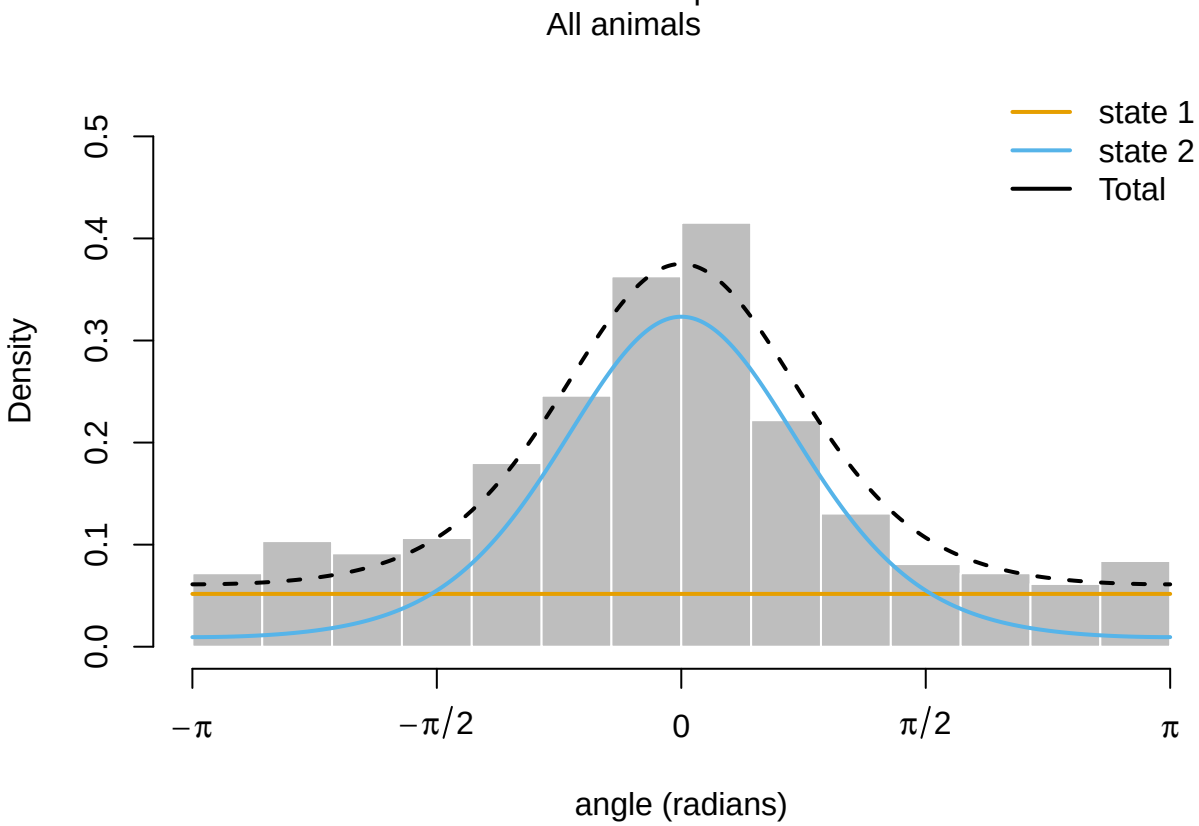
```
  Model      AIC
1  hmm3 1490.009
2  hmm4 1492.245
```

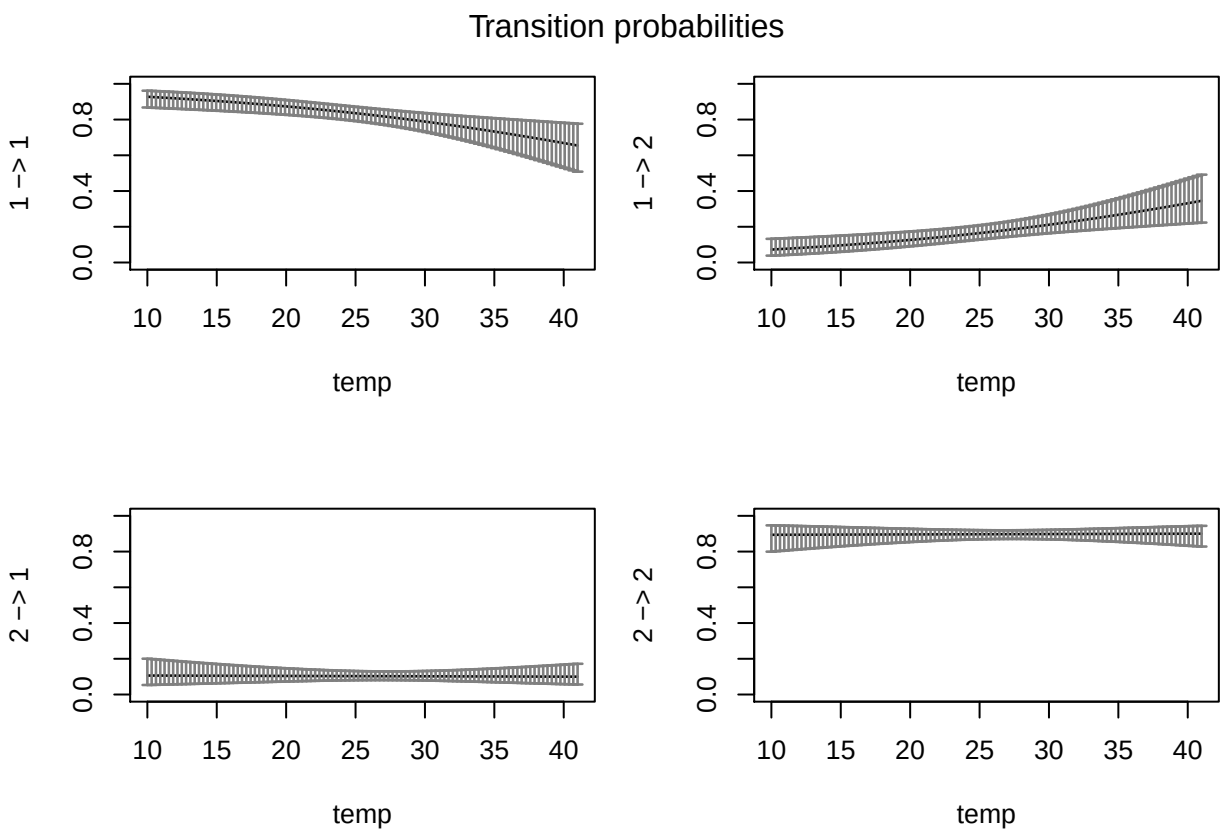
Here, the AIC very slightly favours the simpler model (with a linear effect), and we can visualise the transition probabilities as functions of temperature by plotting the model object.

```
# Plot estimated distributions and transition probabilities as functions of temperature
plot(hmm3, plotTracks = FALSE, ask = FALSE, plotCI = TRUE)
```

```
Decoding state sequence... DONE
```

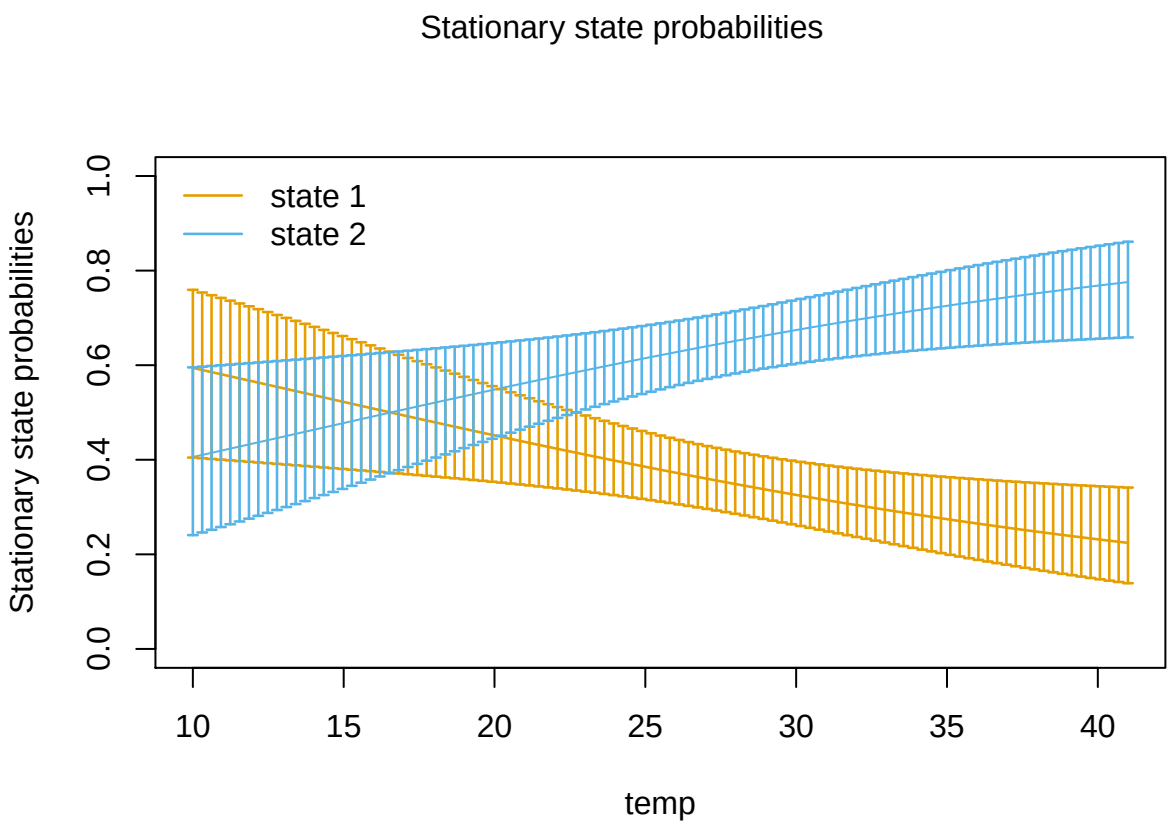






Another useful output is the plot of the stationary state probabilities as functions of the covariates. They measure the probability of being in each state in the long run, for a grid of values of the covariate, and can be displayed with the function `plotStationary`.

```
# Plot stationary state probabilities as functions of temperature
plotStationary(hmm3, plotCI = TRUE)
```



This plot suggests that the elephants were more likely to be in state 1 at low temperatures, and in state 2 at high temperatures. I don't know if there is a clear interpretation for this finding in terms of elephant behaviour.

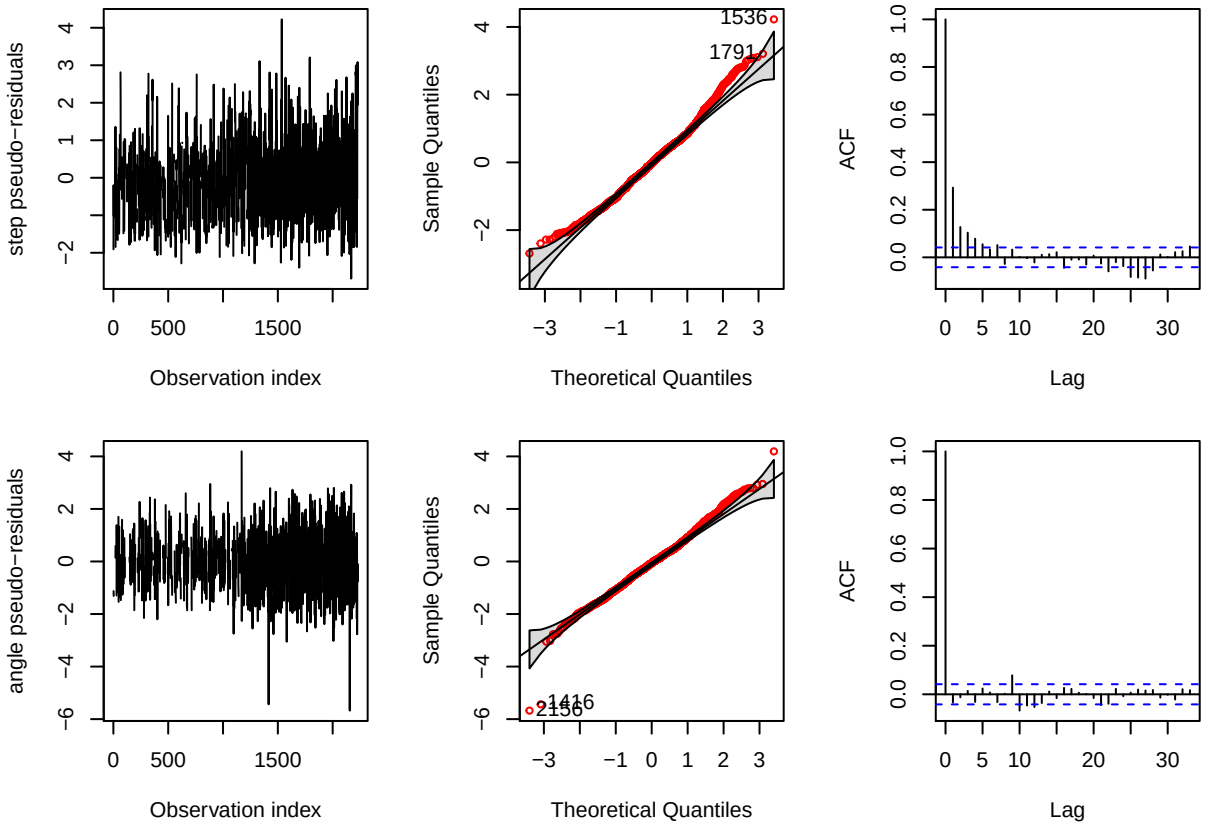
Model checking: pseudo-residuals

Model checking is a crucial step of the analysis, to assess whether the model adequately captures features of the data. One approach to model checking for HMMs is the use of pseudo-residuals. If the model perfectly describes the data, then pseudo-residuals will follow a standard normal distribution; any deviations point to lack of fit. (This is analogous to how residuals are interpreted in a linear regression framework.)

Pseudo-residuals can be visualised using `plotPR`, which creates three plots for each variable: a time series plot, a quantile-quantile (QQ) plot against the standard normal distribution, and a plot of the pseudo-residual autocorrelation function (ACF). Deviations from the identity line in the QQ plot indicate lack of fit, and the ACF plot is useful to check whether the model captured the autocorrelation present in the data.

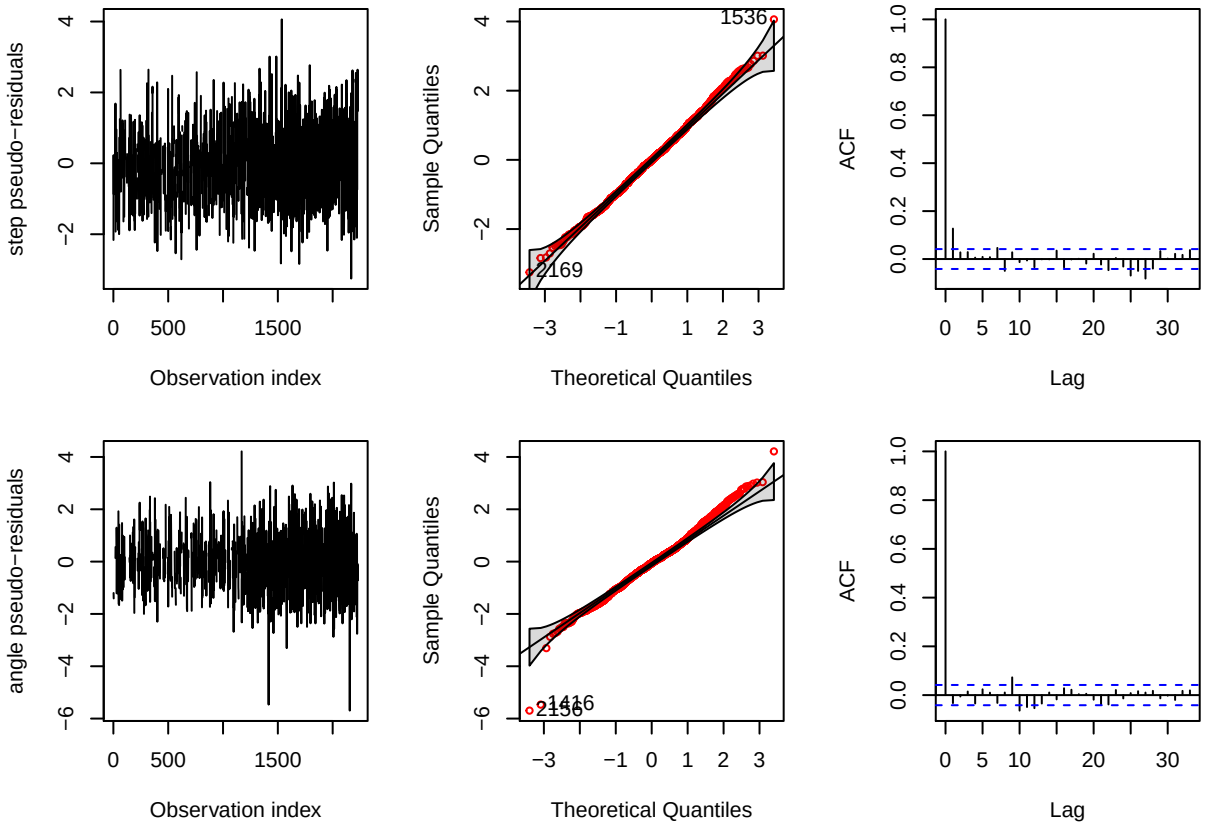
```
# Plot pseudo-residuals for 2-state and 3-state models
plotPR(hmm1)
```

```
Computing pseudo-residuals...
```




```
plotPR(hmm2)
```

```
Computing pseudo-residuals...
```



As expected, goodness-of-fit is better for the 3-state model, which is more flexible and can capture more detailed features of the data.

Regularisation

In cases where we cannot easily fill in missing data rows with NAs, the most common workaround is to regularise the data, i.e., predict locations on a regular time grid. This is for example the case when the locations are completely irregular in time.

Method 1: foieGras

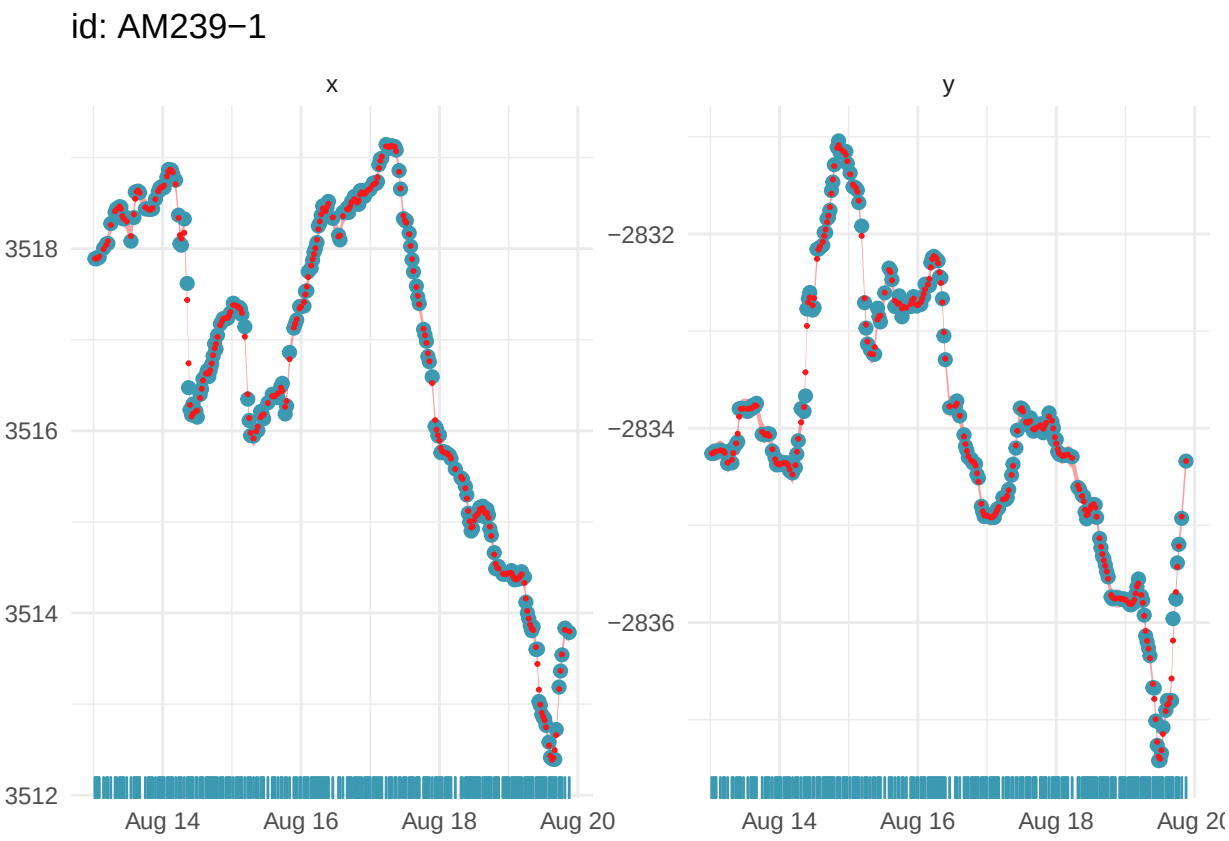
One convenient R package to do this is foieGras, which fits a continuous-time state-space model and uses it to predict missing locations (Jonsen et al. (2020)). foieGras can accommodate data of various qualities, and in particular it can take Argos location class as input, to account for the uncertainty in the observed locations. This process is also often called "filtering," in analogy with problems in signal processing where these methods were first developed.

```
# Change data to format expected by foieGras
data_foieGras <- data_split[,c("ID", "time", "lon", "lat")]
colnames(data_foieGras)[1:2] <- c("id", "date")
# Add column for location quality class (G for "GPS")
data_foieGras$lc <- "G"
# Change order of columns as expected by foieGras
data_foieGras <- data_foieGras[,c(1, 2, 5, 3, 4)]

# Fit state-space model to predict regular locations on 0.5h grid
ssm <- fit_ssm(d = data_foieGras, time.step = 0.5)

# Data frame of regularised locations
```

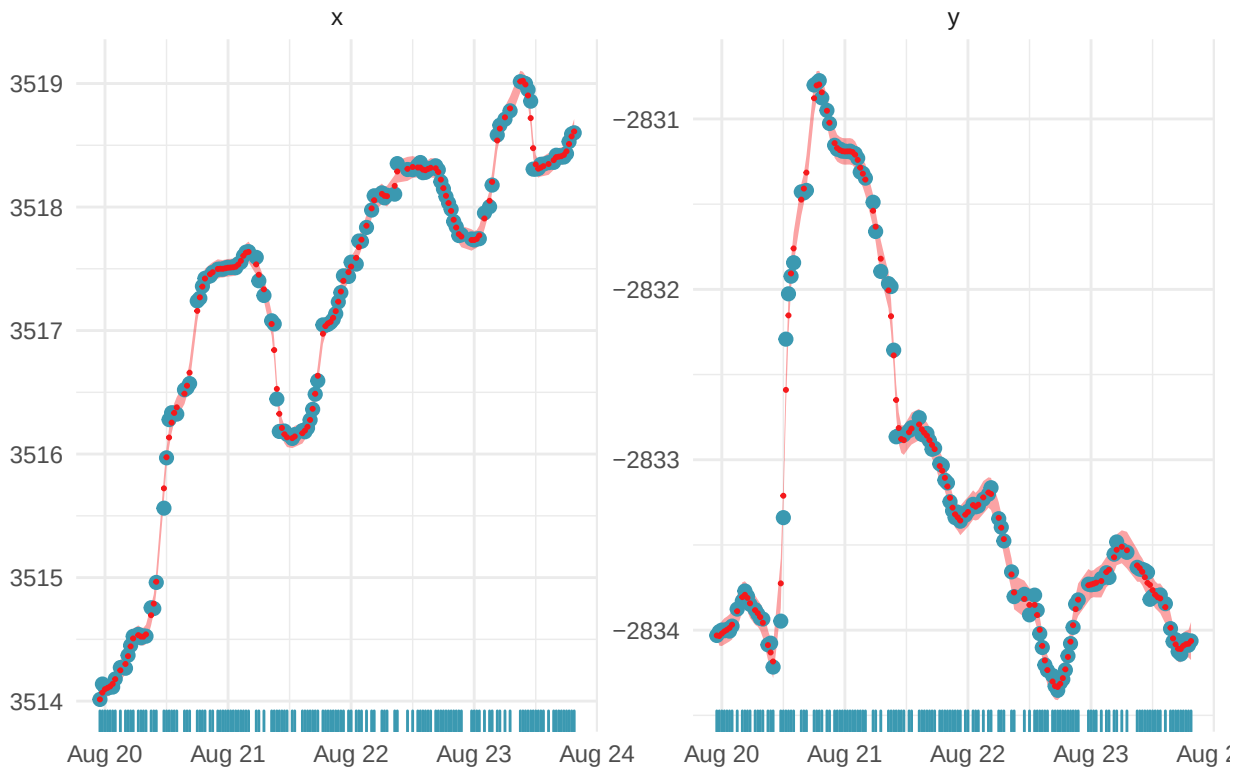
```
pred <- grab(ssm, what = "predicted", as_sf = FALSE)
data_reg <- as.data.frame(pred[, 1:4])
colnames(data_reg)[1:2] <- c("ID", "time")
```

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

```
$`AM239-1`
```




```
$`AM239-2`
```



$`AM239-3`

$`AM239-5`

$`AM239-7`

$`AM239-8`

$`AM253-1`

$`AM253-2`

$`AM253-3`

The output data frame has (predicted) locations at a regular 30-min time resolution, and these can be used as input for an HMM analysis. This time, we use the option `type = "LL"` in `prepData`, to indicate that we are using longitude-latitude locations as input.

```
# Get step lengths and turning angles from regularised data
data_hmm2 <- prepData(data = data_reg, type = "LL", coordNames = c("lon", "lat"))

head(data_hmm2, 10)
```

```
       ID        step       angle                time        x         y
1  AM239-1 0.001127797          NA 2007-08-13 00:00:00 31.60174 -24.80802
2  AM239-1 0.005188721 -0.42079964 2007-08-13 00:30:00 31.60174 -24.80801
3  AM239-1 0.010970135 -0.29954567 2007-08-13 01:00:00 31.60176 -24.80797
4  AM239-1 0.013584181 -0.54189443 2007-08-13 01:30:00 31.60182 -24.80789
5  AM239-1 0.022784029 -0.20122539 2007-08-13 02:00:00 31.60194 -24.80784
6  AM239-1 0.027077310 -0.07326580 2007-08-13 02:30:00 31.60216 -24.80780
7  AM239-1 0.026150446 -0.05145125 2007-08-13 03:00:00 31.60243 -24.80777
8  AM239-1 0.019975963 -0.06659096 2007-08-13 03:30:00 31.60269 -24.80775
```

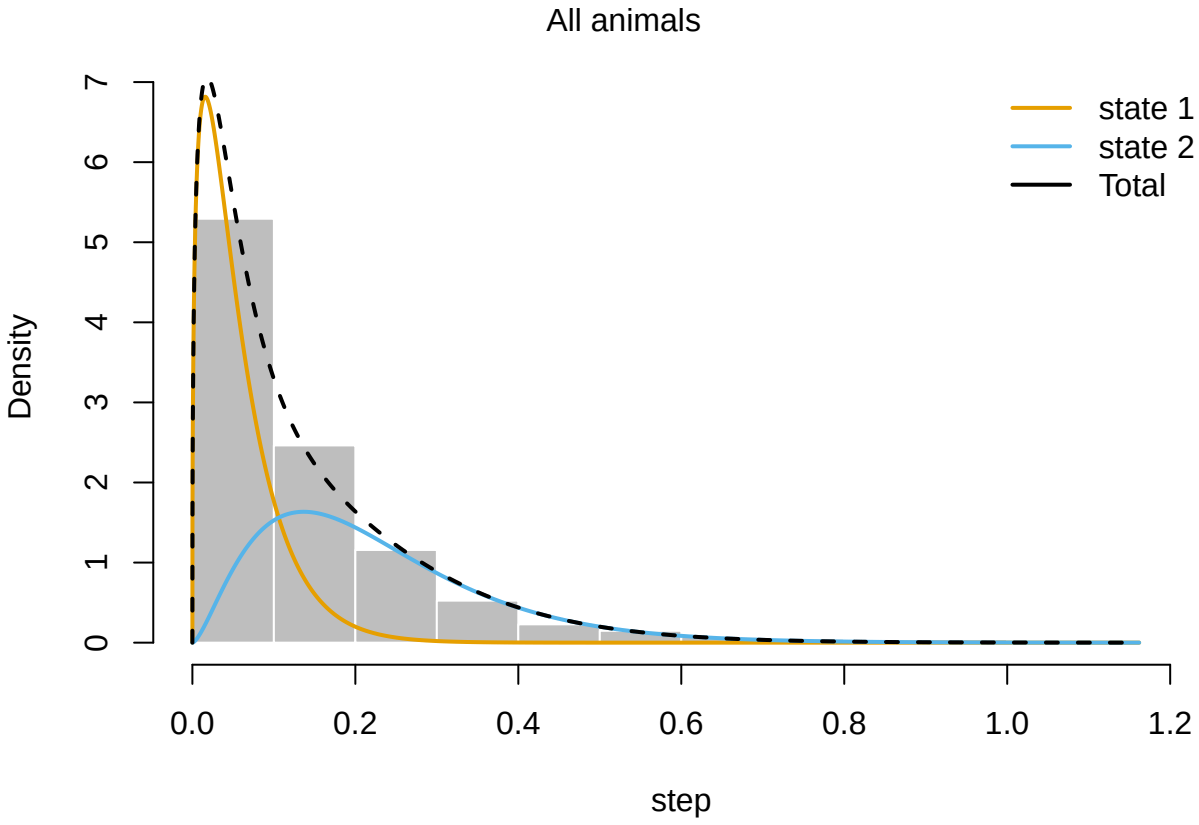
```
9  AM239-1 0.024042334 -0.27477285 2007-08-13 04:00:00 31.60289 -24.80775
10 AM239-1 0.039753777 -0.18910004 2007-08-13 04:30:00 31.60311 -24.80781
```

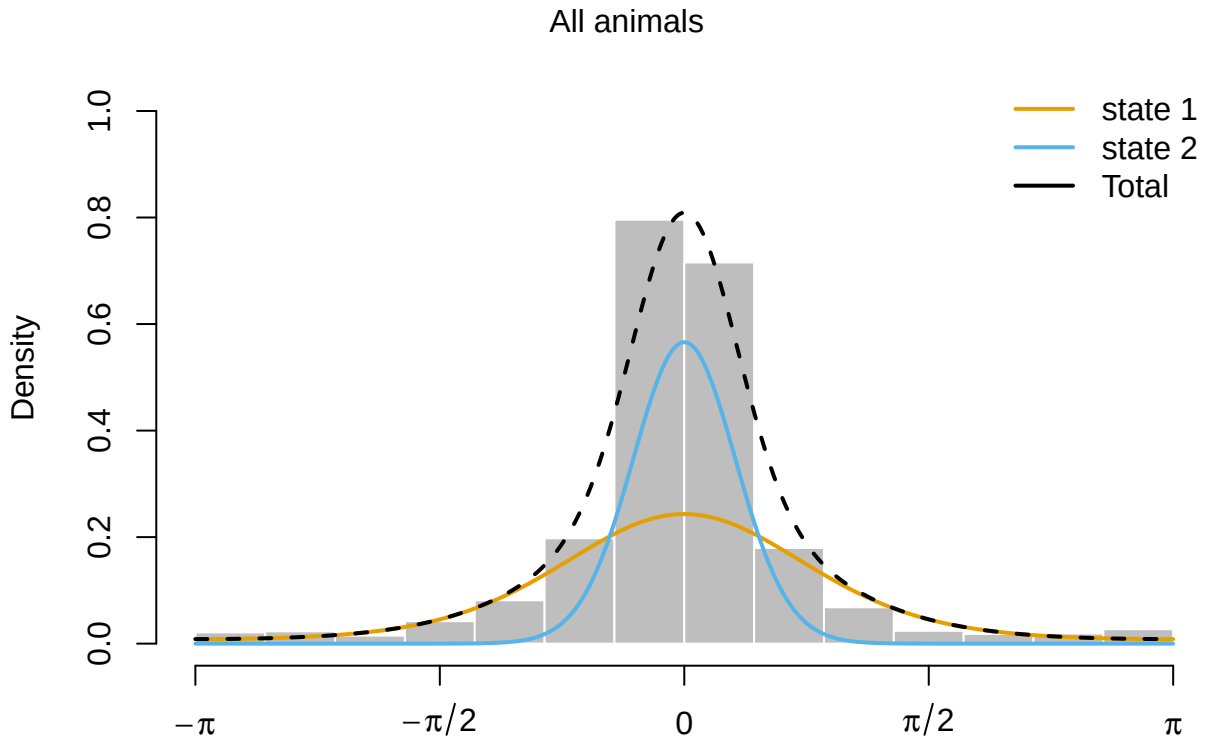
There are almost no missing values in this data set, as the gaps have been filled in. It is important to remember that these predicted locations are uncertain, and that their quality depends on the appropriateness of the foieGras model used for regularisation. This is one important reason to split the tracks at long gaps: regularising over long gaps would lead to very uncertain (and unrealistic) predicted locations, and the following findings would be unreliable. Here, we might consider that the predicted locations are good enough to be used for further analysis. (In practice, this takes careful consideration of data quality, inspection of plots of predicted tracks, etc.)

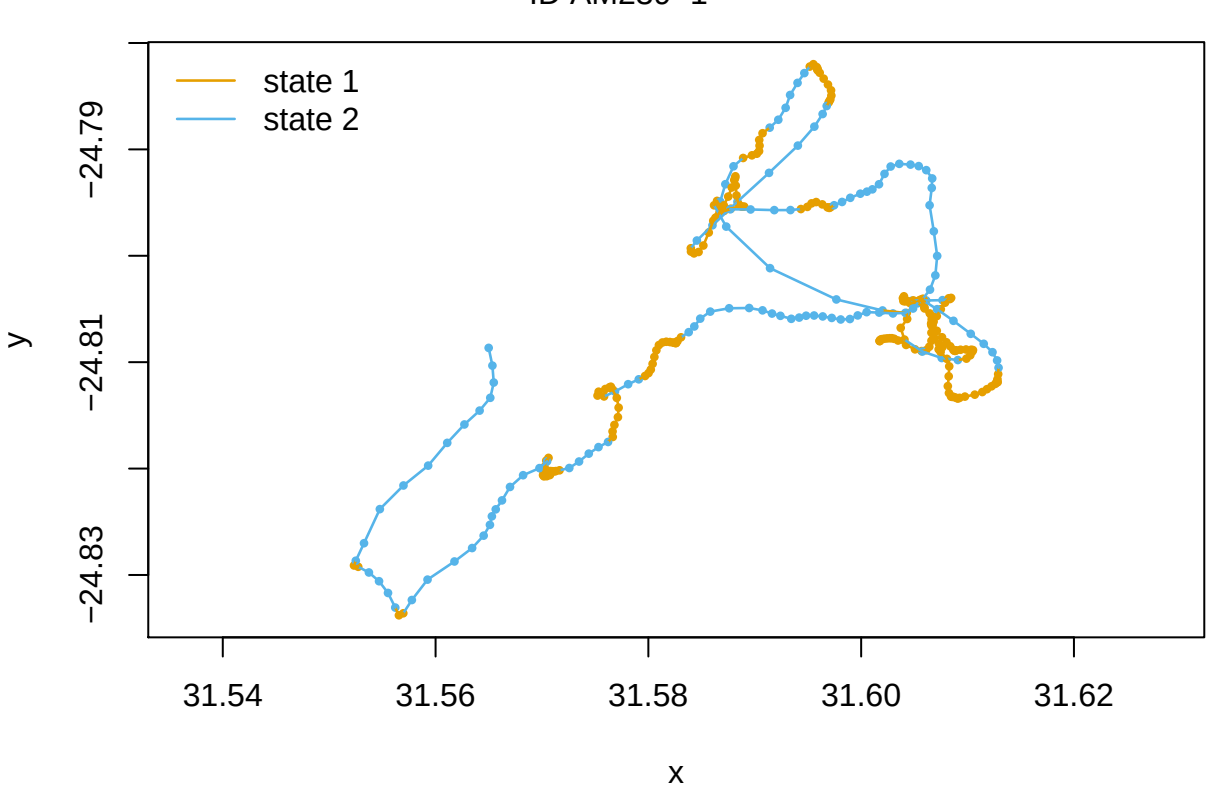
We fit a 2-state HMM to the regularised data.

```
# Fit 2-state HMM to regularised data
hmm4 <- fitHMM(data_hmm2, nbStates = 2, dist = dist, Par0 = Par0_2s)

plot(hmm4, ask = FALSE)
```

```
Decoding state sequence... DONE
```







The findings are very similar to those from the previous analysis, where missing data were specified as NA.

Method 2: crawl (through momentuHMM)

Another option for regularisation is to use the package crawl, which implements another continuous-time model that can predict missing locations (Johnson and London (2018)). There is some integration between crawl and momentuHMM, which makes it easier to use the two packages together. We use the momentuHMM function `crawlWrap` to fit the crawl model and predict locations on a 30-min time grid. The output can then directly be passed to `prepData` to get step lengths and turning angles.

```
# Predict locations on 30-min grid using crawl (through momentuHMM wrapper)
crw_out <- crawlWrap(obsData = data_split, timeStep = "30 min",
                     Time.name = "time", coord = c("x", "y"))
data_hmm3 <- prepData(data = crw_out)

# Fit 2-state HMM to regularised data
hmm5 <- fitHMM(data_hmm3, nbStates = 2, dist = dist, Par0 = Par0_2s)
```

Multiple imputation

As we mentioned, neither regularisation method we used accounted for the uncertainty in the predicted locations. For the HMM analyses, we treated as known data some locations that were actually outputs from another model. This can lead to underestimating the uncertainty in the HMM results, in particular in cases where the is high uncertainty in the predicted locations (e.g., large measurement error, prediction over long gaps).

One partial solution to this problem was recently proposed: multiple imputation (McClintock (2017)). The idea is that, instead of predicting just one trajectory, we could generate multiple plausible trajectories to fill in data gaps. Then, we fit an HMM to each imputed trajectory, and the variability in the fitted models helps us propagate the uncertainty from the missing locations through to the HMM results.

In momentuHMM, multiple imputation can be applied with the function `MIfitHMM`. It takes a fitted crawl model as input (i.e., the output of `crawlWrap`), and takes care of generating imputations and fitting HMMs. It can be quite computationally demanding because of the need to fit the model multiple times, but this can be parallelised because the imputations are independent. For illustration, we only use 10 imputations here, but it is likely that more would be needed in practice.

```
# Fit HMM using multiple imputation
hmm6 <- MIfitHMM(miData = crw_out, nSims = 10, ncores = 3, nbStates = 2,
                 dist = dist, Par0 = Par0_2s)
```

References

Johnson, Devin S., and Josh M. London. 2018. "Crawl: An r Package for Fitting Continuous-Cime Correlated Random Walk Models to Animal Movement Data." https://doi.org/10.5281/zenodo.596464.

Jonsen, Ian D., Toby A. Patterson, Daniel P. Costa, Philip D. Doherty, Brendan J. Godley, W. James Grecian, Christophe Guinet, et al. 2020. "A Continuous-Time State-Space Model for Rapid Quality-Control of Argos Locations from Animal-Borne Tags." *Movement Ecology* 8: 31. https://doi.org/10.1186/s40462-020-00217-7.

McClintock, Brett T. 2017. "Incorporating Telemetry Error into Hidden Markov Models of Animal Movement Using Multiple Imputation." *Journal of Agricultural, Biological and Environmental Statistics* 22 (3): 249–69.

Slotow, R, M Thaker, and AT Vanak. 2019. "Data from: Fine-Scale Tracking of Ambient Temperature and Movement Reveals Shuttling Behavior of Elephants to Water." Movebank data repository. https://doi.org/doi:10.5441/001/1.403h24q5.

Thaker, Maria, Pratik R Gupte, Herbert HT Prins, Rob Slotow, and Abi T Vanak. 2019. "Fine-Scale Tracking of Ambient Temperature and Movement Reveals Shuttling Behavior of Elephants to Water." *Frontiers in Ecology and Evolution* 7: 4.