hmmTMB user guide

Théo Michelot, Richard Glennie

2020-12-15

1 Introduction

The package hmmTMB implements hidden Markov models (HMMs) with flexible covariate dependence in all model parameters. In this document, we briefly describe the structure of the package, and illustrate its use with several examples. For general background on HMMs, see Zucchini, MacDonald, and Langrock (2016) and, for a presentation of HMMs in the context of ecological studies, see McClintock et al. (2020).

HMMs are time series models involving two processes: an unobserved state process (S_t) specified as a Markov chain, and an observation process (Z_t). At each time $t = 1, 2, \dots$, the distribution of the observation Z_t depends on the value S_t of the state process. There are therefore two sets of parameters:

- the state process is formulated in terms of a transition probability matrix (and an initial distribution);
- the observation process is formulated in terms of the state-dependent parameters of the observation distributions.

In hmmTMB, the parameters of the state process and of the observation process can depend on covariates, including linear fixed effects, smooth covariate effects using splines, and i.i.d. normal random effects.

2 Package structure

The package hmmTMB uses the R6 framework for object-oriented programming (Chang (2019)), and it is based on three main classes:

- MarkovChain: model for state process, including formulas for the transition probabilities
- Observation: model for observation process, including state-dependent distributions, and formulas for the state-dependent observation parameters
- HMM: contains a state process model (i.e., a MarkovChain object) and an observation model (i.e., an Observation object).

Model specification, model fitting and model visualisation can all be performed by manipulating objects from those three classes. Following the R6 syntax, an object is created with something like

```
hidden <- MarkovChain$new(n_states = 2, data = data.frame(z = rnorm(100)))
```

This line of code defines a hidden state model with 2 states and no covariate dependence. Once an object is created, various methods (i.e., functions) can be called to manipulate it, using the following syntax:

```
hidden$formulas()
```

```
$`S1>S2`
~1
<environment: 0x564ed140e0d8>

$`S2>S1`
```

```
~1
<environment: 0x564ed1415220>
```

3 Example 1: Elk movement analysis

We illustrate the use of the package on a data set of elk movement (from Morales et al. (2004)), accessible through the moveHMM package. In this example, we showcase the use of smoothing splines to model the relationship between model parameters and covariates.

```
# Load packages
library(hmmTMB)
library(moveHMM)
```

In analyses of animal movement, the step lengths and turning angles are modelled to capture features of the speed and directionality of the movement. We derive those variables from the location using the function prepData in moveHMM.

```
tracks <- prepData(elk_data, type = "UTM", coordNames = c("Easting", "Northing"))

head(tracks)
```

```
       ID       step      angle      x       y dist_water
1 elk-115  5518.4434         NA 769928 4992847     200.00
2 elk-115  1416.5663  0.1262112 766875 4997444     600.52
3 elk-115   239.7525  2.3832412 765949 4998516     561.81
4 elk-115   432.7600  0.9385238 765938 4998276     550.00
5 elk-115   103.7545  1.1375066 766275 4998005     302.08
6 elk-115 12416.4659 -0.9687435 766368 4998051     213.60
```

The data set now has columns for the coordinates (x and y), the step lengths and turning angles, the ID (track identifier), and a covariate (distance to water). hmmTMB doesn't allow for zero inflation for observation distributions with positive support (e.g. gamma), so we jitter step lengths with value zero.

```
indz <- which(tracks$step == 0)
tracks$step[indz] <- runif(length(indz), 0, 1)
```

The data frame passed to hmmTMB should include a column for ID if necessary, as well as for any response variables and covariates included in the model.

The class Observation encapsulates the observation process model of the hidden Markov model. To create an object of that class, we need to specify distributions for the response variables, as well as initial parameter values. We could also include covariate dependence in the parameters of the observation distributions. We use gamma distributions for the step lengths, and von Mises distributions for the turning angles.

```
# Number of states of the HMM
n_states <- 2

# Observation distributions: gamma for the step lengths, and
# von Mises for the turning angles
dists <- list(step = dist_gamma, angle = dist_vm)

# Initial parameters
mean0 <- c(300, 3000)
sd0 <- c(400, 4000)
par0_obs <- list(step = list(shape = mean0^2/sd0^2,
                             scale = sd0^2/mean0),
                 angle = list(mu = c(3, 0),
                              kappa = c(0.2, 1)))

# Create Observation object
obs <- Observation$new(data = tracks, dists = dists,
                       n_states = n_states, par = par0_obs)
```

Similarly, we need to create an object of the class MarkovChain for the hidden process model, encapsulating the formulas and parameter values. Here, we formulate the transition probabilities as functions of the distance to water covariate, with smoothing splines specified using the syntax from the package mgcv (Wood (2017)).

```
# Formula for transition probabilities
formula <- ~ s(dist_water, k = 5, bs = "ts")

# Create MarkovChain object
hid <- MarkovChain$new(n_states = n_states,
                       structure = formula,
                       data = tracks)
```

Combining the MarkovChain and Observation object, we create an object of the HMM class.

```
hmm <- HMM$new(obs = obs, hidden = hid)
```

We fit the model with the method fit, and print the output of the optimiser with res. It contains the parameters estimates (fixed effects), the value of the objective function at the optimum, and diagnostics from the optimiser.

```
hmm$fit()
```

```
#######################
## Observation model ##
#######################
+ step ~ gamma(shape, scale)
  * shape.state1 ~ 1
  * shape.state2 ~ 1
  * scale.state1 ~ 1
  * scale.state2 ~ 1

+ angle ~ vm(mu, kappa)
  * mu.state1 ~ 1
  * mu.state2 ~ 1
  * kappa.state1 ~ 1
  * kappa.state2 ~ 1

#########################
## State process model ##
#########################
                                 state 1                               state 2
state 1                                .  ~s(dist_water, k = 5, bs = "ts")
state 2 ~s(dist_water, k = 5, bs = "ts")                                 .
```

```
hmm$out()
```

```
$par
  coeff_fe_obs   coeff_fe_obs   coeff_fe_obs   coeff_fe_obs   coeff_fe_obs
   -0.17027821    -0.49452633     6.09136549     8.66823310     8.78984319
  coeff_fe_obs   coeff_fe_obs   coeff_fe_obs   coeff_fe_hid   coeff_fe_hid
   -0.03925058    -0.55074674    -1.31391268    -2.09554306    -0.38613779
```

```
log_lambda_hid log_lambda_hid      log_delta
    3.65138853    -0.17947182    -0.74629942

$value
[1] 6928.415

$counts
function gradient
      79       34

$convergence
[1] 0

$message
NULL
```

We can get confidence intervals for the estimated (working) parameters with the method CI_coeff By default, it returns 95% confidence intervals, but the argument level can be specified for different confidence levels.

```r
hmm$CI_coeff()
```

```
                      estimate        lower       upper
coeff_fe_obs1    -0.1702782112  -0.30077876 -0.03977766
coeff_fe_obs2    -0.4945263298  -0.83584091 -0.15321175
coeff_fe_obs3     6.0913654900   5.85660955  6.32612143
coeff_fe_obs4     8.6682330993   8.38151685  8.95494935
coeff_fe_obs5     8.7898431931 -11.76188425 29.34157064
coeff_fe_obs6    -0.0392505791  -0.58059647  0.50209532
coeff_fe_obs7    -0.5507467386  -0.80911292 -0.29238056
coeff_fe_obs8    -1.3139126847  -2.36595742 -0.26186795
coeff_fe_hid1    -2.0955430562  -2.65200529 -1.53908082
coeff_fe_hid2    -0.3861377941  -1.52457559  0.75230000
log_lambda_hid1   3.6513885264  -2.29828752  9.60106457
log_lambda_hid2  -0.1794718209  -3.30039333  2.94144969
log_delta.state1 -0.7462994215  -3.64426115  2.15166231
coeff_re_hid1    -0.0031521995  -0.06293923  0.05663483
coeff_re_hid2    -0.0005213864  -0.07988590  0.07884313
coeff_re_hid3    -0.0023450242  -0.17816167  0.17347163
coeff_re_hid4    -0.1794679282  -0.79346620  0.43453034
coeff_re_hid5    -0.0026157014  -0.39081750  0.38558609
coeff_re_hid6     0.0143861171  -0.52371253  0.55248477
coeff_re_hid7     0.0525058300  -1.13661508  1.24162674
coeff_re_hid8     1.5981093469   0.19277536  3.00344333
```

The methods predict_obspar and predict_tpm can also be used to obtain estimates and confidence intervals of the model parameters directly. For example, say that we want to predict the model parameters for `dist_water = 1000`,

```r
new_data <- data.frame(dist_water = 1000)
hmm$predict_obspar(new_data = new_data, CI = TRUE)
```

```
$estimate
, , 1

                state 1      state 2
step.shape    0.8434301    0.60985971
step.scale  442.0245794 5815.21535849
angle.mu      3.1406360   -0.06164675
angle.kappa   0.5765191    0.26876640


$low
, , 1

                state 1       state 2
step.shape    0.7444330    0.43530991
step.scale  349.8322431 4344.24882117
angle.mu     -3.1414506   -0.91787005
angle.kappa   0.4386498    0.09322082


$upp
, , 1

                state 1      state 2
step.shape    0.9626141    0.8396918
step.scale  561.8456074 7841.1539288
angle.mu      3.1415927    0.7696231
angle.kappa   0.7525409    0.7450250
```

```
hmm$predict_tpm(new_data = new_data, CI = TRUE)
```

```
$estimate
, , 1

          [,1]      [,2]
[1,] 0.8953086 0.1046914
[2,] 0.5199308 0.4800692


$low
, , 1

          [,1]       [,2]
[1,] 0.8380405 0.06751638
[2,] 0.2176724 0.19908546


$upp
, , 1

          [,1]      [,2]
[1,] 0.9324836 0.1619595
[2,] 0.8009145 0.7823276
```

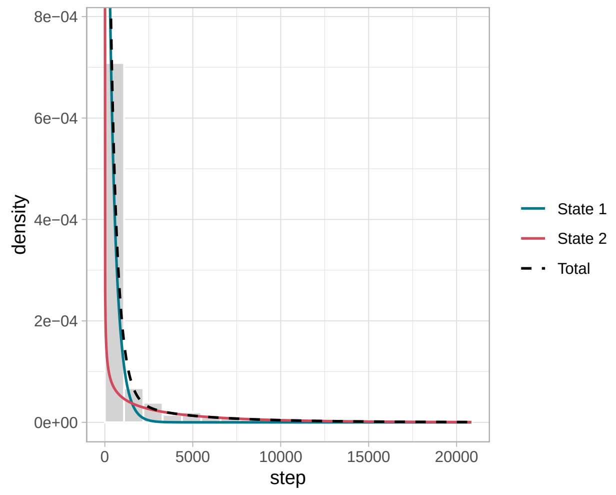
The method viterbi implements the Viterbi algorithm, to estimate the most likely state sequence.

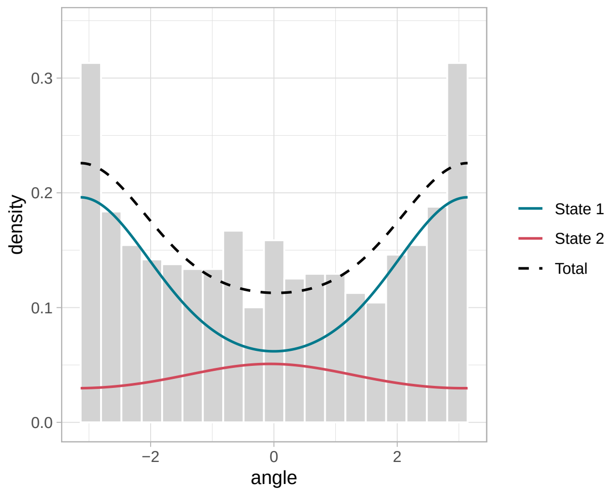
```
states <- hmm$viterbi()

head(states)
```

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

Various plotting functions are provided to visualise a fitted model. Using the method Observation$plot_dist, we can plot histograms of the observed data, overlaid with the estimated state-dependent probability density functions.

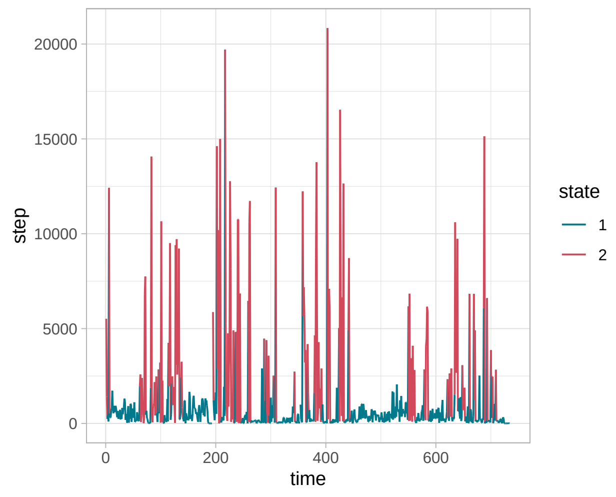
```
# Proportion of states in Viterbi sequence, to weigh pdfs
w <- table(states)/length(states)
obs$plot_dist("step", weights = w)
obs$plot_dist("angle", weights = w)
```

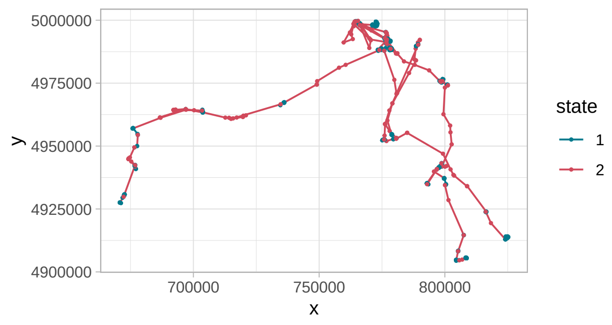




We can plot the data coloured by the Viterbi-decoded states. All plotting functions return ggplot objects, and can easily be edited using ggplot commands, e.g. to change axis labels or colours.

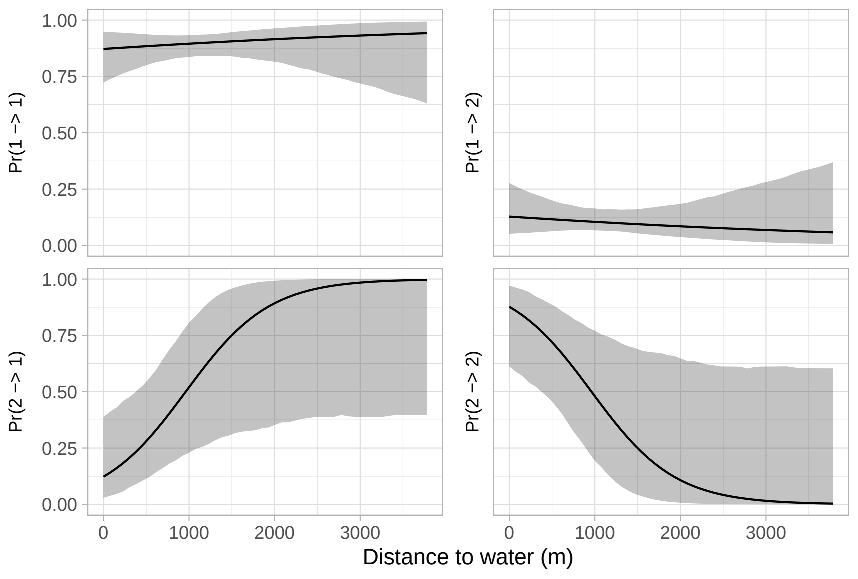
```
hmm$plot_ts("step")
hmm$plot_ts("x", "y") + coord_equal() + geom_point(size = 0.5)
```





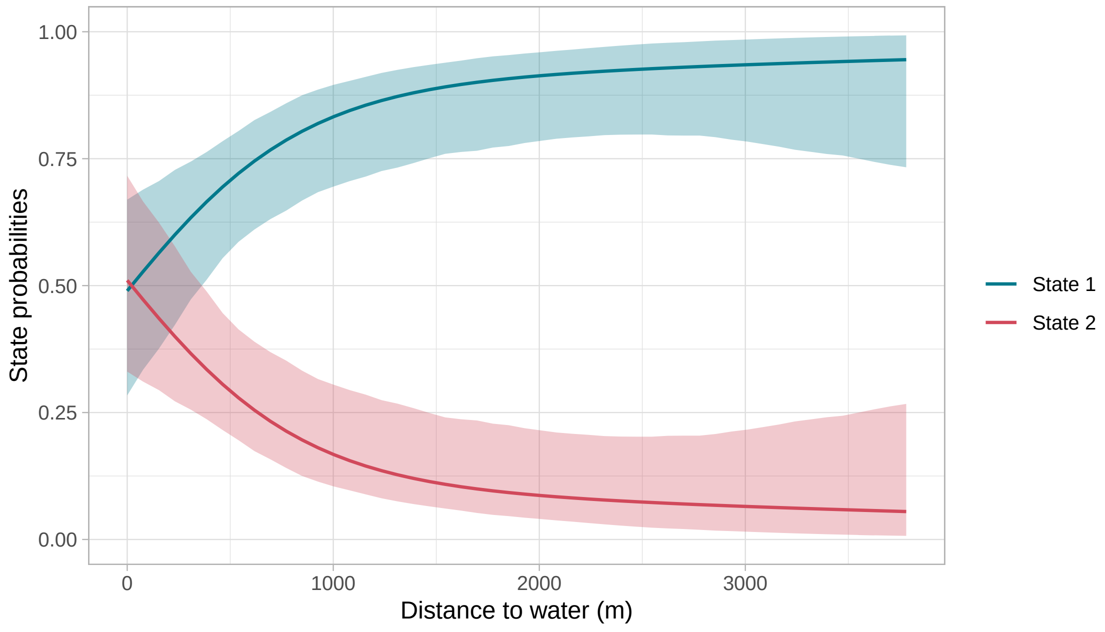
The function plot_tpm creates graphs of the transition probabilities as functions of a covariate, here distance to water.

```
hmm$plot_tpm("dist_water") + xlab("Distance to water (m)")
```



With the function plot_stat_dist, we can visualise the stationary state probabilities as functions of the distance to water.

```
hmm$plot_stat_dist("dist_water") +
  xlab("Distance to water (m)")
```



We can simulate from the fitted model with the method simulate, for example to check how well the model captures features of the real data. Here, we simulate step lengths and turning angles from the model.

```
# Number of simulated realisations
n_sim <- 500
# Data frame of covariate values for simulation
new_covs <- data.frame(dist_water = sample(tracks$dist_water,
                                           replace = TRUE,
                                           size = n_sim))
```

```
# Simulate from fitted model
sim_data <- hmm$simulate(n = n_sim, data = new_covs)
```

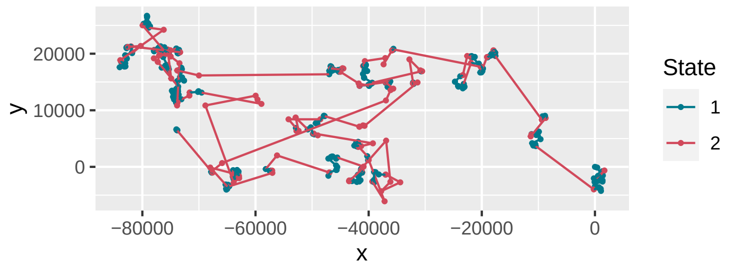
The output is a data frame with the same columns as the input, plus columns for the simulated variables, and a column for the simulated states.

```
head(sim_data)
```

```
  dist_water ID      step      angle
1    2073.19  1  249.9047 -0.2872475
2    1751.61  1  178.5520 -0.0517357
3     254.95  1 1273.1540 -0.7004399
4     305.16  1  264.3837  0.9564024
5     500.00  1  476.7363  1.6600528
6     145.77  1  768.0081 -3.0343708
```

We then derive the corresponding locations to visualise the simulated movement trajectory, and extract the states used in the simulation (stored as an attribute of the output of the `simulate` method).

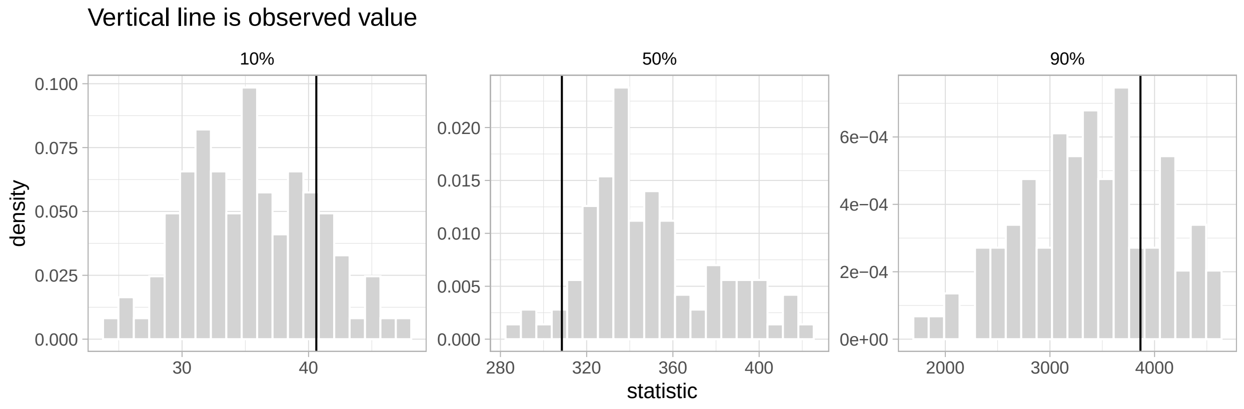
```
# Get locations from simulated step lengths and turning angles
bearings <- cumsum(sim_data$angle)
sim_dxy <- sim_data$step * cbind(cos(bearings), sin(bearings))
sim_data$x <- cumsum(c(0, sim_dxy[-n_sim, 1]))
sim_data$y <- cumsum(c(0, sim_dxy[-n_sim, 2]))

# Add state to simulated data frame
sim_data$state <- attr(sim_data, "state")

# Plot simulated trajectory
ggplot(sim_data, aes(x, y, col = factor(state), group = NA)) +
  scale_color_manual("State", values = hmmTMB_cols) +
  geom_point(size = 0.7) +
  geom_path() +
  coord_equal()
```



The HMM class has a dedicated method for simulation-based model checking, called `gof` (for “goodness-of-fit”).

It takes a function as an argument, which returns some statistic of interest from the data set (e.g., some quantile of the observed variable). The method evaluates that function for the observed data set and for a large number of data sets simulated from the fitted model. If the observed value of the statistic is an unusual value for the distribution of simulated statistics, it suggests that that feature of the data was not captured well by the model. Here, we apply the `gof` method for quantiles of the step length distribution:

```
# Function that takes 'data' as input and return the statistic(s) of interest
gof_fn <- function(data) {
    quantile(data$step, c(0.1, 0.5, 0.9), na.rm = TRUE)
}

# Run test
gof_out <- hmm$gof(gof_fn)
```



4 Example 2: wild haggis movement

We further illustrate the functionalities of hmmTMB with the analysis of the wild haggis tracking data set from Michelot, Langrock, and Patterson (2016). The objective of the study is to investigate the effect of terrain slope on the behaviour of wild haggises, accounting for inter-individual differences.

We load the data from Michelot, Langrock, and Patterson (2016), which contains locations for 15 haggis tracks, as well as slope and temperature measurements,

```
URL <- paste0("https://besjournals.onlinelibrary.wiley.com/action/downloadSupplement?",
              "doi=10.1111%2F2041-210X.13066&file=mee313066-sup-0001-rawhaggises.csv")
raw <- read.csv(url(URL))

head(raw)
```

```
  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
```

We use the prepData function from moveHMM to calculate step lengths, and we transform ID into a factor variable,

```
data <- prepData(raw, type = "UTM")
data$ID <- factor(data$ID)
```

```r
head(data)
```

```
  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
```

We define the hidden state process as a 2-state Markov chain, with a smooth effect of “slope” (using thin-plate regression splines), and a normal random intercept for “ID” to allow for differences between haggises. We don’t include the effect of temperature because Michelot, Langrock, and Patterson (2016) found no clear effect.

```r
n_states <- 2
hid <- MarkovChain$new(n_states = n_states,
                       structure = ~s(ID, bs = "re") + s(slope, k = 5, bs = "ts"),
                       data = data)
```

To define the observation model, we specify that the step lengths should be modelled with gamma distributions. We choose initial parameter values for the mean and standard deviation based on data visualisations, and derive initial values for the shape and scale based on that,

```r
# Observation distributions
dists <- list(step = "gamma")

# Initial parameter values
mean0 <- c(1, 5)
sd0 <- c(0.5, 3)
par0 <- list(step = list(shape = mean0^2/sd0^2,
                         scale = sd0/mean0^2))

# Define observation model
obs <- Observation$new(data = data,
                       n_states = n_states,
                       dists = dists,
                       par = par0)
```

We create the HMM object from the MarkovChain and Observation components, fit it using the fit method, and compute the most likely state sequence using the viterbi method,

```r
hmm <- HMM$new(hidden = hid, obs = obs)
hmm$fit()
```

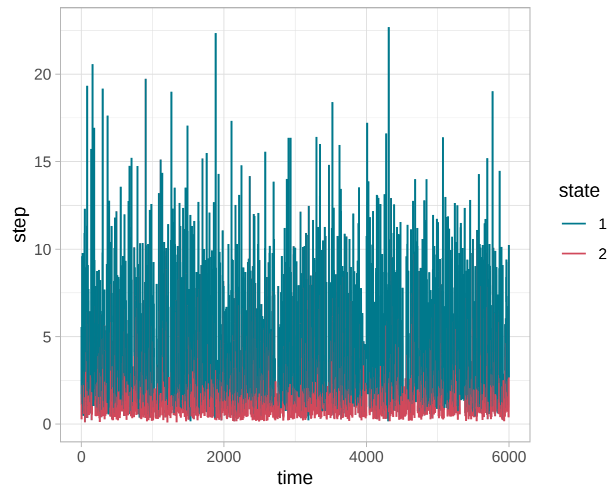
```
#######################
## Observation model ##
#######################
+ step ~ gamma(shape, scale)
  * shape.state1 ~ 1
  * shape.state2 ~ 1
  * scale.state1 ~ 1
  * scale.state2 ~ 1

#########################
## State process model ##
```

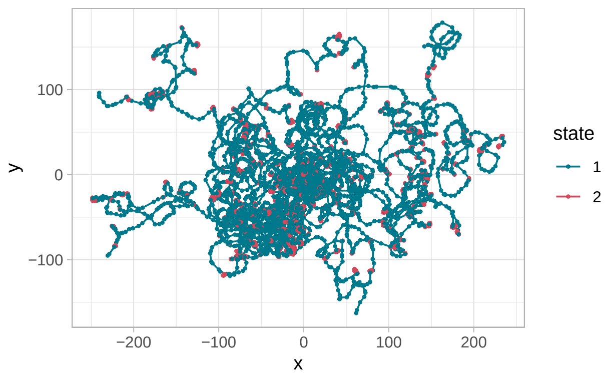
```
##########################
                                                    state 1
state 1                                                   .
state 2 ~s(ID, bs = "re") + s(slope, k = 5, bs = "ts")
                                                    state 2
state 1 ~s(ID, bs = "re") + s(slope, k = 5, bs = "ts")
state 2                                                   .
```

```
data$state <- hmm$viterbi()
```

We can plot the time series of step lengths, or the tracks, coloured by the estimated state sequence,

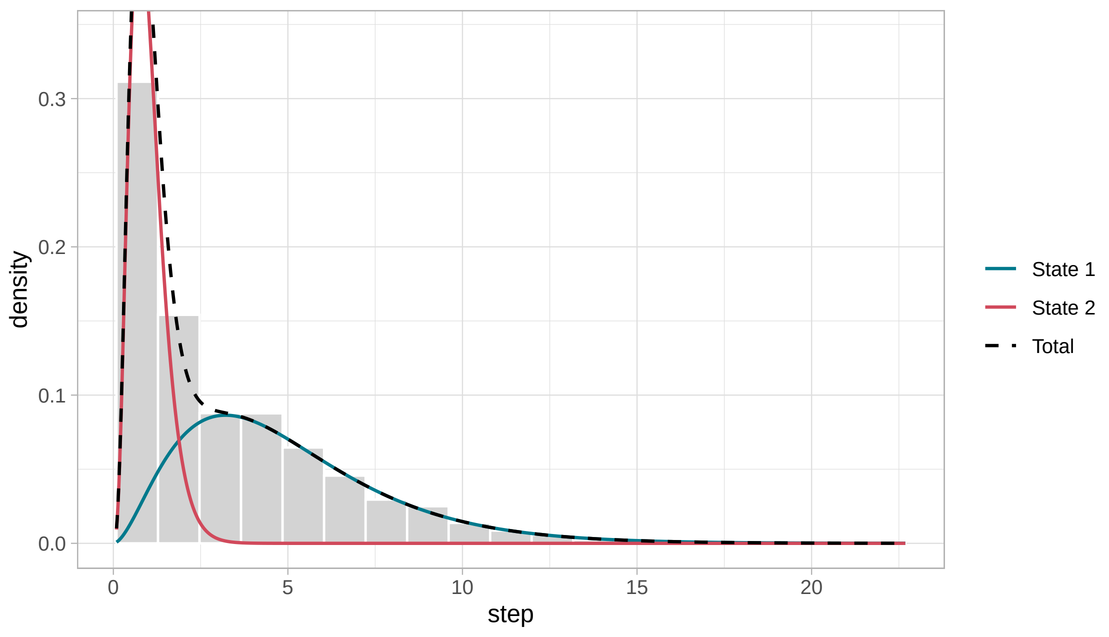
```
hmm$plot_ts("step")

hmm$plot_ts("x", "y") +
    geom_point(size = 0.5) +
    coord_equal()
```





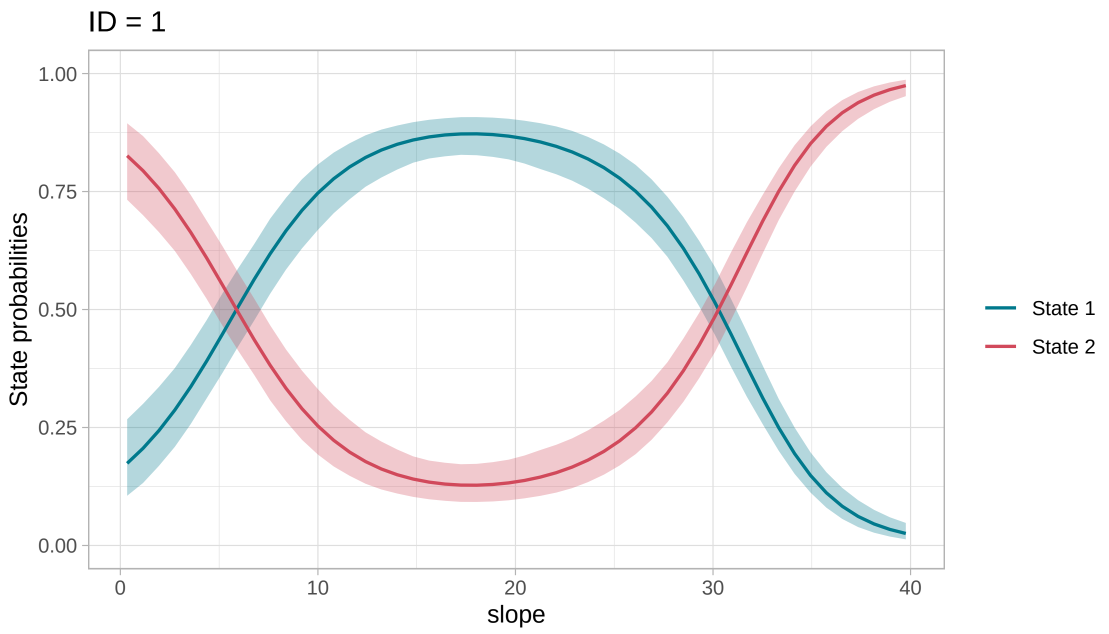
These plots suggest that state 1 captured fast movement behaviour (long step lengths) and state 2 captured slow movement (short step lengths). This can also be seen from the estimated state-dependent distributions of step lengths,

```
obs$plot_dist("step", w = table(data$state)/nrow(data))
```



To visualise the effect of slope on the behaviour of wild haggises, we plot the stationary state probabilities as functions of the slope,

```r
hmm$plot_stat_dist("slope")
```



When there are several covariates, the plotting functions fix the other covariates to their mean value (for continuous covariates) or to their first level (for factor covariates), as shown at the top of the plot. This can be changed by specifying the argument "covs", a data frame with a single row and one named column for each covariate.

We observe the same pattern found by Michelot, Langrock, and Patterson (2016): wild haggises tended to adopt the fast movement state at intermediate slopes (between 5 and 30 degrees), and the slow movement state on very flat or very steep terrains. Michelot, Langrock, and Patterson (2016) compared a linear and a quadratic effect of slope; the smoothing splines used here are more flexible, and the implementation automatically estimates the smoothness of the relationship during model fitting.

A measure of inter-individual heterogeneity is given by the standard deviation of the distribution of random intercepts, and can be obtained using the vcomp method,

```
hid$vcomp()
```

```
                        [,1]
S1>S2.s(ID)    0.1299689803
S1>S2.s(slope) 4.4774249764
S2>S1.s(ID)    0.0008857958
S2>S1.s(slope) 3.7399150762
```

References

Chang, W. 2019. *R6: Encapsulated Classes with Reference Semantics.* https://CRAN.R-project.org/package=R6.

McClintock, Brett T, Roland Langrock, Olivier Gimenez, Emmanuelle Cam, David L Borchers, Richard Glennie, and Toby A Patterson. 2020. "Uncovering Ecological State Dynamics with Hidden Markov Models." *arXiv Preprint, arXiv:2002.10497.*

Michelot, Théo, Roland Langrock, and Toby A Patterson. 2016. "moveHMM: An R Package for the Statistical Modelling of Animal Movement Data Using Hidden Markov Models." *Methods in Ecology and Evolution* 7 (11): 1308–15.

Morales, Juan Manuel, Daniel T Haydon, Jacqui Frair, Kent E Holsinger, and John M Fryxell. 2004. "Extracting More Out of Relocation Data: Building Movement Models as Mixtures of Random Walks." *Ecology* 85 (9): 2436–45.

Wood, S. N. 2017. *Generalized Additive Models: An Introduction with R.* 2nd ed. Chapman; Hall/CRC.

Zucchini, Walter, Iain L MacDonald, and Roland Langrock. 2016. *Hidden Markov Models for Time Series: An Introduction Using R, Second Edition.* CRC press.