Bayesian inference in hmmTMB

Théo Michelot

2025-05-21

This vignette describes functionalities of the package hmmTMB for Bayesian inference, which are based on Stan (Stan Development Team (2019); Stan Development Team (2022)). The package tmbstan conveniently integrates TMB with Stan, such that a TMB model object (such as the one created inside the `HMM` class in hmmTMB) can directly be used to run MCMC in Stan (Monnahan and Kristensen (2018)). For a more general introduction to hmmTMB, consult the vignette "Analysing time series data with hidden Markov models in hmmTMB".

1 Data

As an example, we will use a set of elephant seal movement tracks included in the R package aniMotum (Jonsen et al. (2020), Jonsen et al. (2023)), for which installation instructions can be found at https://ianjonsen.github.io/aniMotum/index.html. Our aim is not to provide a thorough ecological analysis, but only to illustrate features of the hmmTMB package.

The first chunk of code below predicts elephant seal positions at regular intervals using aniMotum, which is required for the hidden Markov model (HMM) analysis, and computes relevant movement variables with moveHMM.

```
library(aniMotum)
library(lubridate)
library(moveHMM)

# Predict positions at regular time intervals
fit <- fit_ssm(sese, model = "crw", time.step = 8)
```

```
fitting crw SSM to 4 tracks...
 pars:   1 1 0 0 -0.0361 0       pars:   0.70855 0.68102 -0.6588 -0.61586 -0.04018 -0.00
 pars:   1 1 0 0 1.80596 0       pars:   0.8076 0.78955 -0.70085 -0.64553 1.90246 0.0384
 pars:   1 1 0 0 0.49479 0       pars:   0.77538 0.75362 -0.68008 -0.65131 0.53864 0.014
 pars:   1 1 0 0 -0.7971 0       pars:   0.93223 0.92808 -0.707 -0.70005 -0.81462 -0.003
```

```
track_pred <- grab(fit, what = "predicted")

# Format for HMM analysis and get relevant variables
data <- data.frame(ID = track_pred$id,
                   x = track_pred$x,
                   y = track_pred$y,
                   time = track_pred$date,
                   day = yday(track_pred$date)/365)
data <- prepData(data, type = "UTM")
head(data)
```

```
            ID     step      angle        x         y                time
1 ct109-085-14 35.91562         NA 7842.458 -6401.444 2015-02-03 00:00:00
2 ct109-085-14 48.30228 0.06489973 7869.345 -6425.256 2015-02-03 08:00:00
3 ct109-085-14 48.50410 0.14105461 7907.507 -6454.868 2015-02-03 16:00:00
4 ct109-085-14 41.18862 0.18580925 7949.627 -6478.920 2015-02-04 00:00:00
5 ct109-085-14 33.36783 0.26099589 7988.552 -6492.386 2015-02-04 08:00:00
6 ct109-085-14 33.83670 0.04983900 8021.833 -6494.788 2015-02-04 16:00:00
         day
1 0.09315068
2 0.09315068
3 0.09315068
4 0.09589041
5 0.09589041
6 0.09589041
```

The three important columns in this data frame are the time series identifier `ID` (one value for each movement track), the step length `step` (which we will use as response variable), and the ordinal day `day` (which we will use as a covariate).

2 Model specification

The step length is the distance travelled by the animal between two observed locations and, as a positive variable, it can for example be modelled with a gamma distribution. To allow for time-varying dynamics, we will consider a 2-state HMM for the step length L_t such that

$$L_t \mid \{S_t = j\} \sim \text{gamma}(\mu_j, \sigma_j)$$

where $\mu_j > 0$ and $\sigma_j > 0$ are state-dependent mean and standard deviation parameters, respectively. That is, the step length is assumed to follow a different gamma distribution in each state. This is a common model in movement ecology to capture different behavioural phases based on an animal's speed (as measured by the step length). We can expect that one state will capture short steps (slow movement), and one state will capture long steps (fast movement).

We also want to understand how the elephant seals' behaviour changes through the year. We will include the effect of the ordinal day on the transition probabilities. In this simple 2-state model, there are two of them: $\gamma_{12} = \Pr(S_{t+1} = 2 \mid S_t = 1)$ and $\gamma_{21} = \Pr(S_{t+1} = 1 \mid S_t = 2)$. We model them as time-varying, with a quadratic effect of ordinal day d_t, i.e.,

$$\text{logit}(\gamma_{ij}^{(t)}) = \beta_{ij0} + \beta_{ij1} d_t + \beta_{ij2} d_t^2$$

where the logit function ensures that the probability is between 0 and 1, and the parameters $\beta_{ij0}, \beta_{ij1}, \beta_{ij2}$ are to be estimated.

2.1 Model structure

Model specification is identical whether maximum likelihood estimation or MCMC sampling is used for model fitting. As such, both methods can easily both be applied and contrasted for a given model object.

We create a `MarkovChain` object for the hidden state model, specifying the number of states and the formula for the transition probabilities (quadratic effect of ordinal day). We use the option `initial_state = 2` to fix the state process to 2 for all individuals. (We will later interpret state 2 to be the "fast movement" state.) We do this here because the initial distribution parameters are often not well identified, which can lead to convergence issues in the MCMC sampling. We could specify initial parameter values through the `tpm0` argument, which would be used as starting points for the MCMC algorithm, but we use the default values here (a transition probability matrix with 0.9 on the diagonal, and covariate effects equal to 0).

```
hid <- MarkovChain$new(data = data,
                       n_states = 2,
                       formula = ~day + I(day^2),
                       initial_state = 2)
```

For the `Observation` model, we use the `gamma2` distribution, which is parameterised in terms of mean and standard deviation rather than shape and rate. We provide initial parameters with the argument `par`, corresponding to plausible parameter values. We choose a larger mean in state 2, with the expectation that state 2 will capture longer step lengths (i.e., fast movement).

```
dists <- list(step = "gamma2")
par0 <- list(step = list(mean = c(10, 40), sd = c(10, 20)))
obs <- Observation$new(data = data,
                       dists = dists,
                       par = par0)
```

We finally combine the two model components into an `HMM` object.

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

2.2 Priors

By default, the priors of an `HMM` object are set to `NA`, which correspond to an improper flat prior on all model parameters. The function `set_priors` can be used to specify priors for the observation parameters and/or the transition probabilities.

In practice, hmmTMB transforms parameters to a "working" scale, i.e., into parameters defined over the whole real line (e.g., a positive parameter is log-transformed into a real working parameter). This is to avoid having to deal with constraints during the model fitting. The priors should be defined for those working parameters, rather than for the "natural" parameters that we are interested in.

We can see a list of the priors, and of the parameters on the working scale, using the functions `priors()` and `coeff_fe()`, respectively.

```
hmm$priors()
```

```
$coeff_fe_obs
                             mean sd
step.mean.state1.(Intercept)   NA NA
step.mean.state2.(Intercept)   NA NA
step.sd.state1.(Intercept)     NA NA
step.sd.state2.(Intercept)     NA NA


$coeff_fe_hid
```

```
                     mean sd
S1>S2.(Intercept)      NA NA
S1>S2.day              NA NA
S1>S2.I(day^2)         NA NA
S2>S1.(Intercept)      NA NA
S2>S1.day              NA NA
S2>S1.I(day^2)         NA NA

$log_lambda_obs
     mean sd

$log_lambda_hid
     mean sd
```

```
hmm$coeff_fe()
```

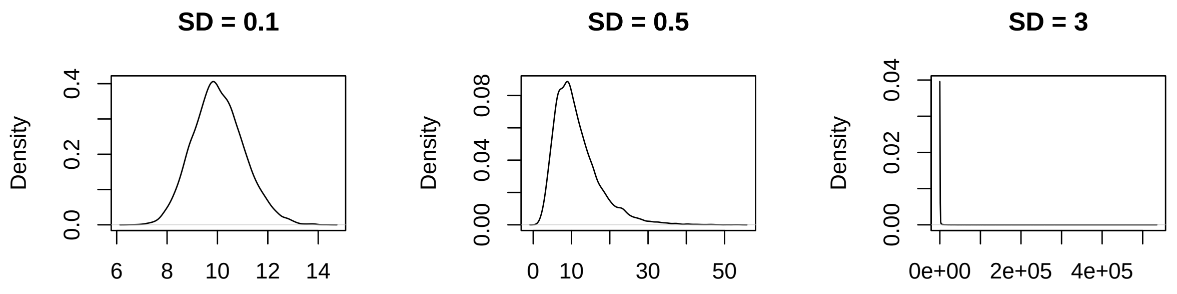
```
$obs
                                  [,1]
step.mean.state1.(Intercept) 2.302585
step.mean.state2.(Intercept) 3.688879
step.sd.state1.(Intercept)   2.302585
step.sd.state2.(Intercept)   2.995732

$hid
                        [,1]
S1>S2.(Intercept) -2.197225
S1>S2.day          0.000000
S1>S2.I(day^2)     0.000000
S2>S1.(Intercept) -2.197225
S2>S1.day          0.000000
S2>S1.I(day^2)     0.000000
```

2.2.1 Priors for observation parameters

The observation model has four working parameters: the log mean and the log standard deviation in each state (both log-transformed because the mean and standard deviation are positive). Denote as μ_j and σ_j the mean and standard deviation of the step length distribution for state $j \in \{1, 2\}$. In hmmTMB, only normal priors can be defined, and they should be specified in a matrix with one row for each working parameter, and two columns for the hyperparameters of the prior distribution (mean and standard deviation of normal prior).

Here is how we might choose prior hyperparameters for the log-mean in state 1, $\log(\mu_1)$. We suspect that state 1 will capture shorter step lengths, and, looking at the data, we can make an educated guess that the mean in state 1 will be somewhere between 0 and 30km. (This is also why we chose an initial parameter value of 10 for μ_1 above.) So, we might set the mean of the prior distribution of $\log(\mu_1)$ to $\log(10)$. Choosing the standard deviation of the normal prior is less straightforward, but we can sample from the prior distribution and transform to an interpretable quantity to check whether a given value seems reasonable (similar to prior predictive checks). Below, we compare the prior distributions on μ_1 (i.e., $\exp(\log(\mu_1))$) obtained for three different values of the prior standard deviation: 0.1, 0.5, and 3.

```r
plot(density(exp(rnorm(1e4, mean = log(10), sd = 0.1))),
     main = "SD = 0.1", xlab = NA)
plot(density(exp(rnorm(1e4, mean = log(10), sd = 0.5))),
     main = "SD = 0.5", xlab = NA)
plot(density(exp(rnorm(1e4, mean = log(10), sd = 3))),
     main = "SD = 3", xlab = NA)
```



The prior distribution for SD = 0.1 is quite narrow around 10, and would perhaps constrain the model too much. On the other extreme, the prior for SD = 3 is extremely wide and gives high probability to unrealistically large values for μ_1 (in the 1000s of km). The middle option, SD = 0.5, seems like a good compromise. We use a similar reasoning to choose the following priors:

$$\log(\mu_1) \sim N(\log(10), 0.5^2)$$
$$\log(\mu_2) \sim N(\log(40), 1^2)$$
$$\log(\sigma_1) \sim N(\log(10), 0.5^2)$$
$$\log(\sigma_2) \sim N(\log(20), 0.5^2)$$

```r
# Parameter of normal priors for observation parameters
prior_obs <- matrix(c(log(10), 0.5,
                      log(40), 1,
                      log(10), 0.5,
                      log(20), 0.5),
                    ncol = 2, byrow = TRUE)
```

2.2.2 Priors for the transition probabilities

In a 2-state model, the working parameters for the hidden state process are the coefficients $(\beta_{120}, \beta_{121}, \beta_{122}, \beta_{210}, \beta_{211}, \beta_{212})$ defined above. For $i \neq j \in \{1, 2\}$, β_{ij0} is an intercept parameter, β_{ij1} is the linear effect of ordinal day, and β_{ij2} is the quadratic effect of ordinal day.

We use the following priors,

$$
\begin{aligned}
\beta_{ij0} &\sim N(-2, 1) \\
\beta_{ij1} &\sim N(0, 20) \\
\beta_{ij2} &\sim N(0, 20)
\end{aligned}
$$

The mean for the intercept is chosen as -2 because $\text{logit}(0.1) \approx -2$, i.e., the prior suggests that the off-diagonal elements of the transition probability matrix should be small (as is often the case in practice due to autocorrelation in the hidden process). The coefficient for day and day squared have priors centered on zero with large variance; one could use prior predictive checks to ensure that these cover a reasonable range of possible relationships between ordinal day and transition probabilities. Note that the definition of the working parameters is a little more complicated in models with more than 2 states.

```r
# Parameter of normal priors for transition probabilities
prior_hid <- matrix(c(-2, 1,
                      0, 20,
                      0, 20,
                      -2, 1,
                      0, 20,
                      0, 20),
                    ncol = 2, byrow = TRUE)
```

2.2.3 Updating the priors

Finally, we update the priors stored in the model object using `set_priors()`, and we check that they have been correctly set.

```
# Update priors
hmm$set_priors(new_priors = list(coeff_fe_obs = prior_obs,
                                 coeff_fe_hid = prior_hid))

hmm$priors()
```

```
$coeff_fe_obs
                                 mean  sd
step.mean.state1.(Intercept) 2.302585 0.5
step.mean.state2.(Intercept) 3.688879 1.0
step.sd.state1.(Intercept)   2.302585 0.5
step.sd.state2.(Intercept)   2.995732 0.5

$coeff_fe_hid
                  mean sd
S1>S2.(Intercept)   -2  1
S1>S2.day            0 20
S1>S2.I(day^2)       0 20
S2>S1.(Intercept)   -2  1
S2>S1.day            0 20
S2>S1.I(day^2)       0 20

$log_lambda_obs
     mean sd

$log_lambda_hid
     mean sd
```

3 Fitting the model

The main function to fit a model using Stan in hmmTMB is `HMM$fit_stan()`. It takes the same arguments as `tmbstan()` from the tmbstan package, and documentation for that function should be consulted for more details. Here, we pass two arguments: the number of chains (`chains`) and the number of MCMC iterations in each chain (`iter`). In practice, these arguments should be chosen carefully to ensure convergence of the sampler to the stationary distribution (see Stan documentation for more information); the values below were merely chosen for speed. In this example, running the sampler for 2000 iterations takes about one minute on a laptop.

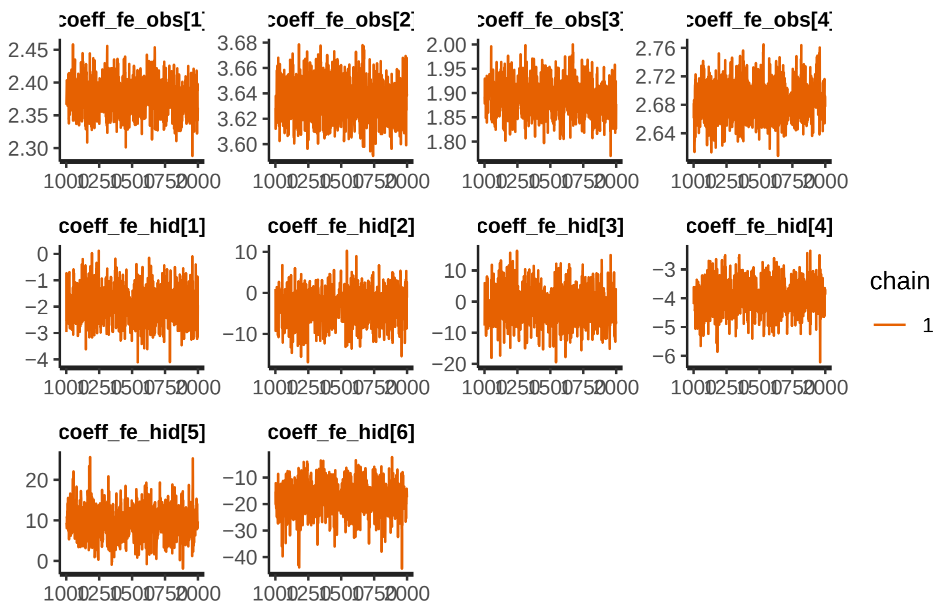
```
hmm$fit_stan(chains = 1, iter = 2000)
```

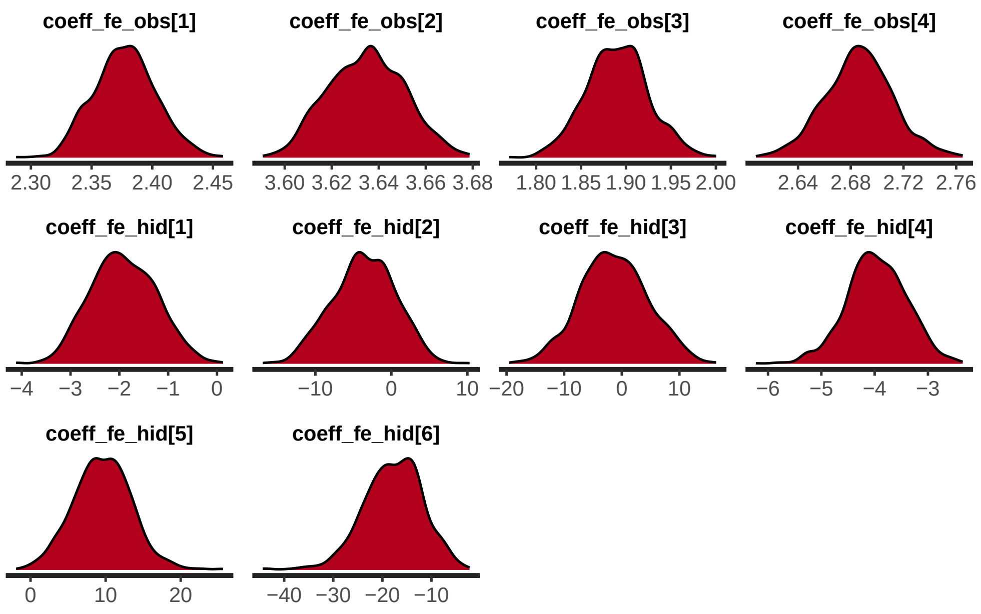
4 Inspecting the results

4.1 Working parameters

After running `fit_stan()`, the output of Stan is accessible with the `out_stan()` function. This is an object of class `stanfit`, and it can directly be used with functions from the rstan package, e.g., to create trace plots or density plots of the posterior samples. Note that these plots show the working parameters.

```
rstan::traceplot(hmm$out_stan())
rstan::stan_dens(hmm$out_stan())
```





4.2 Natural parameters

To inspect the posterior distributions of the natural parameters, which is often more interesting, we can extract posterior samples using `iters()`. This returns a matrix with one column for each parameter and one row for each MCMC iteration. It can directly be used to make traceplots, histograms, etc., or to produce credible intervals.

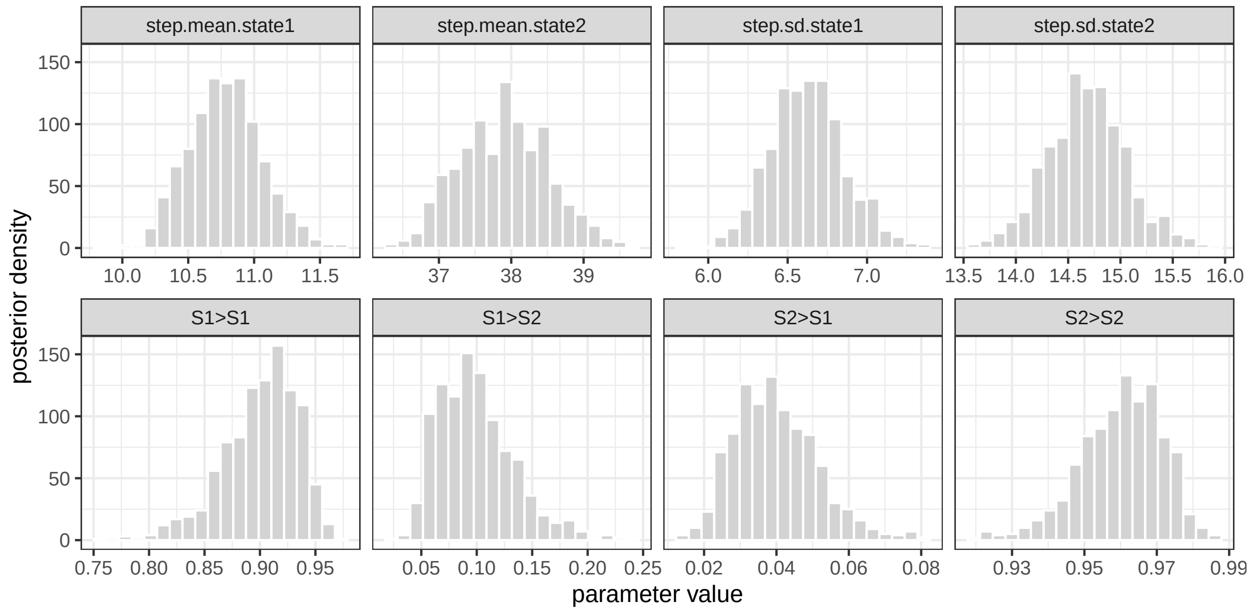
Note that, if a parameter depends on a covariate, the first row of the data set is used to predict the parameter value in `iters()`, so the output should be interpreted with care. In our analysis, the transition probabilities depend on the ordinal day, and they are computed for March 2nd in the plot below. We visualise the effect of ordinal day on the transition probabilities in a later section.

```r
iters <- hmm$iters()
head(iters)
```

```
     step.mean.state1 step.mean.state2 step.sd.state1 step.sd.state2     S1>S1
[1,]         10.65499         37.65349       6.634522       14.61353 0.9505509
[2,]         10.78191         37.03966       6.542455       14.32176 0.8235125
[3,]         10.91379         37.88762       6.888650       14.43322 0.9264828
[4,]         11.06080         38.02146       6.877682       14.68482 0.9259134
[5,]         10.88368         37.91256       6.659800       13.74549 0.8758979
[6,]         10.86458         37.20725       6.575867       13.68309 0.8722138
          S1>S2       S2>S1      S2>S2
```

```
[1,] 0.04944908 0.03095593 0.9690441
[2,] 0.17648751 0.05076417 0.9492358
[3,] 0.07351722 0.02949848 0.9705015
[4,] 0.07408661 0.03337253 0.9666275
[5,] 0.12410214 0.04701836 0.9529816
[6,] 0.12778620 0.03971871 0.9602813
```

```
iters_df <- as.data.frame.table(iters)
ggplot(iters_df, aes(x = Freq)) +
    geom_histogram(bins = 20, fill = "lightgrey", col = "white") +
    facet_wrap("Var2", nrow = 2, scales = "free_x") +
    labs(x = "parameter value", y = "posterior density")
```



For credible intervals, we can use quantiles of the posterior samples. As the transition probabilities depend on a covariate, we only compute credible intervals for the parameters of the step length distributions.

```
# 95% equal-tail credible intervals for observation parameters
apply(iters[,1:4], 2, quantile, probs = c(0.025, 0.975))
```

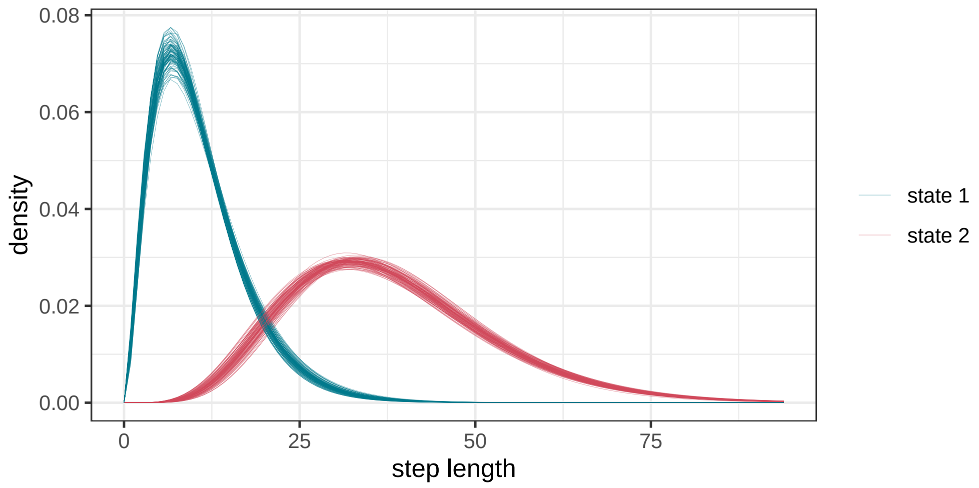
```
      step.mean.state1 step.mean.state2 step.sd.state1 step.sd.state2
2.5%          10.27398         36.80682       6.191359       13.94672
97.5%         11.34614         39.11563       7.114788       15.45881
```

4.3 Posterior samples of observation distributions

Based on the posterior samples that we have for all model parameters, we can get visualise the posterior distribution of various model components, such as the observation distributions. That is, we compute the density function of the step length distribution in each state, for each posterior sample, and plot them to visualise the uncertainty in the distributions. For clearer visualisation, we only select 100 random posterior samples here.

```r
# Select 100 random posterior samples from the
# 1000 post-warmup samples
ind_post <- sort(sample(1:1000, size = 100))

# Grid of step lengths
step_grid <- seq(0, max(data$step, na.rm = TRUE), length = 100)
# Loop over 100 random posterior samples
obsdist_list <- lapply(ind_post, function(i_post) {
    # Get gamma mean and SD
    par_post1 <- matrix(hmm$iters()[i_post,1:4], nrow = 2)
    # Get gamma scale and rate for R dgamma function
    par_post2 <- cbind(par_post1[,1] ^ 2 / par_post1[,2] ^ 2,
                       par_post1[,1] / par_post1[,2] ^ 2)

    # Compute gamma density in each state
    dens_state1 <- dgamma(step_grid,
                          shape = par_post2[1,1],
                          rate = par_post2[1,2])
    dens_state2 <- dgamma(step_grid,
                          shape = par_post2[2,1],
                          rate = par_post2[2,2])

    dens <- data.frame(step = step_grid,
                       value = c(dens_state1, dens_state2),
                       state = rep(paste0("state ", 1:2), each = 100))
    dens$group <- paste0("iter ", i_post, " - ", dens$state)
    return(dens)
})
# Combine data frames for plot
obsdist_df <- do.call(what = rbind, args = obsdist_list)
```

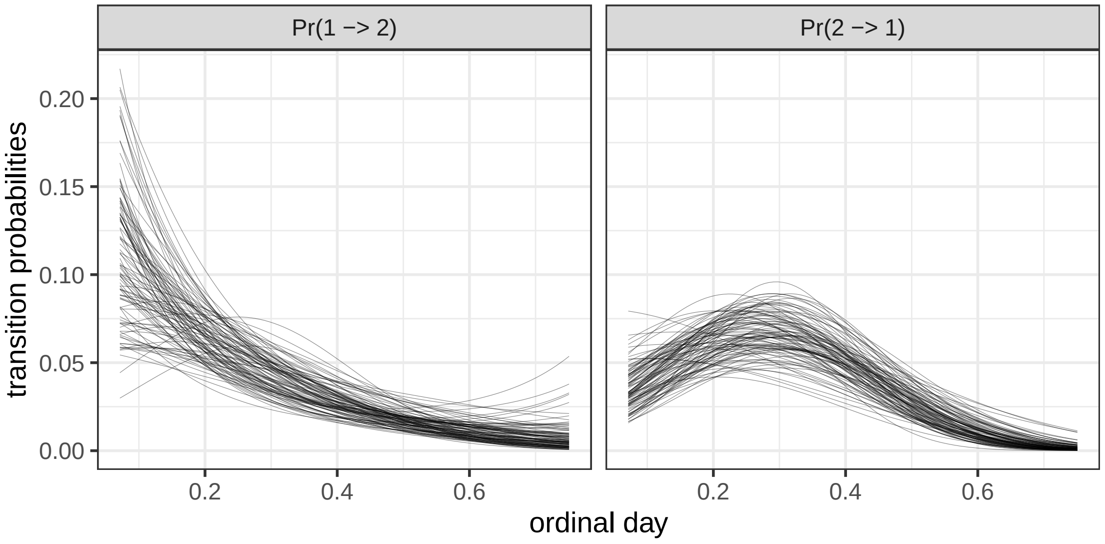
```r
ggplot(obsdist_df, aes(step, value, group = group, col = state)) +
    geom_line(linewidth = 0.1, alpha = 0.5) +
    scale_color_manual(values = hmmTMB:::hmmTMB_cols, name = NULL) +
    labs(x = "step length", y = "density")
```



4.4 Posterior samples of transition probabilities

Likewise, we can visualise the uncertainty in the relationship between the transition probabilities and the covariate (ordinal day). The code below show how we can use posterior samples of the regression coefficients to get posterior samples of the relationship.

```r
# Grid of ordinal day
newdata <- data.frame(day = seq(min(data$day), max(data$day), length = 100))

# Loop over randomly selected posterior samples
probs_list <- lapply(ind_post, function(i_post) {
    # Set parameters to a given posterior sample
    hmm$update_par(iter = i_post)

    # Relationship between transition probabilities and
    # covariate for these parameters
    tpm_post <- hmm$predict(what = "tpm", newdata = newdata)
    probs <- data.frame(prob = c(tpm_post[1,2,], tpm_post[2,1,]))

    probs$transition <- rep(paste0("Pr(", 1:2, " -> ", 2:1, ")"), each = 100)
    probs$group <- paste0("iter ", i_post, " - ", probs$transition)
    return(probs)
})

# Combine data frames for plot
probs_df <- do.call(what = rbind, args = probs_list)
probs_df$day <- newdata$day
ggplot(probs_df, aes(day, prob, group = group)) +
    geom_line(linewidth = 0.1, alpha = 0.5) +
    labs(x = "ordinal day", y = "transition probabilities") +
    facet_wrap("transition")
```



References

Jonsen, Ian D, W James Grecian, Lachlan Phillips, Gemma Carroll, Clive McMahon, Robert G Harcourt, Mark A Hindell, and Toby A Patterson. 2023. "aniMotum, an R Package for Animal Movement Data: Rapid Quality Control, Behavioural Estimation and Simulation." *Methods in Ecology and Evolution* 14 (3): 806–16.

Jonsen, Ian D, Patterson Toby A, 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.

Monnahan, Cole, and Kasper Kristensen. 2018. "No-u-Turn Sampling for Fast Bayesian Inference in ADMB and TMB: Introducing the Adnuts and Tmbstan r Packages." *PloS One* 13 (5). https://doi.org/10.1371/journal.pone.0197954.

Stan Development Team. 2019. "Stan Modeling Language User's Guide and Reference Manual, Version 2.29." Stan Development Team.

———. 2022. "RStan: The R Interface to Stan." https://mc-stan.org/.