Bayesian inference in hmmTMB

Théo Michelot

2022-12-15

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)).

1 Generating data

For the sake of demonstration, we use simulated data in this vignette. You can skip this section if you are not interested in the procedure used to generate artificial data. In the following chunk of code, we:

1. create an empty data set, just as a way to define the number of observations;
2. create a `MarkovChain` object for a 2-state hidden process (S_t), with transition probability matrix

$$\Gamma = \begin{pmatrix} 0.95 & 0.05 \\ 0.2 & 0.8 \end{pmatrix}$$

3. create an `Observation` object for the observation process (Z_t), defined such that

$$Z_t | \{S_t = 1\} \sim N(0, 3)$$
$$Z_t | \{S_t = 2\} \sim N(5, 1)$$

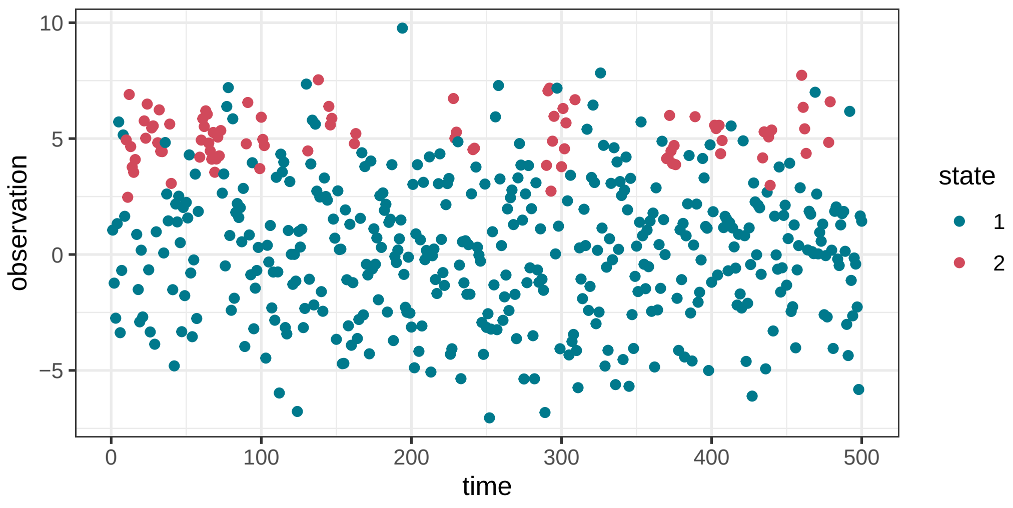
   That is, the observations follow a state-dependent normal distribution.

4. create an `HMM` object from the two model components;
5. simulate observations from the `HMM` object.

```r
# Create empty data set to specify number of observations
n <- 500
data_sim <- data.frame(z = rep(NA, n))

# Hidden state process
hid_sim <- MarkovChain$new(data = data_sim, n_states = 2,
                           tpm = matrix(c(0.95, 0.2, 0.05, 0.8), 2, 2))

# Observation process
par_sim <- list(z = list(mean = c(0, 5), sd = c(3, 1)))
obs_sim <- Observation$new(data = data_sim, dists = list(z = "norm"),
                           n_states = 2, par = par_sim)

# Create HMM and simulate observations
hmm_sim <- HMM$new(hid = hid_sim, obs = obs_sim)
data_sim <- hmm_sim$simulate(n = n, silent = TRUE)

head(data_sim)
```

```
  ID         z
1  1  1.052223
2  1 -1.233942
3  1 -2.746401
4  1  1.330035
5  1  5.717938
6  1 -3.372675
```

```r
# Plot simulated time series
state_sim <- factor(attr(data_sim, "state"))
ggplot(data_sim, aes(1:nrow(data_sim), z, col = state_sim)) +
  geom_point() +
  labs(x = "time", y = "observation") +
  scale_color_manual(values = pal, name = "state")
```



2 Model specification

We now turn to the specification of the model used for analysis.

2.1 Model structure

The steps used to create the model object are similar to the above. This time, the parameters passed as input are starting values, i.e., from where the sampler will start exploring parameter space. We choose values that are somwhat different from the ones used for simulation, but within a plausible range based on the simulated data. For the hidden state process, we use the default initial values (a matrix with 0.9 on the diagonal). We also set `initial_state = "stationary"`, which means that the initial distribution of the hidden state process is fixed to the stationary distribution of the Markov chain, rather than estimated. We do this here because the initial distribution parameters are often not well identified, which can lead to convergence issues in the MCMC sampling.

```
# Hidden state model
hid <- MarkovChain$new(data = data_sim, n_states = 2,
                       initial_state = "stationary")

# Initial parameters for observation process
par <- list(z = list(mean = c(2, 7), sd = c(4, 0.5)))
obs <- Observation$new(data = data_sim, dists = list(z = "norm"),
                       n_states = 2, par = par)

# Create HMM object
```

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

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.

```r
hmm$priors()
```

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

$coeff_fe_hid
                 mean sd
S1>S2.(Intercept)   NA NA
S2>S1.(Intercept)   NA NA

$log_lambda_obs
     mean sd

$log_lambda_hid
     mean sd
```

```r
hmm$coeff_fe()
```

```
$obs
```

```
                                [,1]
z.mean.state1.(Intercept)  2.0000000
z.mean.state2.(Intercept)  7.0000000
z.sd.state1.(Intercept)    1.3862944
z.sd.state2.(Intercept)   -0.6931472

$hid
                       [,1]
S1>S2.(Intercept) -2.197225
S2>S1.(Intercept) -2.197225
```

The observation model has four working parameters: the mean in each state (which is not transformed because its domain is already the real line), and the log standard deviation in each state. 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 (mean and standard deviation of prior). In this example, we choose the following priors:

$$
\begin{aligned}
\mu_1 &\sim N(0, 5^2) \\
\mu_2 &\sim N(0, 5^2) \\
\log(\sigma_1) &\sim N(\log(2), 5^2) \\
\log(\sigma_2) &\sim N(\log(2), 5^2)
\end{aligned}
$$

where μ_j and σ_j are the mean and standard deviation of the observation distribution for state $j \in \{1, 2\}$.

```r
# Parameter of normal priors for observation parameters
prior_obs <- matrix(c(0, 5,
                      0, 5,
                      log(2), 5,
                      log(2), 5),
                    ncol = 2, byrow = TRUE)
```

In a 2-state model, the two working parameters for the hidden state process are logit(γ_{12}) and logit(γ_{21}), where $\gamma_{ij} = \Pr(S_t = j | S_{t-1} = j)$ is the transition probability from state i to state j. As above, we can define a matrix with two columns to specify parameters of normal priors. We use the following priors,

$$
\begin{aligned}
\text{logit}(\gamma_{12}) &\sim N(-2, 1) \\
\text{logit}(\gamma_{21}) &\sim N(-2, 1)
\end{aligned}
$$

The mean 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). Note that the definition of the working parameters is a little more complicated in models with more than 2 states.

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

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
z.mean.state1.(Intercept) 0.0000000  5
z.mean.state2.(Intercept) 0.0000000  5
z.sd.state1.(Intercept)   0.6931472  5
z.sd.state2.(Intercept)   0.6931472  5

$coeff_fe_hid
                  mean sd
S1>S2.(Intercept)   -2  1
S2>S1.(Intercept)   -2  1

$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 `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 30 sec on a laptop.

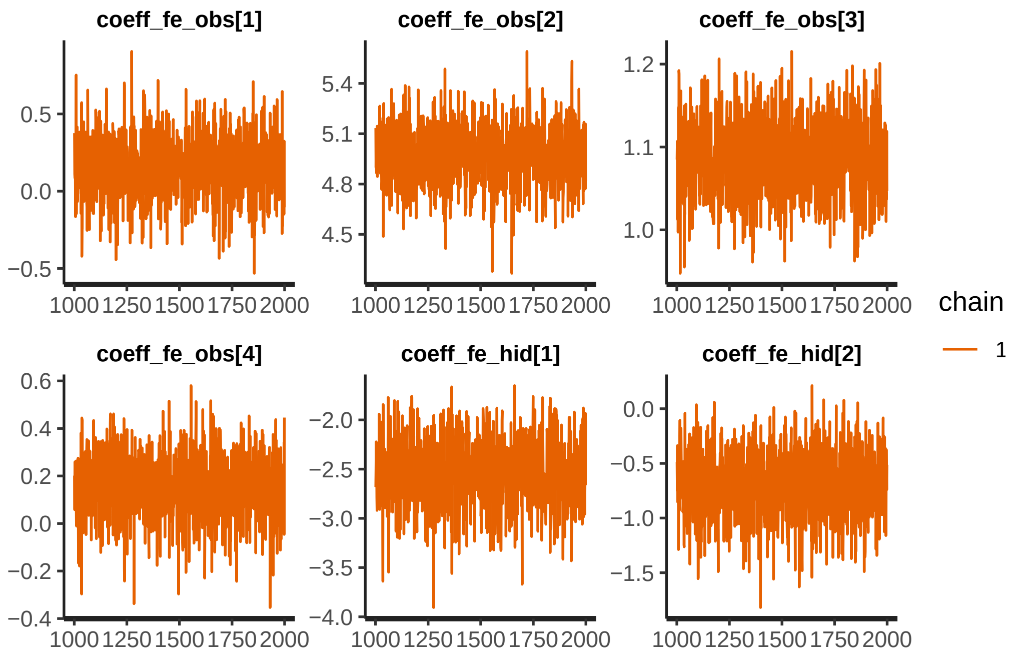
```
hmm$fit_stan(chains = 1, iter = 2000)
```

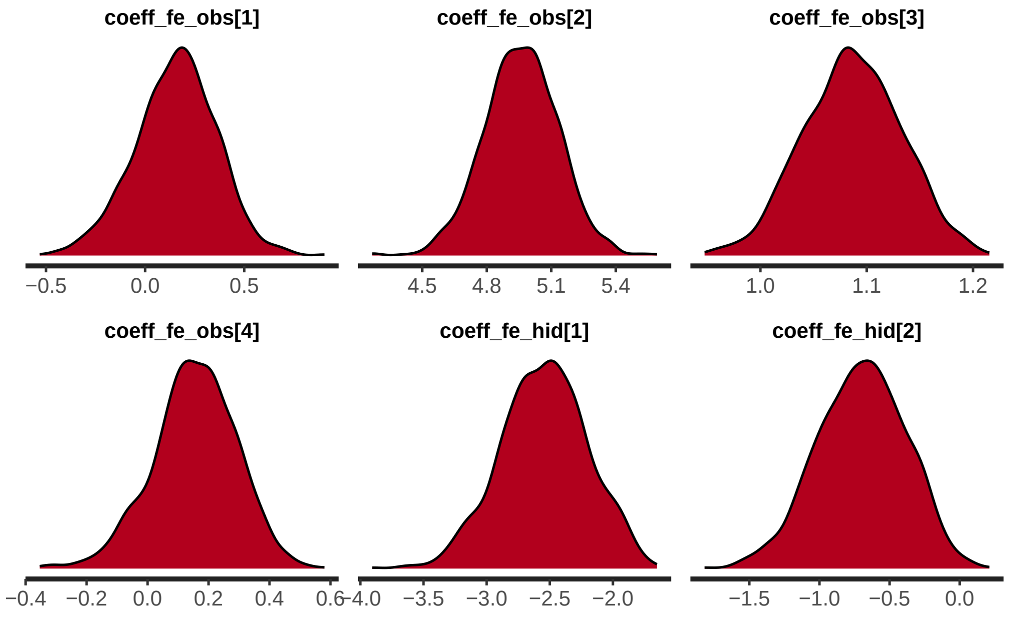
4 Inspecting the results

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 traceplots 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

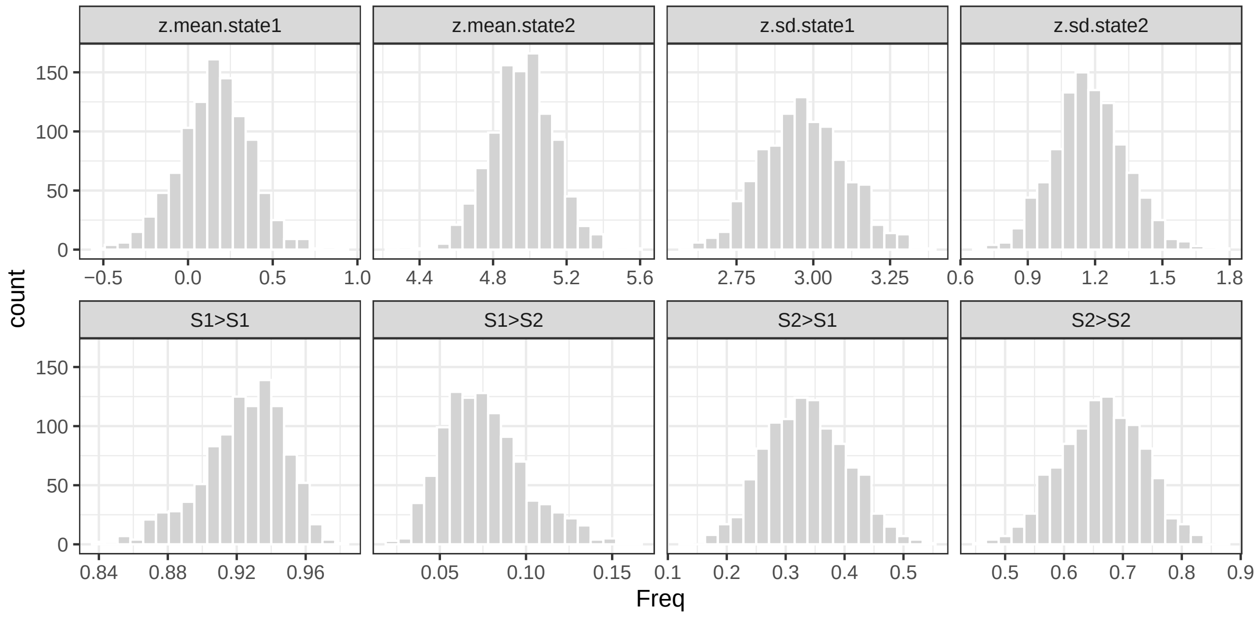
To inspect the posterior distributions of the natural parameters, which is often more interesting, we can extract posterior samples using `iters()`. This returns a matrix with one column for each parameter and one row for each MCMC iteration. It can directly be used to make traceplots, histograms, etc. It looks like the model successfully captured the true parameter values used for simulation. Note that this is an example of label switching, where states 1 and 2 are swapped compared to their order in the simulation model. This can often happen in HMMs, because the labelling of states is arbitrary.

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

```
     z.mean.state1 z.mean.state2 z.sd.state1 z.sd.state2     S1>S1      S1>S2
[1,]    0.37547464      5.133389    2.959224    1.223178 0.9359002 0.06409981
[2,]    0.11046972      4.899000    3.024539    1.059885 0.9239971 0.07600286
[3,]    0.08665420      5.043613    2.753133    1.295757 0.9022169 0.09778305
[4,]    0.18903353      4.888893    2.953271    1.182213 0.9247773 0.07522272
[5,]    0.07031309      4.864269    2.913320    1.282832 0.9180221 0.08197787
[6,]   -0.16511520      4.900028    2.710627    1.147088 0.9418129 0.05818708
         S2>S1     S2>S2
[1,] 0.3803550 0.6196450
[2,] 0.3209213 0.6790787
[3,] 0.4164832 0.5835168
```

```
[4,] 0.2987288 0.7012712
[5,] 0.3572066 0.6427934
[6,] 0.3516954 0.6483046
```

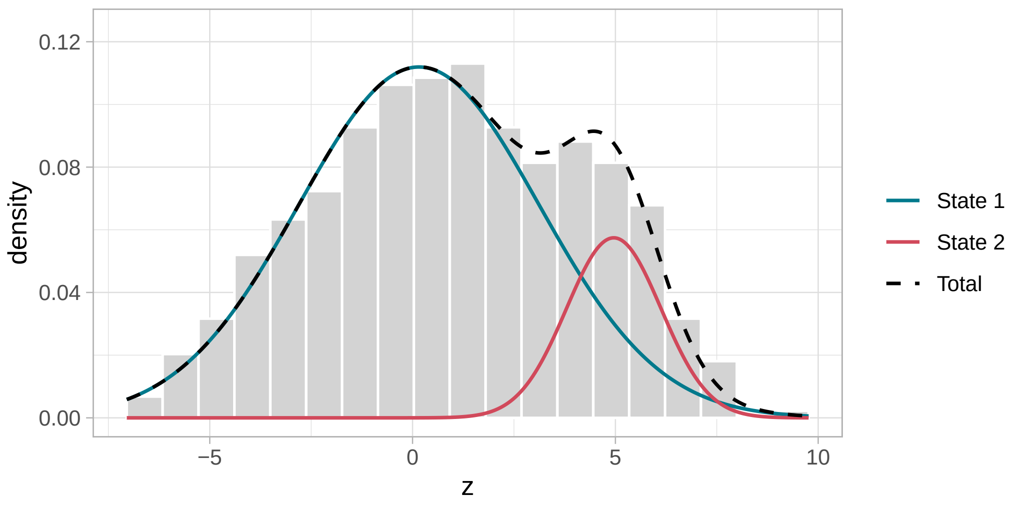
```
iters_df <- as.data.frame.table(iters)
ggplot(iters_df, aes(x = Freq)) +
    geom_histogram(bins = 20, fill = "lightgrey", col = "white") +
    facet_wrap("Var2", nrow = 2, scales = "free_x")
```



4.3 Plotting functions

We can also use other plotting functions as we would for a model fitted using `fit()`. By default, the parameter values used in that case are the posterior means. For example, we can plot the state-dependent distributions over a histogram of the data:

```
hmm$plot_dist("z")
```



References

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/.