Comparison of HMM packages

Théo Michelot

2024-03-31

This vignette presents the analysis of a simulated data set using several software packages for hidden Markov models, for comparison with hmmTMB. Specifically, we describe the syntax required to fit a 2-state hidden Markov model with normally-distributed observations in each package. That is, we consider the model

$$Z_t \mid S_t = j \sim N(\mu_j, \sigma_j^2), \quad j \in \{1, 2\},$$

where (S_t) is a 2-state Markov process with transition probability matrix $\mathbf{\Gamma}$.

1 Simulated data

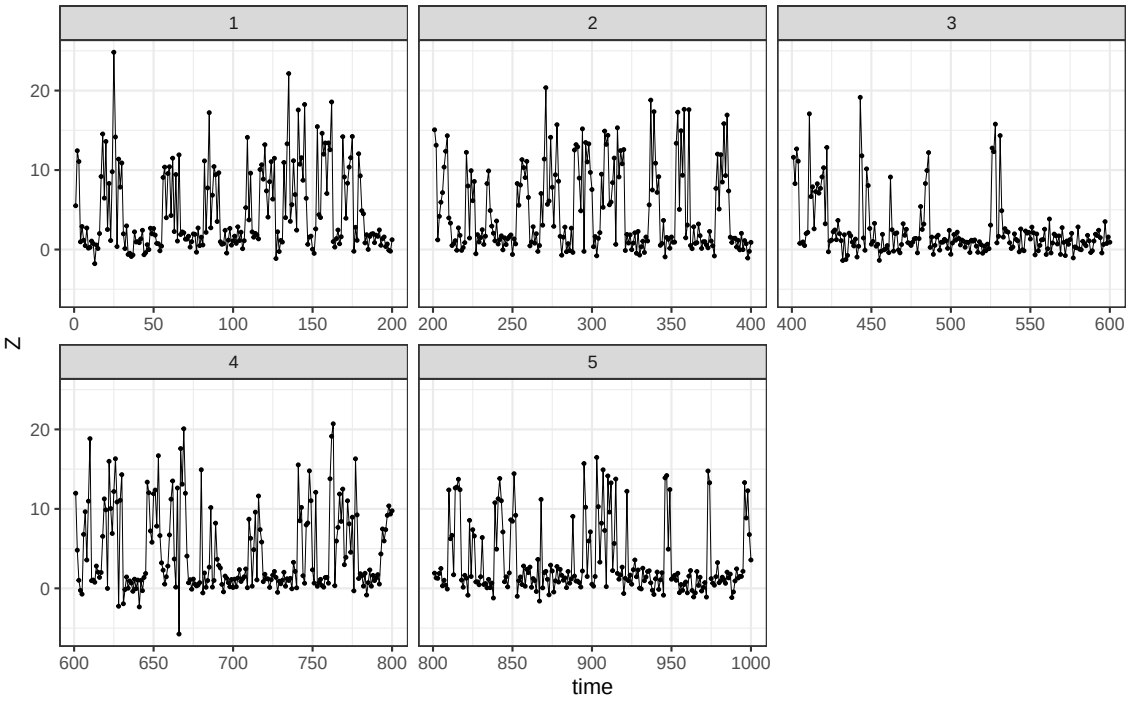
We simulated five time series, each of length 200, from a 2-state normal HMM with parameters

$$\mu_1 = 1, \quad \mu_2 = 10$$

$$\sigma_1 = 1, \quad \sigma_2 = 5$$

$$\Gamma = \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{pmatrix}$$

```
library(ggplot2)
theme_set(theme_bw())

ggplot(data, aes(time, Z)) +
    geom_point(size = 0.5) +
    geom_line(linewidth = 0.2) +
    facet_wrap("ID", scales = "free_x")
```



2 hmmTMB

The package hmmTMB is described by Michelot (2022), and its workflow is the following:

1. Create hidden state model object of class `MarkovChain`. This requires specifying a number of states, and optionally passing an initial transition probability matrix for the optimisation.
2. Create observation model object of class `Observation`. This requires specifying the family of distributions used for the observation variables in each state (here, `"norm"` for normal distributions). A list of initial parameter values must be passed.
3. Create the hidden Markov model object of class `HMM`.
4. Call methods on `HMM` object, such as `$fit()` to estimate the model parameters, and `$viterbi()` to estimate the most likely state sequence.

hmmTMB automatically detects the column named `ID` to identify the different time series in the data.

```
library(hmmTMB)

# Initial transition probabilities
tpm0 <- matrix(c(0.7, 0.3,
```

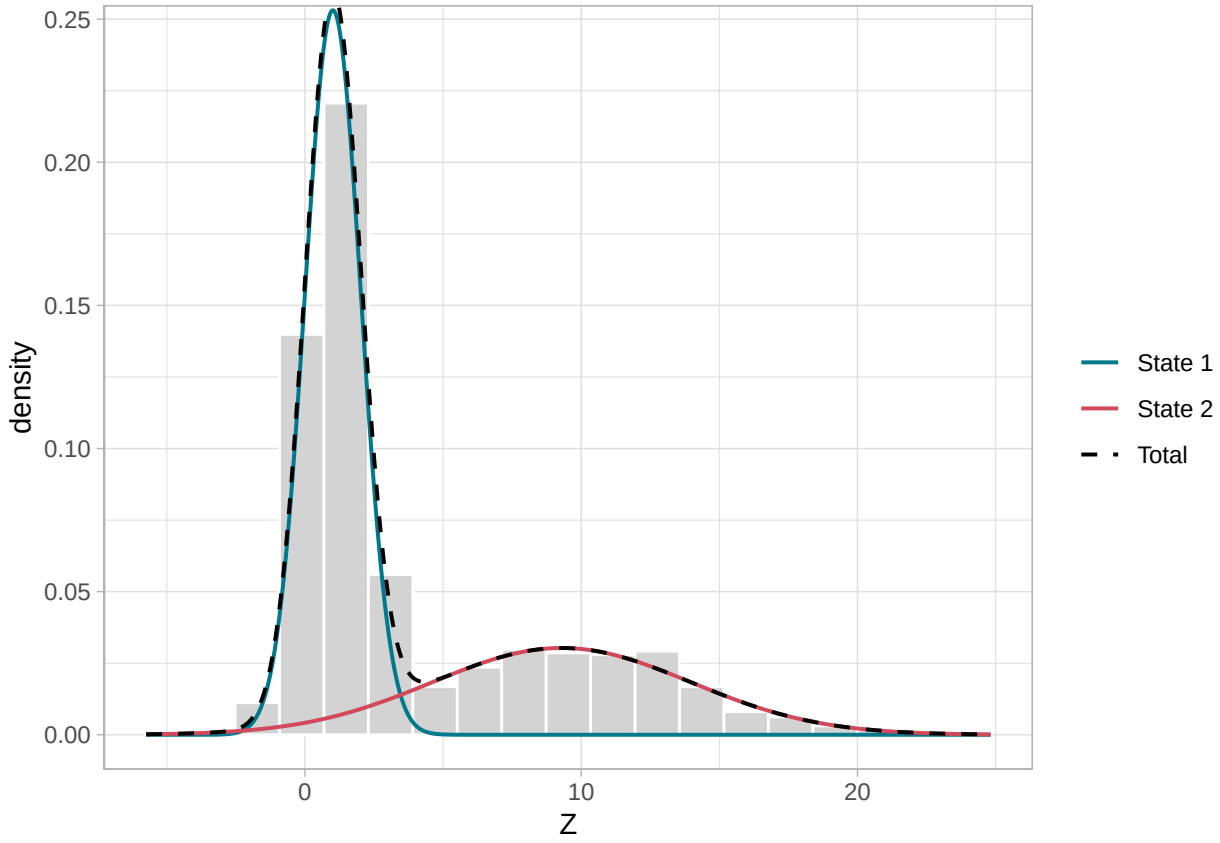
```
                0.3, 0.7),
              nrow = 2, byrow = TRUE)

# Define hidden state model
hid <- MarkovChain$new(data = data,
                       n_states = 2,
                       tpm = tpm0)

# Define observation model
obs <- Observation$new(data = data,
                       dists = list(Z = "norm"),
                       n_states = 2,
                       par = list(Z = list(mean = c(0, 15), sd = c(2, 10))))

# Create and fit HMM
mod1 <- HMM$new(obs = obs, hid = hid)
mod1$fit(silent = TRUE)

# Show parameters
lapply(mod1$par(), round, 3)
```

```
$obspar
, , 1

       state 1 state 2
Z.mean   1.012   9.292
Z.sd     1.017   4.672


$tpm
, , 1

        state 1 state 2
state 1   0.908   0.092
state 2   0.171   0.829
```

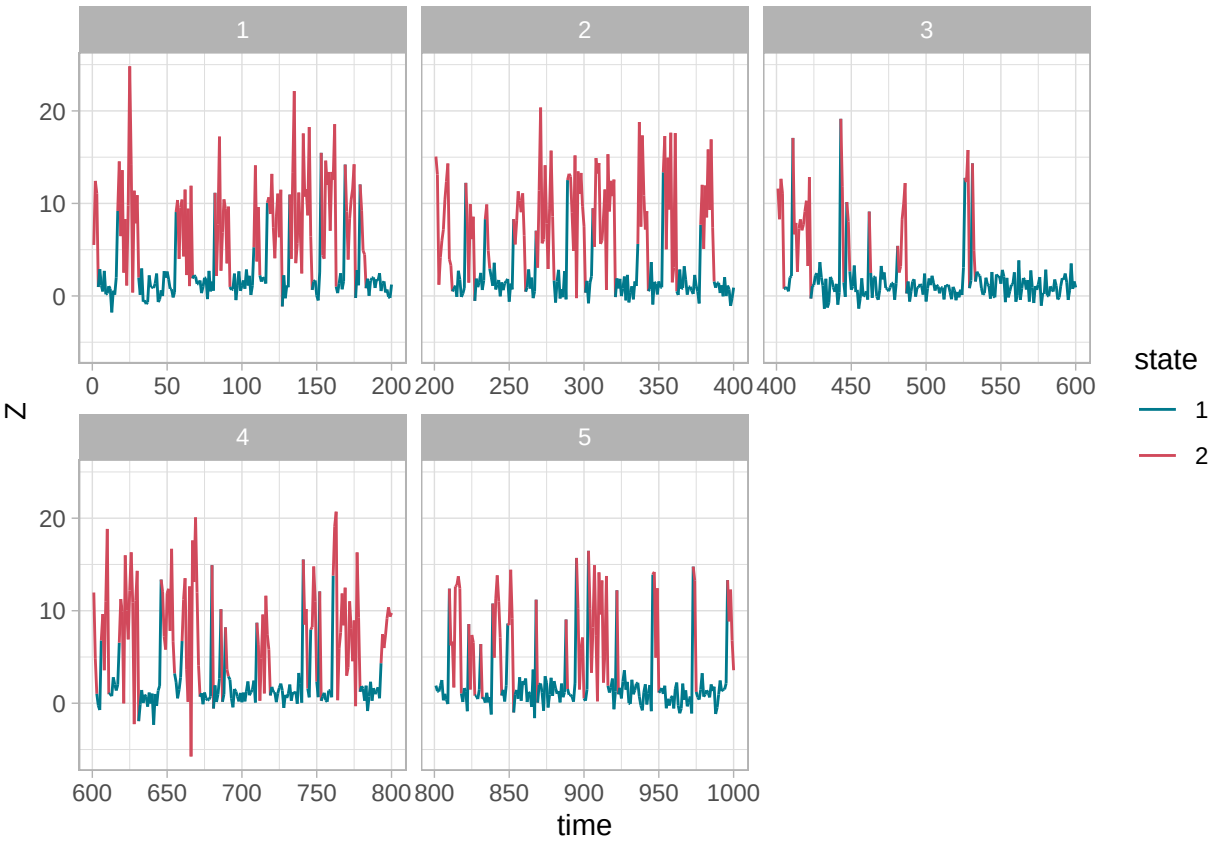
```r
# Most likely state sequence
s1 <- mod1$viterbi()
head(s1)
```

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

```r
# Model visualisation
mod1$plot_dist("Z")
```




```r
mod1$plot_ts("Z") +
    facet_wrap("ID", scale = "free_x")
```



3 depmixS4

depmixS4 is a flexible R package for hidden Markov models (Visser and Speekenbrink (2010)). Its main function is `depmix`, which is used to create a model object that can then be fitted. Its arguments include initial values for the transition probabilities and observation parameters, the family of observation distributions (`gaussian()`), and a vector of the lengths of the different time series in the data (`ntimes`). The functions `fit` and `viterbi` are used to estimate model parameters and decode the state process, respectively.

```
library(depmixS4)

# Lengths of time series
ntimes <- rle(data$ID)$lengths

# Create model
mod2 <- depmix(Z ~ 1,
               data = data,
               nstates = 2,
```

```
                family = gaussian(),
                ntimes = ntimes,
                trstart = tpm0,
                respstart = c(0, 2, 15, 10)) # mean1, sd1, mean2, sd2

# Fit model
fit2 <- fit(mod2)
```

```
converged at iteration 10 with logLik: -2287.146
```

```
# Get estimated parameters
summary(fit2, which = "all")
```

```
Initial state probabilities model
  pr1   pr2
0.815 0.185

Transition matrix
        toS1  toS2
fromS1 0.829 0.171
fromS2 0.093 0.907

Response parameters
Resp 1 : gaussian
    Re1.(Intercept) Re1.sd
St1           9.291  4.673
St2           1.012  1.017
```

```
# Most likely state sequence
s2 <- viterbi(fit2)$state
head(s2)
```

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

4 momentuHMM

The package momentuHMM was developed for the analysis of ecological data (in particular animal tracking data), but it is very general and can be used to implement many HMM formulations (McClintock and Michelot (2018)). We first use the function `prepData`, which

ensures that the data has the right format, with the argument `coordNames = NULL` to indicate that "coordinate" variables are not included. We then call `fitHMM`, passing initial values for the model parameters. Starting values for the transition probabilities need to be specified on the (multinomial logit) link scale, in a matrix with as many columns as there are non-diagonal transition probabilities. For a 2-state model, we can use `qlogis` to get the transformed parameters; in a model with more states, the multinomial logit link function would be needed to be implemented. (Note that `beta0` is an optional argument and, by default, `fitHMM` initialises the transition probability matrix to have large diagonal elements.)

```r
library(momentuHMM)

# Prepare data for momentuHMM
data_hmm <- prepData(data = data, coordNames = NULL)

# Link-transformed initial transition probabilities
beta0 <- matrix(qlogis(c(tpm0[1, 2], tpm0[2, 1])), nrow = 1)

# Fit HMM
mod3 <- fitHMM(data = data_hmm,
               nbStates = 2,
               dist = list(Z = "norm"),
               Par0 = list(Z = c(0, 15, 2, 10)),
               beta0 = beta0)

# Print parameters
mod3$mle[c("Z", "gamma")]
```

```
$Z
      state 1  state 2
mean 1.011507 9.290014
sd   1.016629 4.672859

$gamma
          state 1    state 2
state 1 0.9075065 0.09249346
state 2 0.1708621 0.82913787
```

```r
# Most likely state sequence
s3 <- viterbi(mod3)
head(s3)
```

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

5 LMest

LMest was developed by Bartolucci, Pandolfi, and Pennoni (2017) for the "analysis of longitudinal continuous and categorical data" using hidden Markov models (also called latent Markov models by its authors). The function `lmestCont()` implements hidden Markov models for continuous data, using state-dependent normal distributions. One difference with other packages is that LMest does not seem to allow for state-specific variances (only state-specific means) for the observation distribution, so the results are a little different.

```r
library(LMest)

# Fit model
mod4 <- lmestCont(responsesFormula = Z  ~ NULL,
                  latentFormula = ~ 1,
                  index = c("ID", "time"),
                  data = data,
                  k = 2,
                  modBasic = 1,
                  output = TRUE,
                  out_se = TRUE,
                  tol = 10^-1)
```

```
Missing data in the dataset dealt with as MAR
------------|-------------|-------------|-------------|-------------|-------------|
      k     |    start    |     step    |     lk      |    lk-lko   | discrepancy |
------------|-------------|-------------|-------------|-------------|-------------|
          2 |           0 |           0 |     -2949.4 |             |             |
          2 |           0 |           1 |    -2935.34 |      14.057 |     1.46256 |
------------|-------------|-------------|-------------|-------------|-------------|
```

```r
# State-dependent observation mean
mod4$Mu
```

```
   state
          1        2
  Z 1.802353 6.306233
```

```r
# Observation variance
mod4$Si
```

```
         [,1]
[1,] 22.85452
```

```r
# Coefficients for transition probs on logit scale
mod4$Ga
```

```
             logit
                      1         2
  (Intercept) -2.356609 -2.291829
```

```r
# Transform to transition probabilities
hid$par2tpm(mod4$Ga)
```

```
           [,1]       [,2]
[1,] 0.91345815 0.08654185
[2,] 0.09180195 0.90819805
```

6 hmmlearn (Python)

hmmlearn is one of the main Python packages for hidden Markov models (hmmlearn development team (2024)). We demonstrate its workflow below, which is similar to the R packages described previously, and requires specifying the number of states, the initial parameters, and so on.

```python
import numpy as np
import pandas as pd
from itertools import groupby
from hmmlearn import hmm

# Load data and extract observation variable
data = pd.read_csv('data.csv')
z = data[['Z']]
n = len(z)
```

```
# Find length of each time series
id = data[['ID']].values
lengths = [(sum(1 for i in g)) for k,g in groupby(id)]

# Create HMM
mod4 = hmm.GaussianHMM(n_components = 2,
                       init_params = 's')

# Set initial parameters
mod4.transmat_ = np.array([[0.7, 0.3],
                           [0.3, 0.7]])
mod4.means_ = np.array([[0], [15]])
mod4.covars_ = np.array([[2 ** 2], [10 ** 2]])

# Fit HMM
mod4.fit(z, lengths)
```

```
GaussianHMM(init_params='s', n_components=2)
```

```
# Extract estimated parameters
print(mod4.means_)
```

```
[[1.01154611]
 [9.29513704]]
```

```
print(mod4.covars_ ** 0.5)
```

```
[[[1.01678611]]

 [[4.66972488]]]
```

```
print(mod4.transmat_)
```

```
[[0.90735541 0.09264459]
 [0.1713001  0.8286999 ]]
```

```
# Most likely state sequence
s4 = mod4.decode(z)[1]
print(s4[:6])
```

```
[1 1 1 0 0 0]
```

References

Bartolucci, Francesco, Silvia Pandolfi, and Fulvia Pennoni. 2017. "LMest: An R Package for Latent Markov Models for Longitudinal Categorical Data." *Journal of Statistical Software* 81 (4): 1–38. https://doi.org/10.18637/jss.v081.i04.

hmmlearn development team. 2024. "Hmmlearn." *GitHub Repository*. https://github.com/hmmlearn/hmmlearn; GitHub.

McClintock, Brett T, and Théo Michelot. 2018. "momentuHMM: R Package for Generalized Hidden Markov Models of Animal Movement." *Methods in Ecology and Evolution* 9 (6): 1518–30.

Michelot, Théo. 2022. "hmmTMB: Hidden Markov Models with Flexible Covariate Effects in R." *arXiv Preprint arXiv:2211.14139*.

Visser, Ingmar, and Maarten Speekenbrink. 2010. "depmixS4: An R Package for Hidden Markov Models." *Journal of Statistical Software* 36: 1–21.