Analysing time series data with hidden Markov models in hmmTMB

Théo Michelot

2024-04-30

Contents

```r
# Load packages and color palette
library(ggplot2)
theme_set(theme_bw())
library(hmmTMB)
pal <- hmmTMB:::hmmTMB_cols
```

This vignette is a good starting point to learn about hmmTMB. It describes the main features of the package through one detailed example: the analysis of a data set of energy prices in Spain.

1 Data preparation

In hmmTMB, the data set to analyse must be passed as a data frame that contains the following variables:

- `ID` is a reserved column name, for the identifier of the time series (in cases where there are several time series in the data). If the data only consists of one time series, like in the following example, then this can be omitted.
- one column for each response variable ("data stream") that will be included in the model. There should be at least one of these.
- one column for each covariate that will be included in the model, if any.

There are two other reserved column names, which are optional: (1) `state` should only be included if the state process is known for some observations (see "Advanced features" vignette), and (2) `time` is used to check that the observations are equally spaced (if provided).

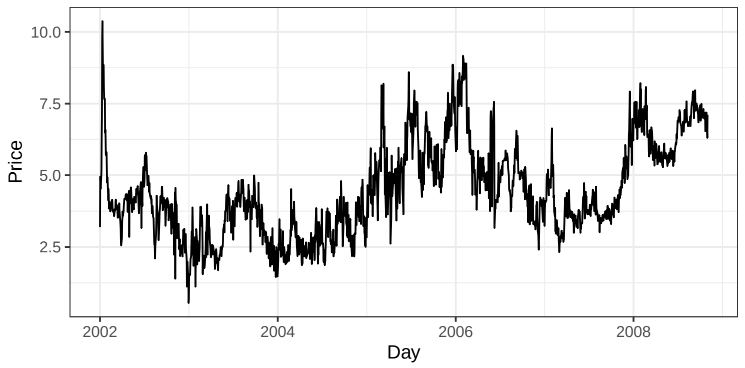
In this vignette, we will use the data set shown below, which is taken from the MSwM R package (Sanchez-Espigares and Lopez-Moreno (2021)). The column `Price` is the response variable that we want to model with an HMM, representing the price of energy in Spain (in cents/kWh) over 1784 working days from Jan 1, 2002 to Oct 31, 2008. All other columns are covariates related to the prices of raw matrials (`Oil`, `Gas`, `Coal`), to energy demand (`Demand`), and to the state of the financial market (`EurDol` and `Ibex35`). See `?MSwM::energy` for more detail about each variable.

```r
data(energy, package = "MSwM")
head(energy)
```

```
     Price      Oil      Gas     Coal   EurDol Ibex35   Demand
1 3.188083 22.43277 14.40099 38.35157 1.134687 8.3976 477.3856
2 4.953667 22.27263 19.02747 38.35157 1.106439 8.3771 609.1261
3 4.730917 22.65383 18.48417 38.35157 1.106684 8.5547 650.3715
4 4.531000 23.67657 18.30143 38.35157 1.116819 8.4631 647.0499
5 5.141875 23.67209 14.55602 38.35157 1.122965 8.1773 627.9698
6 6.322083 23.60534 15.22485 38.35157 1.122460 8.1866 693.2467
```

We add a column for the date, to make time series plots.

```
# Create a grid of days that excludes weekends
day1 <- as.POSIXct("2002/01/01", format = "%Y/%m/%d", tz = "UTC")
day2 <- as.POSIXct("2008/10/31", format = "%Y/%m/%d", tz = "UTC")
days_with_we <- seq(day1, day2, by = "1 day")
which_we <- which(format(days_with_we, '%u') %in% 6:7)
days <- days_with_we[-which_we]
energy$Day <- days

ggplot(energy, aes(Day, Price)) + geom_line()
```



2 Model specification

There are many good references presenting the mathematical formulation of HMMs in more or less detail; see for example Rabiner (1989) for a seminal introduction, Stamp (n.d.) for a computer scientist's introduction, McClintock et al. (2020) for a fairly non-technical description (with a focus on ecological applications), and Zucchini, MacDonald, and Langrock (2017) for a monograph. We will only provide some superficial description of the model formulation in this document.

An HMM is based on the assumption that the distribution of some observed variable(s) is driven by an unobserved state, i.e., each state gives rise to a different distribution for the observed variable. There are therefore two model components:

1. a model for the hidden state process (S_t), which is typically assumed to be a first-order Markov process;
2. a model for the observation process (Z_t), which may be multivariate. This model describes the distribution of the observation, conditional on the state process.

2.1 The R6 approach

hmmTMB uses R6 classes to implement model objects (Chang (2021)), the most important being `MarkovChain`, `Observation`, and `HMM`. With R6, we first need to create an object using the syntax

```
object <- Class$new(...)
```

where `Class` should be replaced by the class of the object that we create, and where the brackets contain arguments needed to create the object. (The details are described below.) Then, we can interact with the object using "methods", i.e., functions specific to that class. The syntax for this is slightly different from most R code, and looks something like

```
object$method(...)
```

2.2 Hidden state process

The hidden state process is stored as a `MarkovChain` object in hmmTMB. The main modelling decision required at this stage is the number of states of the model, i.e., the number of values that the hidden process S_t can take. Here, we choose $N = 2$ states, assuming that the observed energy price arises from one of two distributions, depending on the underlying state.

```
hid1 <- MarkovChain$new(data = energy, n_states = 2)
```

There are a few other arguments we could specify for more complex model formulations, and this is discussed in more detail in other parts of the vignette. In particular, we could pass an initial transition probabilitiy matrix with the argument `tpm`. This would then be used as a starting point for the parameter estimation. (By default, the transition probability matrix is initialised with 0.9 along the diagonal, and $0.1/N$ elsewhere.) We could also include covariates on the dynamics of the Markov chain, as illustrated later in this document, using the `formula` argument.

2.3 Observation model

The observation model, which defines the conditional distribution of the observations (given the state), is stored in an `Observation` object. Here, the response variable `Price` is strictly positive, but all values are pretty far from zero, so we use a normal distribution in each state. That is, we assume

$$Z_t|\{S_t = j\} \sim N(\mu_j, \sigma_j),$$

where $\mu_j > 0$ and $\sigma_j > 0$ are the mean and standard deviation of the distribution of price Z_t in state $j \in \{1, 2\}$. In hmmTMB, `"norm"` refers to the normal distribution. The distributions are passed in a named list, with one element for each response variable; here, there is only one.

In addition to the family of distribution, we need to choose initial parameter values for the observation process. This is because the model is fitted by numerical optimisation of the likelihood function, and the optimiser requires a starting point. The general idea is to pick values that are plausible given the data and the way we expect the states to be defined. We might for example use some quantiles of the observed prices for the mean parameter. These are also provided to the `Observation` object as a list, with the following structure:

- each element of the list corresponds to a response variable (here there is only one: `Price`), and is itself a list;
- each element of the inner list corresponds to a parameter (here, `mean` and `sd`), and is a vector of length the number of states.

We provide two initial means and two initial standard deviations (one for each state), for the price variable.

```r
# List observation distribution(s)
dists <- list(Price = "norm")
# List of initial parameter values
par0_2s <- list(Price = list(mean = c(3, 6), sd = c(1, 1)))
# Create observation model object
obs1 <- Observation$new(data = energy, dists = dists, par = par0_2s)
```

2.4 Hidden Markov model

An HMM is the combination of the hidden state and the observation model, and so we create an `HMM` object using the two components. Printing it in the command line shows some information about the model formulation, as well as initial parameter values (because the model hasn't been fitted yet). In cases where the model includes covariates, i.e., the parameters are time-varying, these initial values correspond to the first row of the data set (which is what `t = 1` means).

```r
hmm1 <- HMM$new(obs = obs1, hid = hid1)

hmm1
```

```
#######################
## Observation model ##
```

```
#######################
+ Price ~ norm(mean, sd)
  * mean.state1 ~ 1
  * mean.state2 ~ 1
  * sd.state1 ~ 1
  * sd.state2 ~ 1

> Initial observation parameters (t = 1):
           state 1 state 2
Price.mean       3       6
Price.sd         1       1

#########################
## State process model ##
#########################
        state 1 state 2
state 1       .      ~1
state 2      ~1       .

> Initial transition probabilities (t = 1):
        state 1 state 2
state 1     0.9     0.1
state 2     0.1     0.9
```

3 Model fitting

Model fitting consists in estimating all model parameters, in particular the transition probabilities of the state process, and the state-dependent observation parameters. Fitting this simple model takes less than a second on a laptop.

```
hmm1$fit(silent = TRUE)
```

3.1 Accessing the estimated parameters

Once it has been fitted, the model stores the estimated parameters, and we can see them by printing the model. Here, we find

$$\mu_1 = 3.4, \quad \mu_2 = 6.0$$
$$\sigma_1 = 0.8, \quad \sigma_2 = 1.1$$
$$\Gamma = \begin{pmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{pmatrix}$$

```
hmm1
```

```
#######################
## Observation model ##
#######################
+ Price ~ norm(mean, sd)
  * mean.state1 ~ 1
  * mean.state2 ~ 1
  * sd.state1 ~ 1
  * sd.state2 ~ 1

> Estimated observation parameters (t = 1):
           state 1 state 2
Price.mean   3.362   6.021
Price.sd     0.802   1.141

#########################
## State process model ##
#########################
        state 1 state 2
state 1       .      ~1
state 2      ~1       .

> Estimated transition probabilities (t = 1):
        state 1 state 2
state 1   0.992   0.008
state 2   0.011   0.989
```

The first state therefore captures periods with lower energy prices (with a mean of 3.4 cents/kWh), and the second state captures higher prices (mean = 6 cents/kWh).

In this simple model, the transition probabilities and the observation parameters are not covariate-dependent, and so we can directly interpret the values shown in the printed output as the estimated parameter values. However, note that, if covariates were included (either in the `MarkovChain` or `Observation` model), the corresponding HMM parameters would be time-varying. In that case, printing the model would show the parameters for the first time step of the data (as indicated by `t = 1` in the output).

In the case with covariates, a more flexible way to get estimated parameters is the method `HMM$par()`. By default, it returns the HMM parameters for the first row of data, but this can be changed with the argument `t`. For example, `hmm1$par(t = 10)` would return the estimated transition probabilities and observation parameters for the 10th data row. In a covariate-free model, the returned parameters are the same regardless of `t` because they are not time-varying. We discuss methods to access model parameters in the later section on "Including covariates in an HMM".

4 Model interpretation

4.1 State inference

It is often of interest to make inferences about the unobserved state process. In this example, we might want to know which time periods were more likely to be in the "high energy price" state. There are a few different functions for this purpose in hmmTMB.

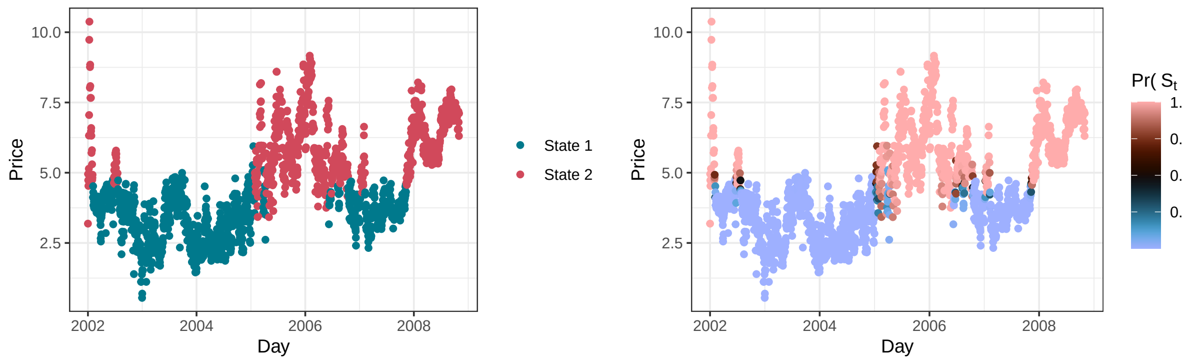
- `viterbi()` can be used for "global decoding", i.e., to obtain the most likely state sequence over the whole data set. This is done using the so-called Viterbi algortihm. It returns a vector of same length as the data set, where each element is the state for the corresponding time step. This may for example be useful to plot the time series, coloured by most likely state.

- `state_probs()` can be used for "local decoding", and returns a matrix of probabilities of being in each state at each time step. This gives a little more information than the Viterbi algorithm, and provides a measure of the uncertainty in the state classification. Note that, although the state probabilities and the Viterbi algorithm usually agree, they are not guaranteed to (see Zucchini, MacDonald, and Langrock (2017) for a discussion of the difference between local and global decoding).

- `sample_states()` implements the forward-filtering backward-sampling algorithm, and returns posterior samples of the state sequence. This is similar to the output of the Viterbi algorithm, but accounts for the uncertainty in the classification.

We create a new data set, for plotting, which includes the data variables and two other columns: one for the Viterbi state sequence, and one for the probability of being in state 2 (high price state).

```r
data_plot <- energy

# Coloured by most likely state sequence
data_plot$state <- factor(paste0("State ", hmm1$viterbi()))
ggplot(data_plot, aes(Day, Price, col = state)) +
  geom_point() +
  scale_color_manual(values = pal, name = NULL)

# Coloured by state probability
data_plot$pr_s2 <- hmm1$state_probs()[,2]
ggplot(data_plot, aes(Day, Price, col = pr_s2)) +
  geom_point() +
  scico::scale_color_scico(palette = "berlin",
                           name = expression("Pr("~S[t]~"= 2)"))
```



The second plot suggests that most time steps are classified with high probability of being in either state 1 or state 2, but there is high uncertainty (i.e., $\Pr(S_t = 1) \approx \Pr(S_t = 2) \approx 0.5$) for observations that have intermediate values.

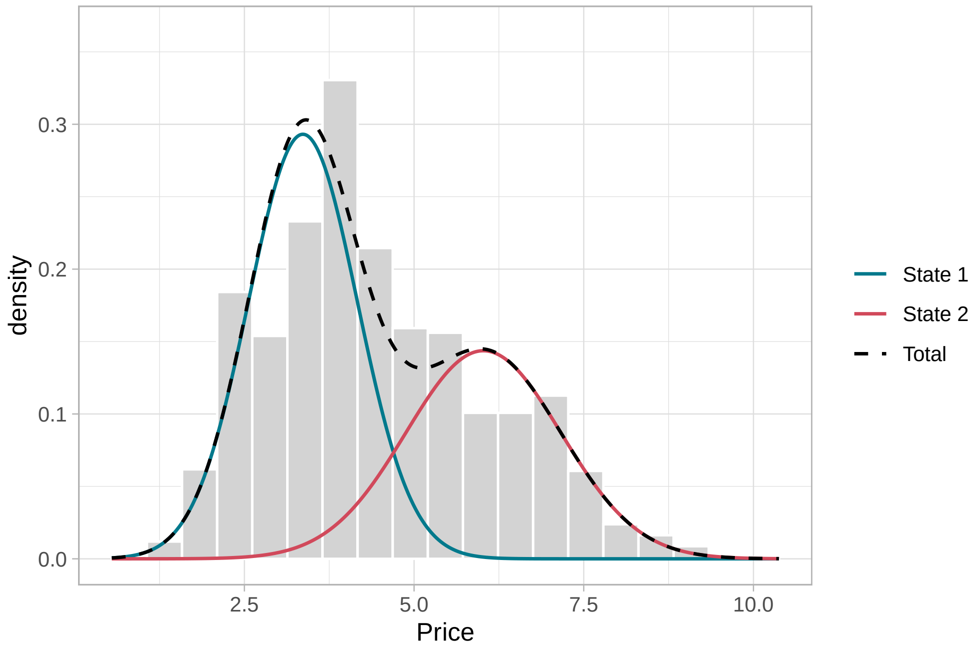
4.2 Model visualisation

There are several built-in functions to create plots of a fitted model in hmmTMB. For example, `hmm1$plot_ts("Price")` produces a time series plot of the `Price` variable, coloured by the most likely state sequence (similar to the one we made manually above).

To help interpreting the states, or to assess goodness-of-fit, it is often useful to plot the estimated state-dependent observations distributions. This is what the function `plot_dist()`

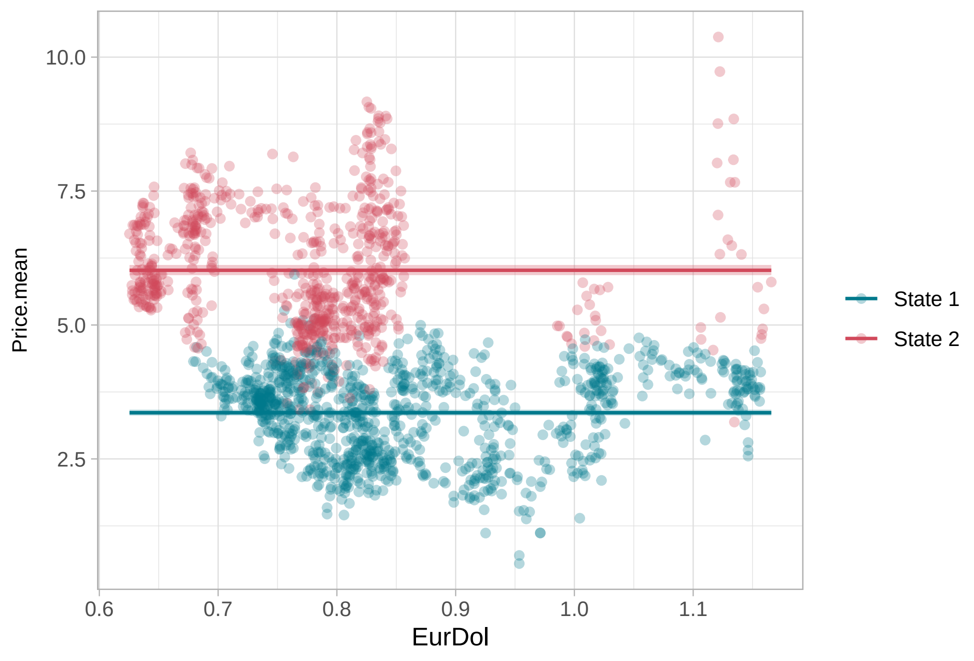
does. More specifically, it shows a histogram of the observations, overlaid with curves of the estimated distributions, weighted by the proportion of time spent in each state (measured from Viterbi sequence). From this plot, it is clear that state 1 tends to capture smaller prices, and state 2 higher prices, but there is some overlap between the two states for values around 5 cents/kWh.

```r
hmm1$plot_dist("Price")
```



In models that include covariates, the function `plot()` can be used to visualise the relevant model parameters as functions of covariates (e.g., transition probabilities, or observation parameters). This first model does not have covariates, but we will add some later so, for the sake of comparison, we show a plot of the mean of the price distribution in each state, as a function of the `EurDol` exchange rate variable. In that plot, we also include the observations, coloured by the Viterbi sequence.

```r
hmm1$plot("obspar", "EurDol", i = "Price.mean") +
  geom_point(aes(x = EurDol, y = Price, fill = state, col = state),
             data = data_plot, alpha = 0.3)
```



5 Model checking

5.1 Pseudo-residuals

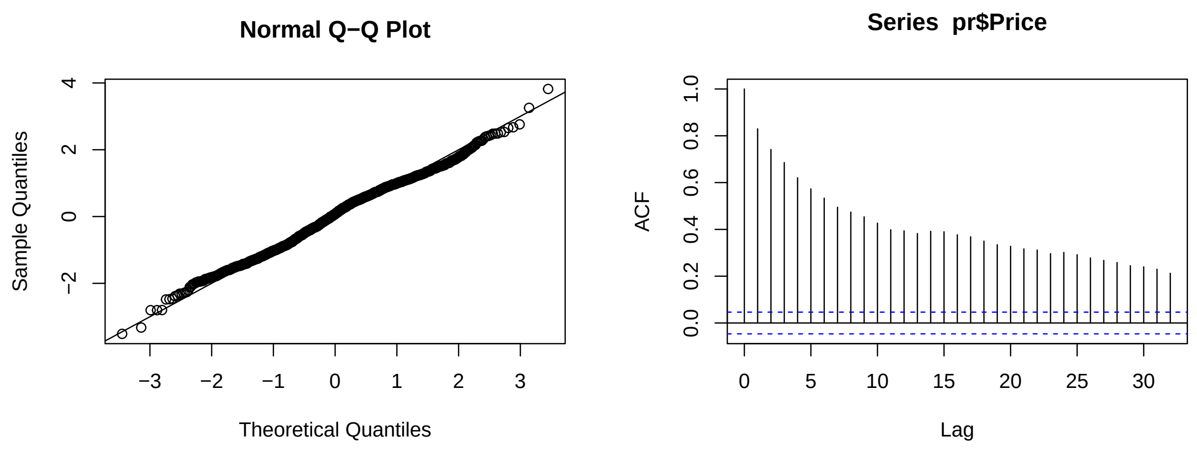
Pseudo-residuals are one method to assess goodness-of-fit of the observation distributions in an HMM (Zucchini, MacDonald, and Langrock (2017)). Similarly to residuals in linear models, pseudo-residuals should be independent and normally distributed if the model assumptions were satisfied. Deviations from these properties suggest lack of fit. We compute the pseudo-residuals for `Price` using the `pseudores()` function, and investigate the two properties (normal distribution and independence) separately, using a quantile-quantile plot and an autocorrelation function (ACF) plot, respectively.

```r
# Get pseudo-residuals in a matrix (one row for each response variable)
pr <- hmm1$pseudores()
```

```
Computing CDFs... done
Computing residuals for Price ... done
```

```r
# Check normality; should be along 1:1 diagonal
qqnorm(pr$Price)
abline(0, 1)

# Check independence; should decay to zero
```

```r
acf(pr$Price)
```



There is some deviation from the 1:1 diagonal in the quantile-quantile plot, but the fit seems adequate overall, suggesting that a mixture of 2 normal distributions works well for this data set. On the other hand, the ACF plot indicates that there is strong residual autocorrelation; i.e., much of the autocorrelation in the data was not captured by the model. This is clearly apparent from the time series plots shown above: successive prices within each state are not independent. There is no single remedy for this problem, but several approaches could be considered, including increasing the number of states, or including covariates to account for this autocorrelation.

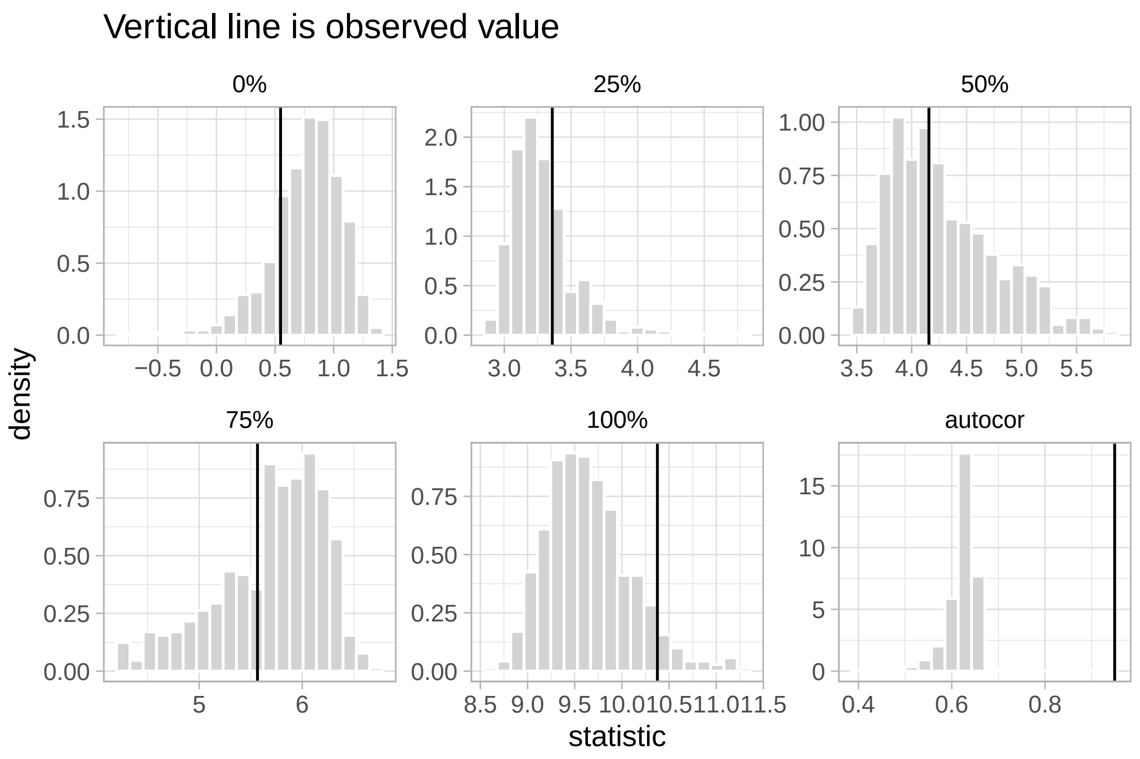
5.2 Posterior predictive checks

Another strategy to check goodness-of-fit is to generate simulated data from the fitted model, and then compare those to the observed data. Some relevant summary statistics can be compared between the true and simulated data sets using the `check()` function in hmmTMB. It takes as an argument a goodness-of-fit function, which itself takes a dataset as input and return the statistic(s) to use for comparison of observed and simulated data.

We choose the following statistics:

- the 0%, 25%, 50%, 75%, and 100% quantiles (i.e., min, median, max, and quartiles);
- the autocorrelation of the observation process, measured as the correlation between Z_{t-1} and Z_t.

```r
# This function returns the statistics that we want to compare,
# in a named vector
gof <- function(data) {
  s <- c(quantile(data$Price, seq(0, 1, by = 0.25)),
         autocor = cor(data$Price[-1], data$Price[-nrow(data)]))
}

# Run posterior predictive checks
checks <- hmm1$check(check_fn = gof, silent = TRUE, nsims = 500)

# Plot histograms of simulated statistics
checks$plot
```



The observed values of the 0%, 25%, 50%, and 75% quantiles are well within the range of simulated values, suggesting that those features are well captured by the model. The 100% quantile (i.e., the maximum) is almost always smaller in the simulations than in the real data, which means that the model underestimates how long the upper tail is. The autocorrelation is also captured poorly, as we also saw from the pseudo-residuals. Here, the distribution of simulated autocorrelations are between 0.55 and 0.65, whereas the true value is around 0.95.

6 Changing the number of states

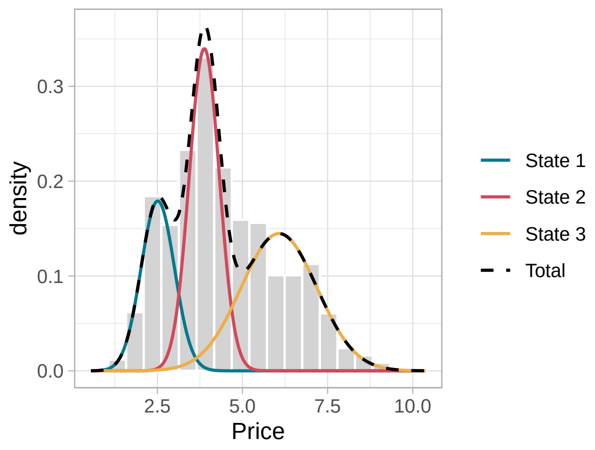
Using a different number of states requires very little changes in the code shown above. Consider that we now want to use $N = 3$ states. We need to specify `n_states = 3` when creating the `MarkovChain` component of the model. We also need to change the initial values for the `Observation` object, as we now need three values of each parameter. Everything else is identical.

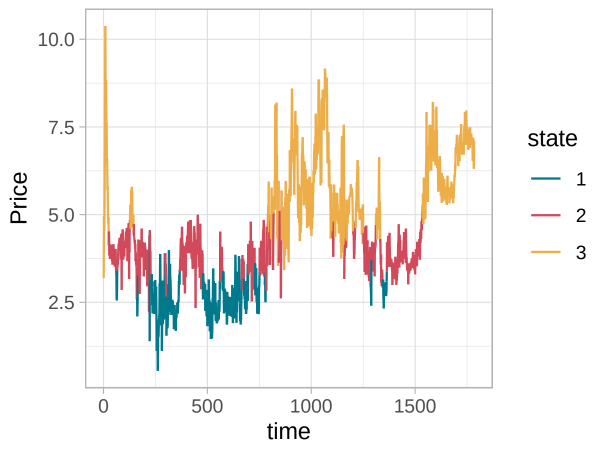
```r
# Hidden state process
hid2 <- MarkovChain$new(data = energy, n_states = 3)

# Observation model
par0_3s <- list(Price = list(mean = c(2.5, 4, 6), sd = c(0.5, 0.5, 1)))
obs2 <- Observation$new(data = energy, dists = dists, par = par0_3s)

# Create and fit HMM
hmm2 <- HMM$new(obs = obs2, hid = hid2)
hmm2$fit(silent = TRUE)

# Show parameters
hmm2
```

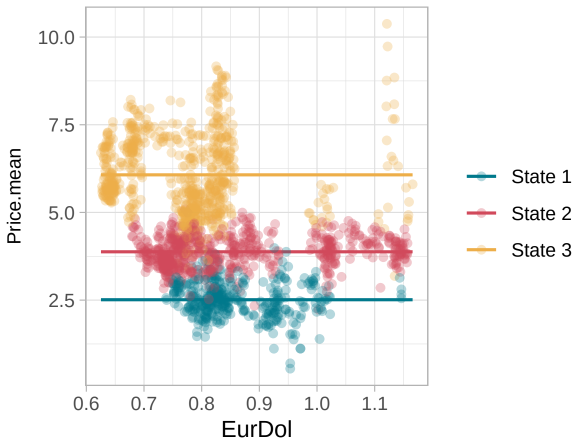
```
#######################
## Observation model ##
#######################
+ Price ~ norm(mean, sd)
  * mean.state1 ~ 1
  * mean.state2 ~ 1
  * mean.state3 ~ 1
  * sd.state1 ~ 1
  * sd.state2 ~ 1
  * sd.state3 ~ 1

> Estimated observation parameters (t = 1):
           state 1 state 2 state 3
Price.mean   2.511   3.878   6.072
Price.sd     0.491   0.441   1.114

#########################
```

```
## State process model ##
#########################
        state 1 state 2 state 3
state 1       .      ~1      ~1
state 2      ~1       .      ~1
state 3      ~1      ~1       .

> Estimated transition probabilities (t = 1):
        state 1 state 2 state 3
state 1   0.962   0.038   0.000
state 2   0.022   0.961   0.017
state 3   0.000   0.016   0.984
```

```
# Plot state-dependent distributions
hmm2$plot_dist("Price")

# Time series plot coloured by most likely state sequence
hmm2$plot_ts("Price")

# Plot prices against euro-dollar exchange rate,
# with estimated state-specific means
data_plot$state <- factor(paste0("State ", hmm2$viterbi()))
hmm2$plot("obspar", "EurDol", i = "Price.mean") +
  geom_point(aes(x = EurDol, y = Price, fill = state, col = state),
             data = data_plot, alpha = 0.3)
```







There is now one state for lower prices (state 1), one state for higher prices (state 3), and another state for intermediate prices (state 2). Choosing the number of states requires a trade-off between flexibility and interpretability: a model with more states is able to capture more detailed features of the data-generating process, but fewer states are typically easier to interpret. This issue is for example discussed by Pohle et al. (2017).

7 Including covariates in an HMM

One of the main functionalities of hmmTMB, compared to other R packages for hidden Markov modelling, is to allow for flexible covariate models on most model parameters. In particular, these can include linear and nonlinear relationships (the latter modelled using splines), and random effects. We showcase how covariates can be included in the two model components: the state process, or the observation model.

7.1 Covariates on transition probabilities

A popular extension of HMMs is to include the effects of covariates on the transition probabilities of the state process. This addresses questions of the type “Is the probability of switching from state 1 to state 2 affected by some covariate?”, or “Is the probability of being in state 1 affected by some covariate?”.

We investigate the effect of oil prices on the transition probabilities. Intuitively, we expect an effect because higher oil prices would likely lead to higher energy prices. The covariate dependence is specified through the argument `formula` when creating a `MarkovChain` object. This can either be an R formula (like in the example below), or a matrix of character strings where each element is the formula for the corresponding transition probability matrix. In the latter case, the diagonal should be filled with `"."` because the diagonal transition probabilities are not modelled directly. In this example, we assume the same formula for all transition probabilities: a linear relationship with `Oil`. Model specification is otherwise unchanged.

```r
# State process model
f <- ~ Oil
hid3 <- MarkovChain$new(data = energy, n_states = 2, formula = f)

# Observation model
obs3 <- Observation$new(data = energy, dists = dists, par = par0_2s)

# Fit HMM
hmm3 <- HMM$new(obs = obs3, hid = hid3)
hmm3$fit(silent = TRUE)
```

The regression coefficients which describe the relationship between the covariate and the transition probabilities can be printed using the `coeff_fe()` function. The estimated coefficients suggest that oil price has a positive effect on $\Pr(S_{t+1} = 2|S_t = 1)$ (coeff = 0.06), and a negative effect on $\Pr(S_{t+1} = 1|S_t = 2)$ (coeff = -0.07). That is, the process is more likely to transition to state 2 and to stay in state 2 when oil is expensive, which is consistent with our intuition.

```r
hmm3$coeff_fe()
```

```
$obs
                                   [,1]
Price.mean.state1.(Intercept)  3.3644286
Price.mean.state2.(Intercept)  6.0292220
Price.sd.state1.(Intercept)   -0.2203988
Price.sd.state2.(Intercept)    0.1278369

$hid
                         [,1]
S1>S2.(Intercept) -7.21869570
S1>S2.Oil          0.06264683
S2>S1.(Intercept) -1.23941310
S2>S1.Oil         -0.06729944
```

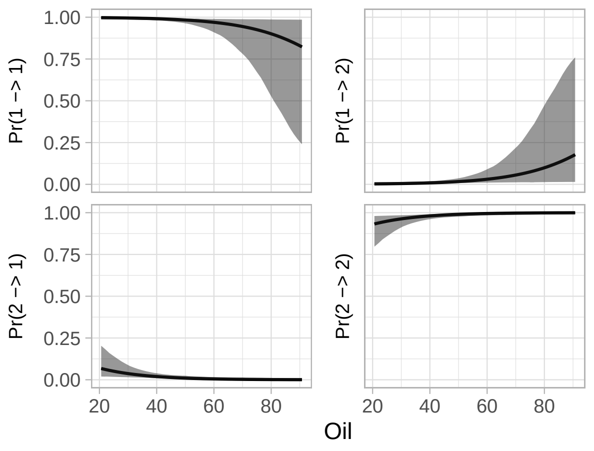
Confidence intervals on the estimated model parameters can also be derived using `confint()`, with an optional argument for the level of the interval (default: 0.95 for 95% confidence interval). The 95% confidence intervals for the effect of oil on the two transition probabilities don't overlap with zero.

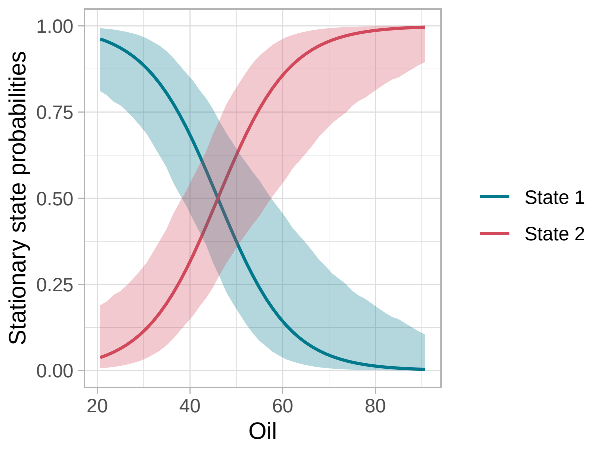
```r
round(hmm3$confint(level = 0.95)$coeff_fe$hid, 2)
```

```
                                      se
S1>S2.(Intercept) -7.22 -9.78 -4.66 1.31
S1>S2.Oil          0.06  0.01  0.12 0.03
S2>S1.(Intercept) -1.24 -3.49  1.01 1.15
S2>S1.Oil         -0.07 -0.12 -0.02 0.03
```

The relationship can also be visualised (with confidence bands), by plotting the transition probabilities as functions of the covariate. We can also plot the stationary state probabilities, which measure the probability of being in each state in the long run, over a grid of covariate values. The latter option can sometimes help with interpretation, like in this example: in the second plot, it is clear that the probability of being in state 2 increases with oil price.

```r
# Plot transition probabilities as functions of oil price
hmm3$plot("tpm", var = "Oil")

# Plot stationary state probabilities as functions of oil price
hmm3$plot("delta", var = "Oil")
```





We used the formula `~Oil` to include a linear effect of oil price. We could include a nonlinear effect with a formula like `~ s(Oil, k = 5, bs = "cs")`. We illustrate spline models below, when including covariates on the observation parameters.

7.2 Covariates on observation parameters

Another option to include covariates in an HMM is to assume an effect on the parameters of the state-dependent observation distributions. The question of interest is then "In each state, does the distribution of observations depend on some covariate?". This is a powerful tool which, as we will see below, can be used to implement state-switching regression models. However, note that interpretation can become difficult in such models. For example, if the mean energy price depends not only on the currently active state, but also on some covariate, what is the interpretation of each state? What if the mean is sometimes higher in state 1 and sometimes higher in state 2 (depending on the covariate)?

In the following, we consider an analysis similar to that of Langrock et al. (2017), where the effect of the euro-dollar exhange rate is included as a covariate.

7.2.1 Linear model

We now assume that the mean energy price in each state is a linear function of the euro-dollar exchange rate. This corresponds to the Markov-switching generalised linear model, an extension of Markov-switching regression:

$$\begin{aligned} Z_t | \{S_t = j\} &\sim N(\mu_j, \sigma_j) \\ \mu_j &= \beta_0^{(j)} + \beta_1^{(j)} x_{1t}, \end{aligned}$$

where $\beta_0^{(j)}$ is the intercept, and $\beta_1^{(j)}$ the slope for the exchange rate covariate x_{1t} in state $j \in \{1, 2\}$.

We specify this model using the `formulas` attribute of the `Observation` object. This is defined as a nested list, with one element for each parameter of each observed variable.

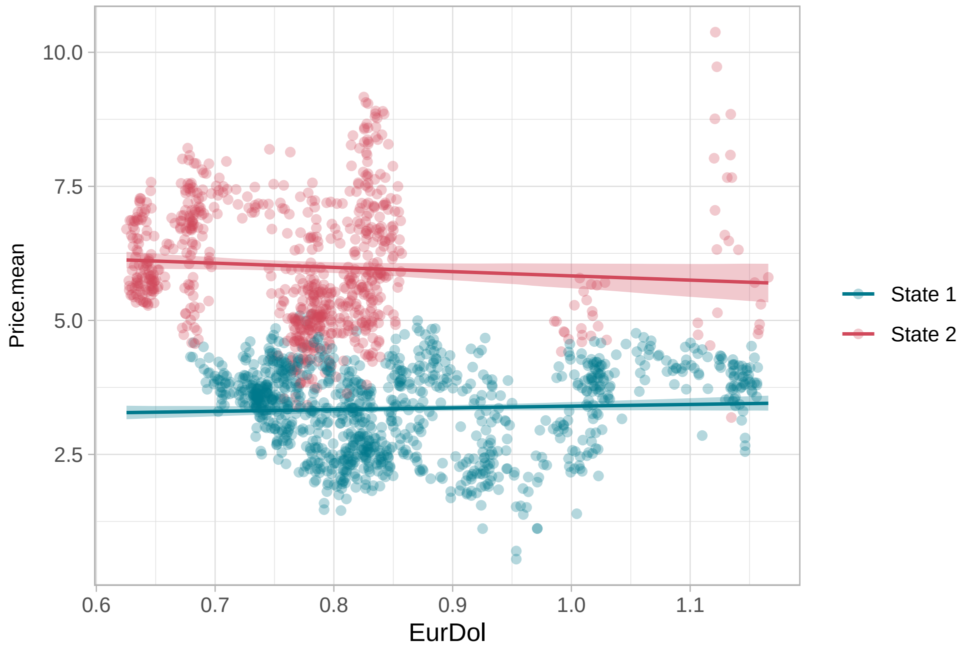
```r
# State process
hid4 <- MarkovChain$new(data = energy, n_states = 2)

# Observation model
f <- list(Price = list(mean = ~ EurDol))
obs4 <- Observation$new(data = energy, dists = dists,
                        par = par0_2s, formulas = f)

# Fit HMM
hmm4 <- HMM$new(obs = obs4, hid = hid4)
hmm4$fit(silent = TRUE)

# Look at regression coefficients
hmm4$coeff_fe()
```

```
$obs
```

```
                                  [,1]
Price.mean.state1.(Intercept)  3.0790748
Price.mean.state1.EurDol       0.3207086
Price.mean.state2.(Intercept)  6.6210952
Price.mean.state2.EurDol      -0.7912204
Price.sd.state1.(Intercept)   -0.2258342
Price.sd.state2.(Intercept)    0.1338693

$hid
                          [,1]
S1>S2.(Intercept) -4.891296
S2>S1.(Intercept) -4.553888
```

The estimated coefficients are $\hat{\beta}_1^{(1)} = -2.3$ and $\hat{\beta}_1^{(1)} = -0.9$, suggesting that the euro-dollar exchange rate has a negative effect on mean energy price in each state. We can visualise this relationship with the `plot` function.

```
data_plot$state <- factor(paste0("State ", hmm4$viterbi()))
hmm4$plot("obspar", "EurDol", i = "Price.mean") +
  geom_point(aes(x = EurDol, y = Price, fill = state, col = state),
             data = data_plot, alpha = 0.3)
```



7.2.2 Nonlinear model

The plot of `Price` against `EurDol` shown above indicates that the relationship between those two variables is quite complex, and probably nonlinear. Nonlinear models can be fitted in hmmTMB, using some model specification functionalities from the R package mgcv, which implements generalised additive models. This gives great flexibility to capture complex covariate effects.

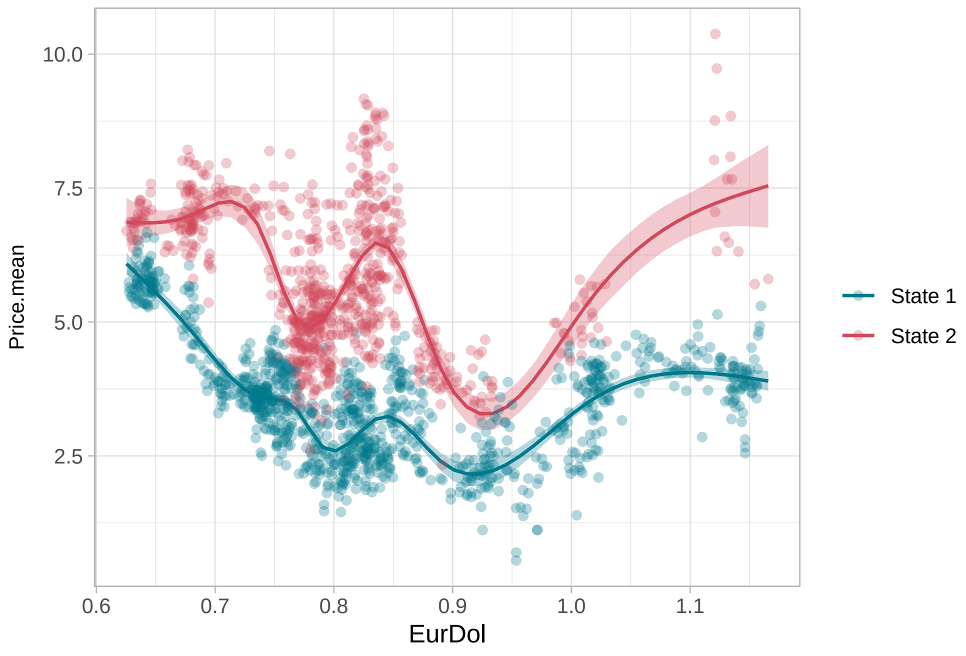
To estimate the effect of the euro-dollar exchange rate on mean price, we adapt the linear model shown above to

$$\begin{aligned} Z_t | \{S_t = j\} &\sim N(\mu_j, \sigma_j) \\ \mu_j &= \beta_0^{(j)} + f_1^{(j)}(x_{1t}), \end{aligned}$$

where $f_1^{(j)}$ is a smooth function, modelled using splines. This defines a Markov-switching generalised additive model, as described for example by Langrock et al. (2017). For more details about spline-based models, see for example Wood (2017).

We update the linear formula passed to the `Observation` object, following the mgcv syntax, to `~ s(EurDol, k = 8, bs = "cs")`. This indicates that we want to use a cubic spline basis of dimension 8. We can create and fit the model as before. This takes longer than the linear model, because the formulation is much more flexible, and more parameters need to be estimated. In particular, similarly to generalised additive models, the smoothness of the relationship is estimated from the data, rather than assumed, which can be computational.

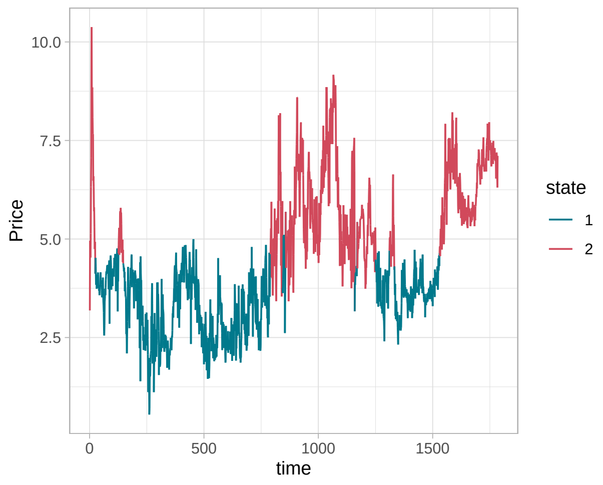
```r
hid5 <- MarkovChain$new(data = energy, n_states = 2)
f <- list(Price = list(mean = ~s(EurDol, k = 10, bs = "cs")))
obs5 <- Observation$new(data = energy, dists = dists,
                        par = par0_2s, formulas = f)
hmm5 <- HMM$new(obs = obs5, hid = hid5)

hmm5$fit(silent = TRUE)

data_plot$state <- factor(paste0("State ", hmm5$viterbi()))
hmm5$plot("obspar", "EurDol", i = "Price.mean") +
  geom_point(aes(x = EurDol, y = Price, fill = state, col = state),
             data = data_plot, alpha = 0.3)
```

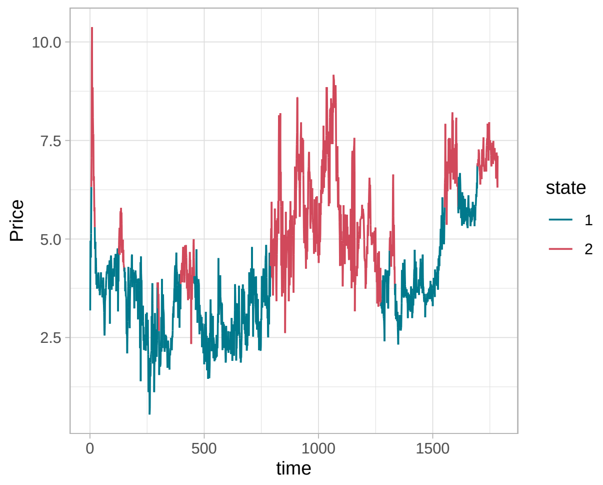


These results illustrate both the flexibility of non-parametric models to capture complex relationships between HMM parameters and covariates, and the challenge of interpreting the results. The definition of states 1 and 2 is not as clear as before, because the mean price in state 2 for some values of the covariate is lower than the mean price in state 1 for some other values.

We can compare the most likely state sequences in the two models, which are somewhat different.

```
hmm4$plot_ts("Price")
hmm5$plot_ts("Price")
```





7.3 Random effects

Using the convenient mgcv syntax, it is also possible to specify random effects on the transition probabilities or the observation parameters. This may be particularly useful in studies where there are multiple time series, which should be pooled into a common model, while accounting for heterogeneity between them.

In the energy example, we only have one time series, so we (rather artificially) use the year as the group variable for a random effect on the mean parameter of the price in each state. That is, we consider the model

$$
\begin{aligned}
Z_t | \{S_t = j\} &\sim N(\mu_j, \sigma_j) \\
\log(\mu_j) &= \beta_0^{(j)} + \gamma_k^{(j)}
\end{aligned}
$$

where $k \in \{2002, 2003, \dots, 2008\}$ is the index for the year, and where the year-specific intercepts are assumed to be independent and identically distributed in each state, as

$$\gamma_k^{(j)} \sim N(0, s_j^2)$$

In this model, the observation process has six parameters to estimate:

- the standard deviation of price in each state, σ_1 and σ_2;
- the shared intercept in each state, $\beta_0^{(1)}$ and $\beta_0^{(2)}$;
- the variance of the random intercepts in each state, s_1^2 and s_2^2.

For the analysis, we first extract the year for each row of the `energy` data set. Note that we treat it as a factor (categorical) variable here, rather than a continuous variable.

```r
energy$Year <- factor(format(energy$Day, "%Y"))
head(energy$Year)
```

```
[1] 2002 2002 2002 2002 2002 2002
Levels: 2002 2003 2004 2005 2006 2007 2008
```

To implement the model described above, we need to include `Year` as a random effect on the state-dependent observation parameters. More specifically, it should affect the mean of the price variable in each state. To indicate that it is a random effect, we use the mgcv syntax `s(Year, bs = "re")`. Model fitting takes a little longer than in model with fixed covariate effects, because TMB needs to integrate over the random effects in this example.

```r
# Formula for RE of year on mean price
f <- list(Price = list(mean = ~ s(Year, bs = "re")))
```

```
# Create Observation object with RE formula
obs6 <- Observation$new(data = energy, dists = dists,
                        par = par0_2s, formula = f)

# Create hidden state process
hid6 <- MarkovChain$new(data = energy, n_states = 2)
# Create and fit HMM
hmm6 <- HMM$new(obs = obs6, hid = hid6)
hmm6$fit(silent = TRUE)
```

We can see estimates of all the model parameters listed above using the `coeff_fe()` and `sd_re()` functions on the `Observation` model object. The former returns σ_j and $\beta_0^{(j)}$, and the latter returns s_j for each state.

```
# Fixed effect coefficients
round(obs6$coeff_fe(), 2)
```

```
                                 [,1]
Price.mean.state1.(Intercept)  3.84
Price.mean.state2.(Intercept)  5.68
Price.sd.state1.(Intercept)   -0.41
Price.sd.state2.(Intercept)   -0.35
```

```
# Std dev of random effects
round(obs6$sd_re(), 2)
```

```
                            [,1]
Price.mean.state1.s(Year) 0.84
Price.mean.state2.s(Year) 1.71
```

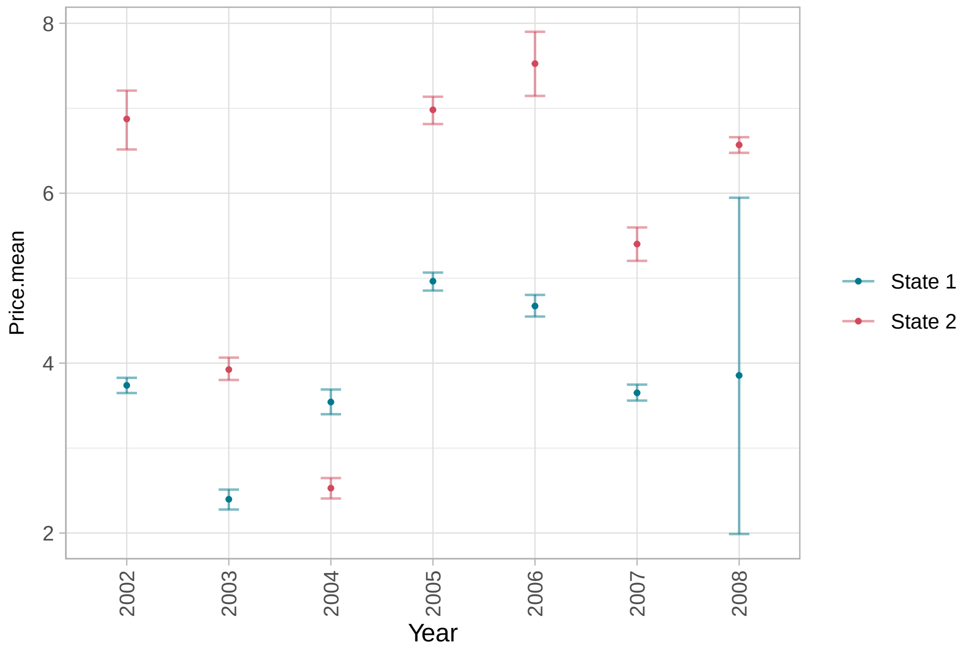
The year-specific random intercepts $\gamma_k^{(j)}$ are also predicted, and can be printed using `coeff_re()`. For each state, there are 7 levels: one for each year from 2002 to 2008.

```
# Predicted random intercepts
round(obs6$coeff_re(), 2)
```

```
                               [,1]
Price.mean.state1.s(Year).1 -0.11
Price.mean.state1.s(Year).2 -1.45
Price.mean.state1.s(Year).3 -0.30
Price.mean.state1.s(Year).4  1.12
Price.mean.state1.s(Year).5  0.83
Price.mean.state1.s(Year).6 -0.19
Price.mean.state1.s(Year).7  0.01
Price.mean.state2.s(Year).1  1.19
Price.mean.state2.s(Year).2 -1.76
Price.mean.state2.s(Year).3 -3.16
Price.mean.state2.s(Year).4  1.30
Price.mean.state2.s(Year).5  1.84
Price.mean.state2.s(Year).6 -0.28
Price.mean.state2.s(Year).7  0.88
```

There is no clear pattern in the predicted random intercepts here. The mean energy price in each state can also be plotted against the year, to visualise the between-year heterogeneity with uncertainty bounds.

```
hmm6$plot("obspar", "Year", i = "Price.mean")
```



7.4 Accessing the estimated parameters

7.4.1 Linear predictor coefficients and HMM parameters

When covariates are included in any component of the HMM, it is important to distinguish between two types of parameters that can be extracted from the model:

- The linear predictor coefficients, which describe the (fixed or random) effects of covariates on the HMM parameters. They can be accessed with the methods `HMM$coeff_fe()` (fixed effects) and `HMM$coeff_re()` (random effects). The fixed effect coefficients include for example the intercepts and linear effects. The random effect coefficients include things like random intercepts and basis coefficients for a non-linear effect.

- The HMM parameters themselves, i.e., transition probabilities and parameters of the state-dependent observation distributions. Importantly, these parameters are covariate-dependent, and so, to get their values, we need to fix the covariates. The HMM parameters shown when the model is printed correspond to the first row of the data set, i.e., the covariates are set to their values in the first data row. (The output shows `t = 1` as a reminder.) The method `HMM$par()` can be used to get parameter values corresponding to any data row, using the argument `t`. For example, `t = 100` returns HMM parameters obtained by setting the covariates to their values in the 100th row of data. The argument `t` can also be a vector of multiple values, to get HMM parameters at several time steps, and `t = "all"` returns HMM parameters for all data rows. In the output of `HMM$par()`, the parameters are returned as an array, with one layer (one "slice") for each data row selected through `t`.

7.4.2 Illustration

Let's use `hmm5`, in which the observation model contains the non-linear effect of a covariate, to go through the outputs of those different methods.

The method `coeff_fe()` returns a list with two elements: `obs` (for the observation model) and `hid` (for the hidden state model). Here, it only contains intercept coefficients (one for each estimated HMM parameter), because there are no linear covariate effects included. If the linear effect of a covariate was included, this is where we would find the estimated slope. Note that, although there are four transition probabilities in a two-state model, only two intercepts are shown for the hidden state model: one for the probability of a transition from state 1 to state 2, and one for the probability of a transition from state 2 to state 1. This is because these are the only two transition probabilities that are actually estimated; the other two transition probabilities (from 1 to 1, and from 2 to 2) are derived based on the constraint that rows of the transition probability matrix must sum to 1.

```
# Fixed effect coefficients of the linear predictor
hmm5$coeff_fe()
```

```
$obs
                                     [,1]
Price.mean.state1.(Intercept)  3.43202598
Price.mean.state2.(Intercept)  5.86204119
Price.sd.state1.(Intercept)   -0.54571840
Price.sd.state2.(Intercept)   -0.05294928

$hid
                         [,1]
S1>S2.(Intercept) -4.747164
S2>S1.(Intercept) -4.583396
```

The method `coeff_re()` also returns a list with one element for `obs` and one for `hid`. The latter is empty here because there are no random effects or non-linear terms in the hidden state model. This model includes a non-linear effect of the covariate `EurDol` on the mean price in both states. In each state, this is modelled with a spline constructed with 9 basis functions (because we chose `k = 10` in the formula). So, `coeff_re()` shows the 18 estimated basis coefficients; the first 9 describe the relationship between mean price and covariate in state 1, and the last 9 describe the relationship in state 2.

```r
# Random effect coefficients of the linear predictor
hmm5$coeff_re()
```

```
$obs
                                      [,1]
Price.mean.state1.s(EurDol).1  0.5871880
Price.mean.state1.s(EurDol).2 -0.6524441
Price.mean.state1.s(EurDol).3 -0.8164244
Price.mean.state1.s(EurDol).4 -1.6125780
Price.mean.state1.s(EurDol).5 -1.5214130
Price.mean.state1.s(EurDol).6 -1.0989498
Price.mean.state1.s(EurDol).7 -1.7318545
Price.mean.state1.s(EurDol).8 -0.7900151
Price.mean.state1.s(EurDol).9  0.1659304
Price.mean.state2.s(EurDol).1  0.8585674
Price.mean.state2.s(EurDol).2  0.4691970
Price.mean.state2.s(EurDol).3 -1.0836836
Price.mean.state2.s(EurDol).4 -1.1659430
Price.mean.state2.s(EurDol).5 -0.2061459
Price.mean.state2.s(EurDol).6  0.2876016
Price.mean.state2.s(EurDol).7 -2.0357765
Price.mean.state2.s(EurDol).8 -0.8316627
Price.mean.state2.s(EurDol).9  1.5688614

$hid
     [,1]
```

The method `par()` gives us a list with elements `obspar` (observation parameters) and `tpm` (transition probability matrix). Say that we want to know what the HMM parameters are for the 1st and 10th rows of data; to get these, we specify `t = c(1, 10)`. Each element of the list is an array with as many slice as there are time steps in the argument `t`, i.e., two layers in the example below because `t` is of length 2. The observation parameters are different at `t = 1` and `t = 10` because they depend on covariates. However, the transition probabilities are the same on both layers of the array, because they don't depend on covariates.

```
# HMM parameters at t = 1 and t = 10
hmm5$par(t = c(1, 10))
```

```
$obspar
, , 1

             state 1   state 2
Price.mean 4.0004409 7.3241680
Price.sd   0.5794254 0.9484281

, , 2

             state 1   state 2
Price.mean 4.0346223 7.2051195
Price.sd   0.5794254 0.9484281



$tpm
, , 1

           state 1     state 2
state 1 0.99139836 0.008601638
state 2 0.01011673 0.989883269

, , 2
```

```
            state 1     state 2
state 1 0.99139836 0.008601638
state 2 0.01011673 0.989883269
```

7.4.3 Uncertainty quantification

This distinction is also important for uncertainty quantification. The method `HMM$confint()` returns confidence intervals for the linear predictor coefficients, whereas `HMM$predict()` returns confidence intervals for the HMM parameters (for some given covariate values). Note that `HMM$predict()` can generally be useful in cases where we want to predict the HMM parameters for some new values of the covariates (rather than for a given row of the data set).

8 Extensions and other features

8.1 Model predictions

From a model that includes covariates, we can predict the HMM parameters using the function `predict()`, possibly with confidence intervals. It takes as input the new data set, and the component of the model that should be predicted (either `tpm` for transition probabilities, `delta` for stationary state probabilities, and `obspar` for observation parameters). Here, we consider the model with oil price as a covariate on the transition probabilities as an example. If we want confidence intervals, we also need to specify the argument `n_post`, which is the number of posterior samples used to approximate the confidence interval.

```r
# Covariate data for predictions
newdata <- data.frame(Oil = c(20, 80))

# Predict transition probabilities
hmm3$predict(what = "tpm", newdata = newdata, n_post = 1e3)
```

```
$mle
, , 1

            state 1     state 2
state 1 0.99744147 0.002558534
state 2 0.07008385 0.929916147


, , 2
```

```
            state 1   state 2
state 1 0.900871797 0.0991282
state 2 0.001327183 0.9986728


$lcl
, , 1

           state 1      state 2
state 1 0.98903711 0.0005993775
state 2 0.02119709 0.7658564180

, , 2

             state 1    state 2
state 1 0.5120443171 0.01372046
state 2 0.0001537039 0.99015694


$ucl
, , 1

          state 1    state 2
state 1 0.9994006 0.01096289
state 2 0.2341436 0.97880291

, , 2

            state 1   state 2
state 1 0.986279545 0.4879557
state 2 0.009843063 0.9998463
```

```
# Predict stationary state probabilities
hmm3$predict(what = "delta", newdata = newdata, n_post = 1e3)
```

```
$mle
        state 1    state 2
[1,] 0.96477905 0.03522095
```

```
[2,] 0.01321166 0.98678834

$lcl
         state 1    state 2
[1,] 0.807800815 0.00603029
[2,] 0.000939756 0.83301454

$ucl
       state 1   state 2
[1,] 0.9939697 0.1921992
[2,] 0.1669855 0.9990602
```

8.2 Simulating from an HMM

The function `simulate()` can be used to simulate from an `HMM` model object. This first generates one realisation from the hidden Markov chain, and then simulates observations based on the hidden state and the observation model. We need to pass to `simulate()` the number `n` of observations that should be simulated and, if the model includes covariate dependences, a data frame `data` with columns for the covariates.

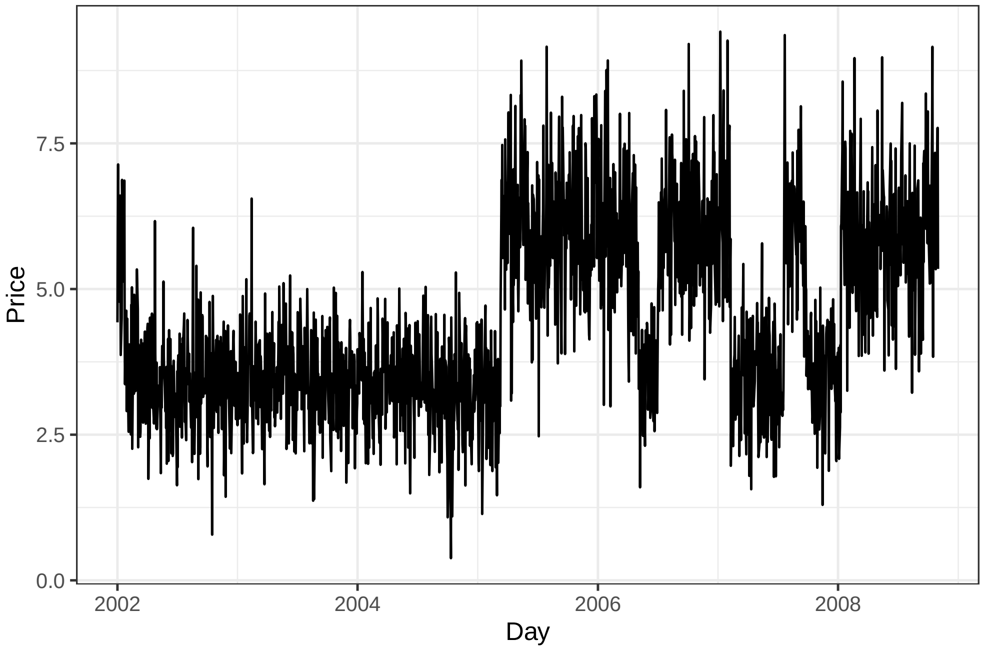
In the code below, we simulate from the model `hmm3`, which included the effect of oil prices on the transition probabilities. Plotting the simulated time series gives some insights into how well the model captures features of the real data. In particular, the simulated time series does not display the same autocorrelation as the observations.

```
# Simulate a time series of same length as data
sim_data <- hmm3$simulate(n = nrow(energy), data = energy, silent = TRUE)
head(sim_data)
```

```
     Price      Oil      Gas     Coal   EurDol Ibex35   Demand        Day Year
1 4.428623 22.43277 14.40099 38.35157 1.134687 8.3976 477.3856 2002-01-01 2002
2 6.037817 22.27263 19.02747 38.35157 1.106439 8.3771 609.1261 2002-01-02 2002
3 7.138635 22.65383 18.48417 38.35157 1.106684 8.5547 650.3715 2002-01-03 2002
4 5.083465 23.67657 18.30143 38.35157 1.116819 8.4631 647.0499 2002-01-04 2002
5 4.783178 23.67209 14.55602 38.35157 1.122965 8.1773 627.9698 2002-01-07 2002
6 5.063611 23.60534 15.22485 38.35157 1.122460 8.1866 693.2467 2002-01-08 2002
  ID
1  1
2  1
```

```
3   1
4   1
5   1
6   1
```

```r
# Plot simulated prices
ggplot(sim_data, aes(Day, Price)) + geom_line()
```



References

Chang, Winston. 2021. *R6: Encapsulated Classes with Reference Semantics*. https://CRAN.R-project.org/package=R6.

Langrock, Roland, Thomas Kneib, Richard Glennie, and Théo Michelot. 2017. "Markov-Switching Generalized Additive Models." *Statistics and Computing* 27 (1): 259–70.

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." *Ecology Letters* 23 (12): 1878–1903.

Pohle, Jennifer, Roland Langrock, Floris M van Beest, and Niels Martin Schmidt. 2017. "Selecting the Number of States in Hidden Markov Models: Pragmatic Solutions Illustrated Using Animal Movement." *Journal of Agricultural, Biological and Environmental Statistics* 22 (3): 270–93.

Rabiner, Lawrence R. 1989. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* 77 (2): 257–86.

Sanchez-Espigares, Josep A., and Alberto Lopez-Moreno. 2021. *MSwM: Fitting Markov Switching Models*. https://CRAN.R-project.org/package=MSwM.

Stamp, Mark. n.d. "A Revealing Introduction to Hidden Markov Models." http://www.cs.sjsu.edu/faculty/stamp/RUA/HMM.pdf.

Wood, Simon N. 2017. *Generalized Additive Models: An Introduction with R*. CRC press.

Zucchini, Walter, Iain L MacDonald, and Roland Langrock. 2017. *Hidden Markov Models for Time Series: An Introduction Using R, Second Edition*. CRC Press.