*Journal of Statistical Software*

*MMMMMM YYYY, Volume VV, Issue II.* *doi: 10.18637/jss.v000.i00*

# fHMM: Hidden Markov Models for Financial Time Series in R

**Lennart Oelschläger**
Bielefeld University

**Timo Adam**
University of St Andrews

**Rouven Michels**
Bielefeld University

**Abstract**

Hidden Markov models constitute a versatile class of statistical models for time series that are driven by hidden states. In financial applications, the hidden states can often be linked to market regimes such as bearish and bullish markets or recessions and periods of economics growth. To give an example, when the market is in a nervous state, corresponding stock returns often follow some distribution with relatively high variance, whereas calm periods are often characterized by a different distribution with relatively smaller variance. Hidden Markov models can be used to explicitly model the distribution of the observations conditional on the hidden states and the transitions between states, and thus help us to draw a comprehensive picture of market behavior. While various implementations of hidden Markov models are available, an R package that is tailored to financial applications is still lacking. In this paper, we introduce the R package **fHMM**, which provides various tools for applying hidden Markov models to financial time series. It contains functions for fitting hidden Markov models to data, conducting simulation experiments, and decoding the underlying state sequence. Furthermore, functions for model checking, model selection, and state prediction are provided. In addition to basic hidden Markov models, hierarchical hidden Markov models are implemented, which can be used to jointly model multiple data streams that were observed at different temporal resolutions. The aim of the **fHMM** package is to give R users with an interest in financial applications access to hidden Markov models and their extensions.

*Keywords*: hidden Markov models, hierarchical hidden Markov models, regime switching, financial time series, decoding market behavior, R.

## 1. Introduction

In recent years, Hidden Markov models (HMMs) have emerged as a popular tool for modeling time series that are subject to state-switching over time (Zucchini, MacDonald, and Langrock 2016). In their basic form, HMMs comprise an observed state-dependent process that is

driven by a hidden state process, the latter of which is typically modeled using a discrete-time, finite-state Markov chain. In financial applications, the states of the underlying Markov chain can often be linked to market regimes such as bearish and bullish markets or recessions and periods of economics growths. To give an example, when the market is in a nervous state, corresponding stock returns often follow some distribution with relatively high variance, whereas when the market is in a clam state, another distribution with relatively smaller variance is active. By their dependence structure, HMMs naturally account for such disparate patterns and thus allow us to infer hidden market regimes and their underlying dynamics from financial time series.

Over the last decades, HMMs have become increasingly popular in finance. In various studies, they have been applied to model business cycles (Kim and Nelson 1998; Gregoir and Lenglart 2000), to derive stylized facts of stock returns (Bulla and Bulla 2006; Nystrup, Madsen, and Lindström 2015), and to model energy prices conditional on market regimes (Langrock, Adam, Leos-Barajas, Mews, Miller, and Papastamatiou 2018; Adam, Langrock, and Kneib 2019b; Adam, Mayr, and Kneib 2022), to name but a few examples. Lihn (2017) used HMMs to model volatility in the Standard and Poor's 500 index to investigate the conjecture that stock returns exhibit negative correlation with volatility. Nguyen (2018) used HMMs to predict closing prices to derive an optimal trading strategy, which was shown to outperform the conventional buy-and-hold strategy, whereas Bulla, Mergner, Bulla, Sesboüe, and Chesneau (2011); Nystrup *et al.* (2015); Nystrup, Madsen, and Lindström (2018) have shown that HMMs prove useful in asset allocation and portfolio optimization applications. All these examples demonstrate that HMMs constitute a versatile class of statistical models for time series that naturally accounts for the state-switching patterns often found in financial data.

In R (Team 2021), various implementations of HMMs are available. For general purposes, the packages **hmm** (Himmelmann 2010), **depmixS4** (Visser and Speekenbrink 2010), and **msm** (Jackson 2011) are frequently used. In addition, a wide range of special-purpose packages is available, for example **moveHMM** (Michelot, Langrock, and Patterson 2016) and **momentuHMM** (McClintock and Michelot 2018) for modeling ecological time series, **hsmm** (Bulla, Bulla, and Nenadić 2010) and **mhsmm** (O'Connell and Højsgaard 2011) for hidden semi-Markov models, **hmm.discnp** (Turner and Liu 2014) and **countHMM** (Adam 2019) for modeling count data, and **LMest** (Bartolucci, Pandolfi, and Pennoni 2017) for modeling longitudinal data. In Python, the library **hmmlearn** (Lebedex 2022) can be used. Yet, an R package that is tailored to financial applications is still lacking.

In this paper, we introduce the R package **fHMM** (Oelschläger, Adam, and Michels 2022), which aims at complementing the above mentioned collection of implementations by making HMMs accessible to R users with an interest in financial time series. The package functionality can be classified into functions for data preparation, model estimation, and model evaluation, which are illustrated in the flowchart displayed in Figure 1. Functions for data preparation include a convenient interface to Yahoo Finance (https://www.finance.yahoo.com) that allows users to download stock market data. The model is estimated in a maximum likelihood framework, where the likelihood is evaluated using the forward algorithm, which is implemented in C++ and parallelized for fast and efficient computation. Functions for model evaluation include pseudo-residual analyses and the computation of model selection criteria. In addition to basic HMMs, the package also implements hierarchical HMMs (HHMMs). HHMMs (Oelschläger and Adam 2021) constitute an extension that improves the model's capability of distinguishing between short- and long-term trends in the data and allows us to jointly model

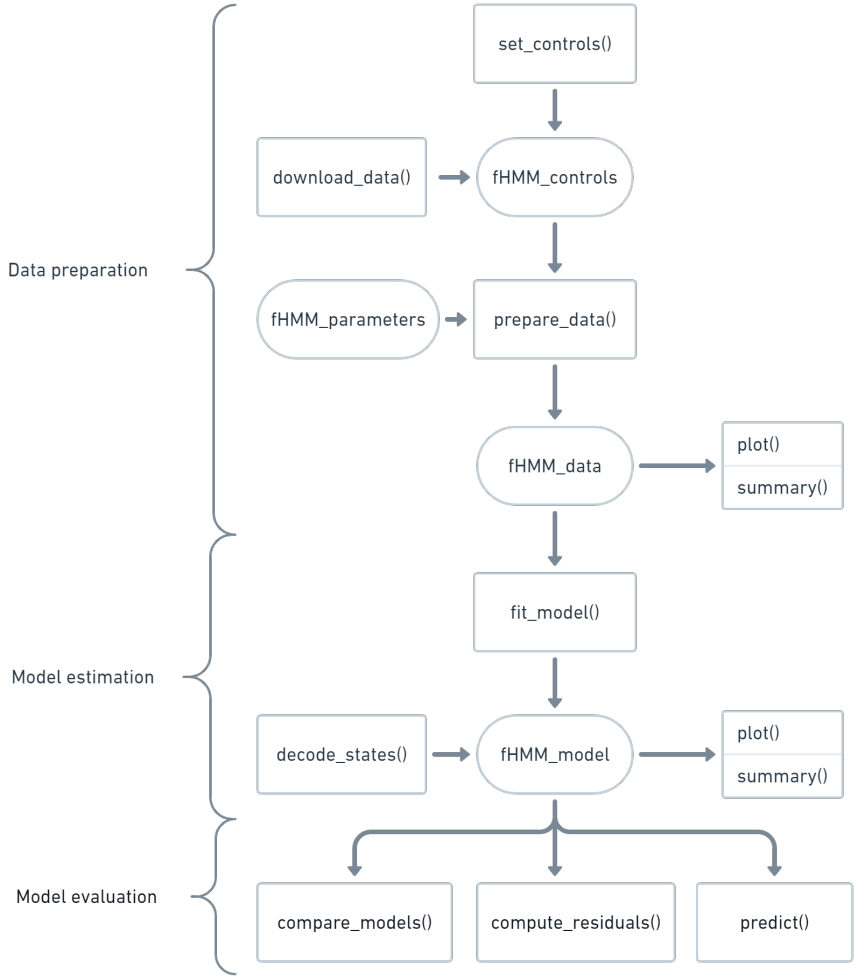



Figure 1: Flowchart. The main functions of the **fHMM** package are visualized using rectangles, while objects are illustrated as ovals.

multiple data streams that were collected at different temporal resolutions, such as monthly trade volumes and daily stock returns (Adam and Oelschläger 2020).

The paper is structured as follows: In Section 2, we introduce HMMs and HHMMs, where we focus on their dependence structure and the underlying model assumptions. In Sections 3–8, we illustrate a typical workflow using the **fHMM** package, where we explain how to specify the model, how to download, prepare, and simulate data, how to fit the model, how to decode the hidden states, how to use a fitted model for state forecasting, how to check the goodness of fit, and how to perform model selection. Each section begins with some theoretical background, which is followed by illustrating examples using stock market data from the Deutscher Aktienindex (DAX) and the Volkswagen Group (VW) as well as simulated data. Each of these sections is complemented by code chunks, which cannot only be used to replicate the examples given in this paper but also serve as a starting point for R users with an interest in financial time series who want to apply HMMs to their own data. Section 9 concludes and gives an outlook of anticipated, future extensions of the **fHMM** package.

## 2. Model definition

Hidden Markov models (HMMs) constitute a modeling framework for time series where a sequence of observations is assumed to depend on a hidden state process. The peculiarity is that, instead of the observation process, the state process cannot be directly observed. However, the hidden states comprise information about the environment the model is applied on. The hidden state process and the observed state-dependent process are connected as follows: we assume that for each point in time $t = 1, \dots, T$, an underlying process $(S_t)_{t=1,\dots,T}$ is in one of $N$ possible states. Then, depending on the active state $S_t \in \{1, \dots, N\}$, the observation $X_t$ from the state-dependent process $(X_t)_{t=1,\dots,T}$ is assumed to be generated by the corresponding state-dependent distribution $f^{(S_t)}$. We assume $(S_t)_t$ to be Markovian, i.e. the active state at time $t$ only depends on the previous state at time $t-1$. Henceforth, the state process is identified by its initial distribution $\delta = (\delta_i), \delta_i = \Pr(S_1 = i), i = 1, \dots, N$, and its transition probability matrix (t.p.m.) $\Gamma = (\gamma_{i,j}), \gamma_{ij} = \Pr(S_t = j | S_{t-1} = i), i, j = 1, \dots, N, t = 2, \dots, T$. Furthermore, $(X_t)_{t=1,\dots,T}$ satisfies the conditional independence assumption, i.e. the observation $X_t$ depends on the current state $S_t$, but is independent from previous observations or states.

When modeling financial time series, the different states can serve as proxies for the current market situation, e.g. calm and nervous. Even though these moods cannot be directly observed, price changes or trading volumes (which can be assumed to depend on the current mood of the market) are observable. Thereby, using an underlying Markov process, we can detect which mood is active at any point in time and how the different moods alternate. Depending on the current mood, a price change is generated by a different distribution. These distributions characterize the moods in terms of expected return and volatility. For example, we can explicitly model price changes at time $t$ by different normal distributions whose mean and variance depend on the current state, $S_t$.

Following Zucchini *et al.* (2016), we assume that the initial distribution $\delta$ equals the stationary distribution $\pi$, where $\pi = \pi\Gamma$, i.e. the stationary and henceforth the initial distribution is determined by $\Gamma$.[1] This is reasonable from a practical point of view: On the one hand, the hidden state process has been evolving for some time before we start to observe it and hence can be assumed to be stationary. On the other hand, setting $\delta = \pi$ reduces the number of parameters that need to be estimated, which is convenient from a computational perspective.

HHMMs constitute a flexible extension of basic HMMs that can be used to jointly model multiple data observed on two different time scales (Oelschläger and Adam 2021). The two time series, one on a coarser and one on a finer scale, differ in the number of observations, e.g. monthly observations on the coarser scale and daily observations on the finer scale. Following the concept of HMMs, we can model both state-dependent time series jointly. First, we treat the time series on the coarser scale as stemming from an ordinary HMM, which we refer to as the coarse-scale HMM: At each time point $t$ of the coarse-scale time space $\{1, \dots, T\}$, an underlying process $(S_t)_t$ selects one state from the coarse-scale state space $\{1, \dots, N\}$. We refer to $(S_t)_t$ as the hidden coarse-scale state process. Depending on which state is active at time $t$, one of $N$ possible distributions $f^{(1)}, \dots, f^{(N)}$ realizes the observation $X_t$. The

[1] A note on the existence of a stationary distribution: If the Markov process is irreducible, it has a unique distribution, which is the solution to the equation system $\pi = \pi\Gamma$. If, additionally, the Markov process is aperiodic, its state distribution converges to the stationary distribution, see Norris (1997). Irreducibility and aperiodicity are usually satisfied in practice.

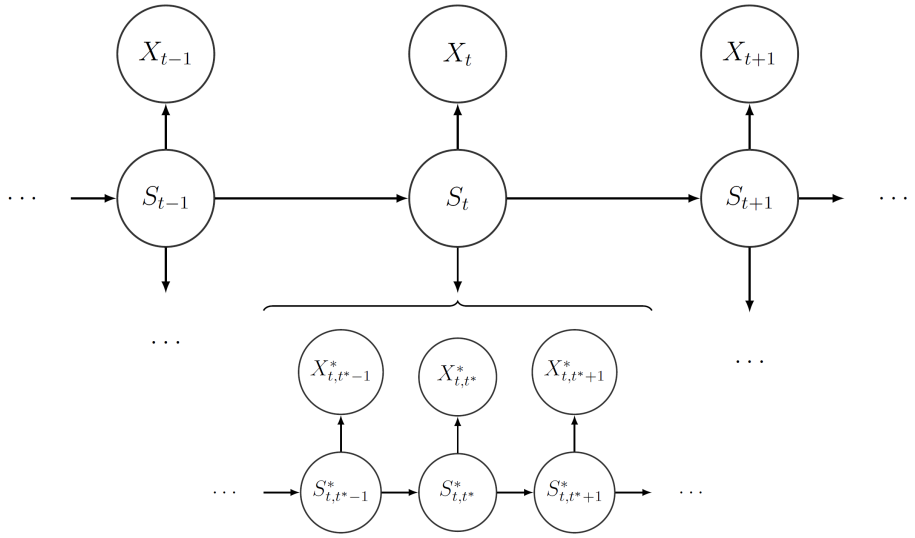



Figure 2: Dependence structure of an HHMM. The dependence structure of a basic HMM is visualized in the upper part.

process $(X_t)_t$ is called the observed coarse-scale state-dependent process. The processes $(S_t)_t$ and $(X_t)_t$ have the same properties as before, namely $(S_t)_t$ is a first-order Markov process and $(X_t)_t$ satisfies the conditional independence assumption. This dependence structure is visualized in the upper part of Figure 2.

For the fine-scale time series, the observed data is split into $T$ distinct chunks, each of them having a correspondence to the $t$-th coarse scale time point. So, the hierarchical structure arises naturally: for each chunk, we model the containing data points via the fine-scale HHM, which is selected by the hidden coarse-scale state $S_t$. Thus, each fine-scale HMM consists of two stochastic processes: the hidden, fine-scale process $(S^*_{t,t^*})_{t^*}$ selecting states from $\{1, \ldots, N^*\}$, the fine-scale state space, for each time point $t^*$ in $\{1, \ldots, T^*\}$, the fine-scale time space and the observed, fine-scale process $(X^*_{t,t^*})_{t^*}$ whose observations are assumed to depend on one of $N^*$ possible distributions $f^{*(i,1)}, \ldots, f^{*(i,N^*)}$, chosen depending on the actual state of the hidden, fine-scale process. By construction, each fine-scale HMM contains of an own t.p.m. $\Gamma^{*(i)}$, initial distribution $\delta^{*(i)}$, stationary distribution $\pi^{*(i)}$, and state-dependent distributions $f^{*(i,1)}, \ldots, f^{*(i,N^*)}$. Similar to the coarse-scale HMM, the hidden, fine-scale process is Markovian and satisfies the conditional independence assumption. In contrast, the observed, fine-scale process has exclusive dependence on the actual state of the hidden, fine-scale process. In the lower part of Figure 2, the dependence and hierarchical structure is visualized.

## 3. Model specification

In the **fHMM** package, models are specified by a named list of controls that is passed to the `set_controls()` function. This usually constitutes the first step when using the package, see Figure 1. The function checks the specifications and returns an '`fHMM_controls`' object, which

stores all settings and thereby provides the information required for other functionalities. In the following, we demonstrate three example specifications that should help the user to tailor an HMM to their need. The examples are continued in the following sections. All possible specifications are documented in detail on the function's help page, which can be accessed from the R console via `help(set_controls, package = "fHMM")`.

**Example 1: DAX.** We fit a 3-state HMM to the closing prices of the DAX (Janssen and Rudolph 1992). Assume that the data are available in the working directory as file `"dax.csv"`. Such data can be obtained directly from Yahoo Finance via the convenience function `download_data()`, see Section 4.

The following code chunk sets the number `states = 3` of hidden states. Any number greater than or equal to 2 is possible. Next, `sdds = "t"` specifies state-dependent t-distributions, which provide a popular choice for modeling log-returns (Platen and Rendek 2008). Alternatively, `sdds = "gamma"` specifies Gamma distributions, which are useful for model trading volumes, see Adam and Oelschläger (2020). The `data` entry sets the path to the data file (`file = "dax.csv"`), the file's column that contains the dates (`date_column = "Date"`), and the data (`date_column = "Close"`). The specification `logreturns = TRUE` transforms the data to log-returns.

```
R> contr_dax <- list(
+    states = 3,
+    sdds   = "t",
+    data   = list(file        = "dax.csv",
+                  date_column = "Date",
+                  data_column = "Close",
+                  logreturns  = TRUE)
+  )
```

Passing this list to the `set_controls()` function returns an object of class '`fHMM_controls`', which contains the specifications and default settings.

```
R> contr_dax <- set_controls(contr_dax)
R> class(contr_dax)

[1] "fHMM_controls"
```

**Example 2: Simulation.** If the `data` entry is not specified, data is simulated according to the model specification. Simulation typically serves to assess the properties of estimation algorithms either for research or in a bootstrap-like fashion, as can be seen for example in Oelschläger (2019). The following code chunk specifies a 2-state HMM with state-dependent Gamma distributions, where the expected values for state 1 and 2 are fixed to `1` and `2`, respectively. The model is fitted to 200 data points (`horizon = 200`) simulated according to this specification based on `runs = 50` randomly initialized numerical optimization runs of the model's likelihood function. Printing the '`fHMM_controls`' object summarizes the model specification:

```
R> contr_sim <- list(
+    states  = 2,
+    sdds    = "gamma(mu = 1|2)",
+    horizon = 200,
+    fit     = list(runs = 50)
+  )
R> (contr_sim <- set_controls(contr_sim))

fHMM controls:
* hierarchy: FALSE
* data type: simulated
* number of states: 2
* sdds: gamma(mu = 1|2)
* number of runs: 50
```

**Example 3: Multiple data streams.** An HHMM can be specified by adding `hierarchy = TRUE` to the controls. The following is a specification for the DAX data on the fine scale and the VW data on the coarse scale (both data sets are contained in the **fHMM** package):

```
R> contr_hhmm <- list(
+    hierarchy = TRUE,
+    states    = c(2,2),
+    sdds      = c("t(df = 1)", "t(df = 1)"),
+    period    = "m",
+    data      = list(file = c(system.file("extdata", "dax.csv",
+                              package = "fHMM"),
+                              system.file("extdata", "vw.csv",
+                              package = "fHMM")),
+                     date_column = c("Date","Date"),
+                     data_column = c("Close","Close"),
+                     from = "2015-01-01",
+                     to = "2020-01-01",
+                     logreturns = c(TRUE,TRUE),
+                     merge = function(x) mean(x))
+  )
R> contr_hhmm <- set_controls(contr_hhmm)
```

The line `states = c(2, 2)` specifies 2 coarse-scale and 2 fine-scale states, respectively. State-dependent t-distributions are used, where the degrees of freedom are fixed to `1` on both time scales (setting `df = Inf` is possible and results in state-dependent normal distributions). Via `period = "m"`, we specify a monthly fine-scale time horizon. Alternatives are `"w"`, `"q"`, and `"y"` for weekly, quarterly, and yearly time horizons, respectively. The observation period is restricted to five years via `from = "2015-01-01"` and `to = "2020-01-01"`. With `logreturns = c(TRUE,TRUE)`, we ensure that the data on both time scales are transformed to log-returns. If the coarse-scale data have a finer temporal resolution than defined by `period`, the data can be merged by specifying a function via the `merge` argument. In this example, the file

`"dax.vw"` contains daily closing prices. Since we specified `merge = function(x) mean(x)`, the monthly average closing prices are used as coarse-scale observations.

## 4. Data management

Empirical data for modeling must be provided as a comma-separated values (.csv) file. Its path must be specified in `set_controls()`, see the previous section. The package includes the convenience function `download_data()` for downloading daily stock market data directly from Yahoo Finance in the required format. The function call is `download_data(symbol, from, to, file)`, where

- `symbol` is the stock's symbol that has to match the official symbol on Yahoo Finance,

- `from` and `to` define the desired time interval (in the format `"YYYY-MM-DD"`),

- `file` is the name of the saved file. Per default, it is saved in the current working directory under the name `<symbol>.csv`.

For example, the 21st century data of the DAX can be downloaded via the following line:

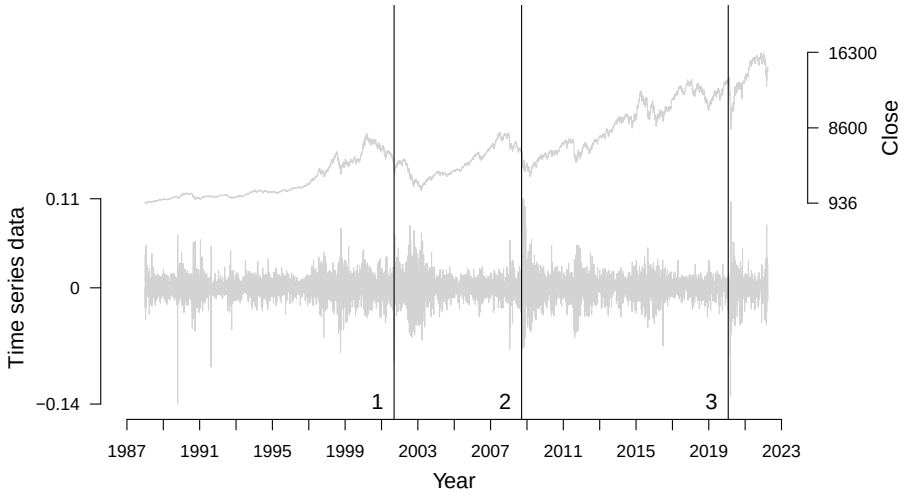
```
R> download_data(symbol = "^GDAXI", from = "2001-01-01", to = Sys.Date())
```

**Example 1: DAX (cont.).** Recall the control specification for the 3-state HMM DAX model from the previous section. The `prepare_data()` function prepares the data based on the specifications and returns an '`fHMM_data`' object. This object can then be passed to the `fit_model()` function for model fitting in the next step, see Section 5. The `summary()` method provides an overview of the data.

```
R> data_dax <- prepare_data(contr_dax)
R> summary(data_dax)

Summary of fHMM empirical data
* number of observations: 8823
* data source: dax.csv
* date column: Date
* log returns: TRUE
```

In addition, the data can be visualized via the `plot()` method. To facilitate interpretation, historical events with a potential influence on the time series can be highlighted as follows:

```
R> events <- fHMM_events(
+    list(
+      dates = c("2001-09-11", "2008-09-15", "2020-01-27"),
+      labels = c("9/11 terrorist attack", "Bankruptcy of Lehman Brothers",
+                 "First COVID-19 case in Germany")
+      )
+    )
R> plot(data_dax, events = events)
```



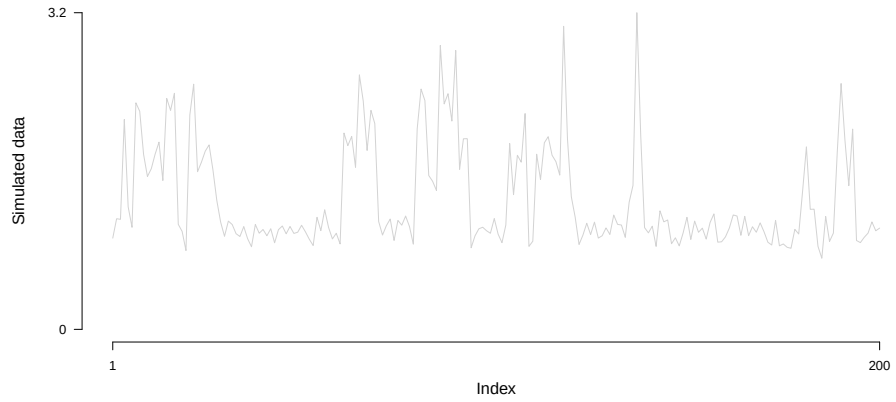

1: 9/11 terrorist attack 2: Bankruptcy of Lehman Brothers 3: First COVID−19 case in Germany

**Example 2: Simulation (cont.).** As mentioned in the previous section, if the `data` argument in the model's controls is unspecified, data is simulated according to the model specification. True model parameters can be specified by defining an '`fHMM_parameters`'-object via the `fHMM_parameters()` function and passing it to `prepare_data()`, for example:

```
R> pars <- fHMM_parameters(
+    controls = contr_sim,
+    Gamma = matrix(c(0.9,0.2,0.1,0.8), nrow = 2),
+    sigmas = c(0.1,0.5)
+  )
R> data_sim <- prepare_data(contr_sim, true_parameters = pars, seed = 1)
```

The visualization of the simulated time series shows the state persistence (induced by $\Gamma_{11} = 0.9$ and $\Gamma_{22} = 0.8$) and the different standard deviations ($\sigma_1 = 0.1$ and $\sigma = 0.5$) of the state-dependent Gamma distributions. Remember that the expected values were set to $\mu_1 = 1$ and $\mu_2 = 2$.

```
R> plot(data_sim)
```




**Example 3: Multiple data streams (cont.).** Data preparation for the HHMM can be done analogously:

```
R> data_hhmm <- prepare_data(contr_hhmm)
```

## 5. Model estimation

The **fHMM** package implements the maximum likelihood method for model estimation. In the following, the likelihood function of an HHMM is derived, the non-hierarchical case can be deduced. We also discuss challenges related to the numerical maximization and subsequently estimate the three running example models.

An HHMM can be treated as an HMM with two conditionally independent data streams; the coarse-scale observations on the one hand and the corresponding chunk of fine-scale observations connected to a fine-scale HMM on the other hand. To derive the likelihood of an HHMM, we start by computing the likelihood of each chunk of fine-scale observations being generated by each fine-scale HMM.

To fit the $i$-th fine-scale HMM, with model parameters $\theta^{*(i)} = (\delta^{*(i)}, \Gamma^{*(i)}, (f^{*(i,k)})_k)$ to the $t$-th chunk of fine-scale observations, which is denoted by $(X_{t,t^*})_{t^*}$, we consider the fine-scale forward probabilities

$$\alpha^{*(i)}_{k,t^*} = f^{*(i)}(X^*_{t,1}, \dots, X^*_{t,t^*}, S^*_{t,t^*} = k),$$

where $t^* = 1, \dots, T^*$ and $k = 1, \dots, N^*$. Using the fine-scale forward probabilities, the fine-scale likelihoods can be obtained from the law of total probability as

$$\mathcal{L}^{\text{HMM}}(\theta^{*(i)} \mid (X^*_{t,t^*})_{t^*}) = \sum_{k=1}^{N^*} \alpha^{*(i)}_{k,T^*}.$$

The forward probabilities can be calculated recursively as

$$\begin{aligned}
\alpha^{*(i)}_{k,1} &= \delta^{*(i)}_k f^{*(i,k)}(X^*_{t,1}), \\
\alpha^{*(i)}_{k,t^*} &= f^{*(i,k)}(X^*_{t,t^*}) \sum_{j=1}^{N^*} \gamma^{*(i)}_{jk} \alpha^{*(i)}_{j,t^*-1}, \quad t^* = 2, \dots, T^*.
\end{aligned}$$

The transition from the likelihood function of an HMM to the likelihood function of an HHMM is straightforward: Consider the coarse-scale forward probabilities

$$\alpha_{i,t} = f(X_1, \ldots, X_t, (X^*_{1,t^*})_{t^*}, \ldots, (X^*_{t,t^*})_{t^*}, S_t = i),$$

where $t = 1, \ldots, T$ and $i = 1, \ldots, N$. The likelihood function of the HHMM results as

$$\mathcal{L}^{\text{HHMM}}(\theta, (\theta^{*(i)})_i \mid (X_t)_t, ((X^*_{t,t^*})_{t^*})_t) = \sum_{i=1}^{N} \alpha_{i,T}.$$

The coarse-scale forward probabilities can be calculated similarly:

$$\begin{aligned}
\alpha_{i,1} &= \delta_i \mathcal{L}^{\text{HMM}}(\theta^{*(i)} \mid (X^*_{1,t^*})_{t^*}) f^{(i)}(X_1), \\
\alpha_{i,t} &= \mathcal{L}^{\text{HMM}}(\theta^{*(i)} \mid (X^*_{t,t^*})_{t^*}) f^{(i)}(X_t) \sum_{j=1}^{N} \gamma_{ji} \alpha_{j,t-1}, \quad t = 2, \ldots, T.
\end{aligned}$$

To account for parameter constraints associated with the transition probabilities (and potentially the parameters of the state-dependent distributions), we use parameter transformations. To ensure that the entries of the t.p.m.s fulfill non-negativity and the unity condition, we estimate unconstrained values $(\eta_{ij})_{i \neq j}$ for the non-diagonal entries of $\Gamma$ and derive the probabilities using the multinomial logit link

$$\gamma_{ij} = \frac{\exp(\eta_{ij})}{1 + \sum_{k \neq i} \exp(\eta_{ik})}, \; i \neq j$$

rather than estimating the probabilities $(\gamma_{ij})_{i,j}$ directly. The diagonal entries result from the unity condition as

$$\gamma_{ii} = 1 - \sum_{j \neq i} \gamma_{ij}.$$

Furthermore, variances are strictly positive, which can be achieved by applying an exponential transformation to the unconstrained estimator.

Two more technical difficulties arise: First, we often face numerical under- or overflow, which can be addressed by maximizing the logarithm of the likelihood and incorporating constants in a conducive way, see Oelschläger and Adam (2021) for the details. Second, as the likelihood is maximized with respect to a relatively large number of parameters, the obtained maximum can be a local rather than the global one. To avoid this problem, it is recommended to run the maximization multiple times from different, possibly randomly selected starting points, and to choose the model that corresponds to the highest likelihood (Zucchini *et al.* 2016). For efficient initialization, **fHMM** uses the first and second data moments as a basis for the initial guesses.

**Example 1: DAX (cont.).** In Section 4, we defined the '`fHMM_data`' object `data_dax`. This object can be directly passed to the `fit_model()` function that numerically maximizes the model's (log-) likelihood function:[2]

[2]The numerical maximization runs can be parallelized by setting the function's `ncluster` argument to a value greater than 1.

```
R> dax_model_3t <- fit_model(data_dax)
```

The estimated model is saved in the **fHMM** package and can be accessed via:

```
R> data(dax_model_3t, package = "fHMM")
```

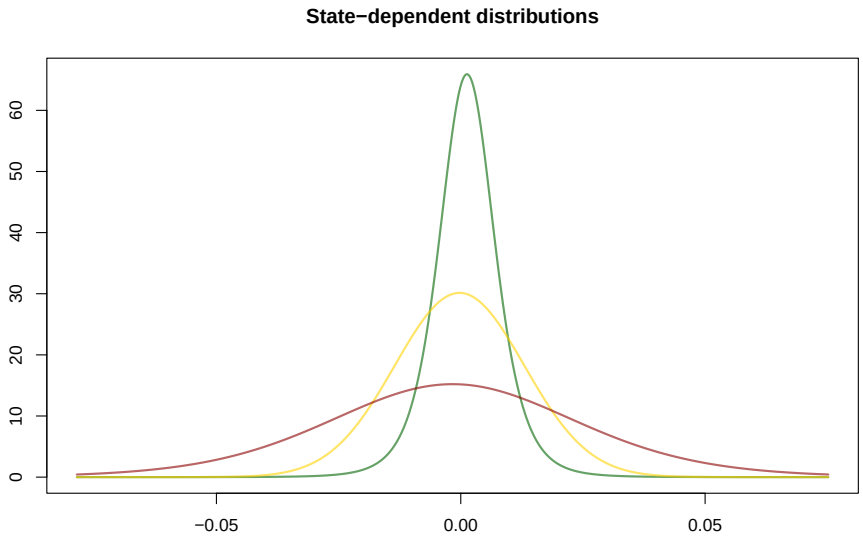
The `coef()` method returns a data frame of the estimated model coefficients along with 1-`alpha` confidence intervals (`alpha = 0.05` being the default) obtained via the inverse Fisher information (`lb` stands for lower- and `ub` upper-bound of the intervals, respectively):

```
R> coef(dax_model_3t, alpha = 0.05)
```

```
                     lb      estimate           ub
Gamma_2.1  1.393745e-02  2.170056e-02 3.357222e-02
Gamma_3.1  1.710420e-06  1.696596e-06 1.671225e-06
Gamma_1.2  1.659146e-02  2.641008e-02 4.179228e-02
Gamma_3.2  9.359401e-03  1.740175e-02 3.213059e-02
Gamma_1.3  1.258535e-08  1.246159e-08 1.226657e-08
Gamma_2.3  2.789877e-03  5.145960e-03 9.431251e-03
mu_1       9.610235e-04  1.268712e-03 1.576400e-03
mu_2      -7.955275e-04 -2.517499e-04 2.920278e-04
mu_3      -3.723796e-03 -1.673649e-03 3.764982e-04
sigma_1    5.332119e-03  5.774343e-03 6.253244e-03
sigma_2    1.269075e-02  1.323801e-02 1.380887e-02
sigma_3    2.329677e-02  2.562622e-02 2.818860e-02
df_1       3.964154e+00  5.272157e+00 7.011747e+00
df_2       7.592272e+04  7.592547e+04 7.592821e+04
df_3       5.501298e+00  1.064107e+01 2.058285e+01
```

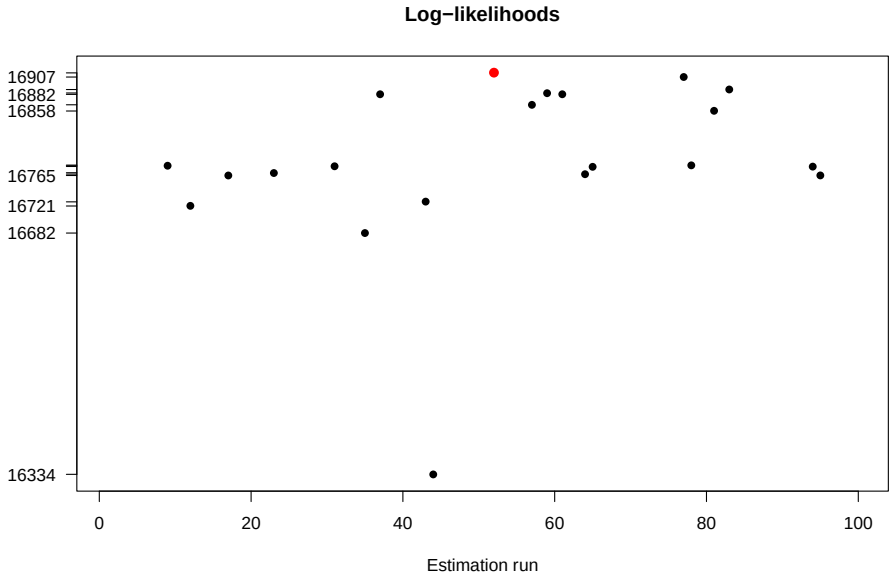
Adding `plot_type = "sdds"` to the `plot()` method visualizes the estimated state-dependent distributions:

```
R> plot(dax_model_3t, plot_type = "sdds")
```




As mentioned above, the HMM likelihood function is prone to local optima. This effect can be visualized by plotting the log-likelihood values in the different optimization runs, where the best run is marked in red:

```
R> plot(dax_model_3t, plot_type = "ll")
```




**Example 2: Simulation (cont.).** Fitting an HMM to the simulated data is analogue via the `fit_model()` function. The estimated model `sim_model_2gamma` is also saved in **fHMM**:

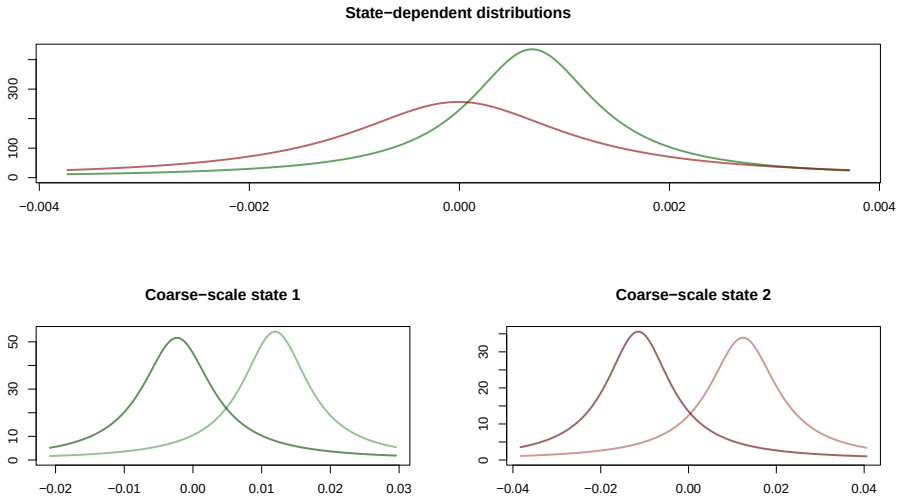
```
R> data(sim_model_2gamma, package = "fHMM")
```

The `summary()` method gives an overview of the estimated model. In the simulated case, we can compare the estimates to the true model coefficients:

```
R> summary(sim_model_2gamma)

Summary of fHMM model

  simulated hierarchy        LL      AIC      BIC
1      TRUE     FALSE -192.9676 393.9353 407.1286

State-dependent distributions:
gamma(mu = 1|2)

Estimates:
               lb estimate     ub true
Gamma_2.1 0.06475  0.17804 0.4039  0.2
Gamma_1.2 0.03052  0.07781 0.1844  0.1
sigma_1   0.44079  0.49962 0.5663  0.5
sigma_2   0.70108  0.92279 1.2146  1.0
```

**Example 3: Multiple data streams (cont.).** In the hierarchical case, we can visualize the estimated state-dependent distributions on both the coarse-scale and the fine-scale layer:

```
R> # dax_vw_model <- fit_model(data_hhmm)
R> data(dax_vw_model, package = "fHMM")
R> plot(dax_vw_model, plot_type = "sdds")
```




# 6. State decoding and prediciton

For financial markets, it is of special interest to infer the underlying (hidden) states in order to gain insight about the actual market situation and for prediction. The Viterbi algorithm (Forney 1973) is a recursive scheme that enables to find the most likely trajectory of hidden states under the estimated HMM. To this end, we follow Zucchini *et al.* (2016) and define

$$\begin{aligned}\zeta_{1i} &= \Pr(S_1 = i, X_1 = x_1) = \delta_i p_i(x_1) \\ \zeta_{ti} &= \max_{s_1,\ldots,s_{t-1}} \Pr(S_{t-1} = s_{t-1}, S_t = i, X_t = x_t)\end{aligned}$$

for $i = 1, \ldots, N$ (the index for the states) and $t = 2, \ldots, T$ (the index of time). Then, the trajectory of most likely states $i_1, \ldots, i_T$ can be calculated recursively backwards from

$$\begin{aligned}i_T &= \operatorname*{argmax}_{i=1,\ldots,N} \zeta_{Ti} \\ i_t &= \operatorname*{argmax}_{i=1,\ldots,N} (\zeta_{ti}\gamma_{i,i_{t+1}}), \quad t = T-1, \ldots, 1.\end{aligned}$$

Transferring the state decoding to HHMMs is straightforward via decoding the coarse-scale states first and afterwards, by using this information, decoding the fine-scale state process, see Adam, Griffiths, Leos-Barajas, Meese, Lowe, Blackwell, Righton, and Langrock (2019a).

In the following, we introduce the `decode_states()` function for state decoding and the `predict()` method for forecasting. As all of the **fHMM** functionalities presented in the remainder of this article are completely analogue for the hierarchical and the simulated case, respectively, we will focus our attention on example 1 and invite the reader to apply the methods to example 2 and 3 on their own.

**Example 1: DAX (cont.).** The underlying states of the 3-state HMM for the DAX can be decoded via the `decode_states()` function, which updates an '`fHMM_model`' object.

```
R> dax_model_3t <- decode_states(dax_model_3t)
```

The state sequence is saved as argument `dax_model_3t$decoding`:

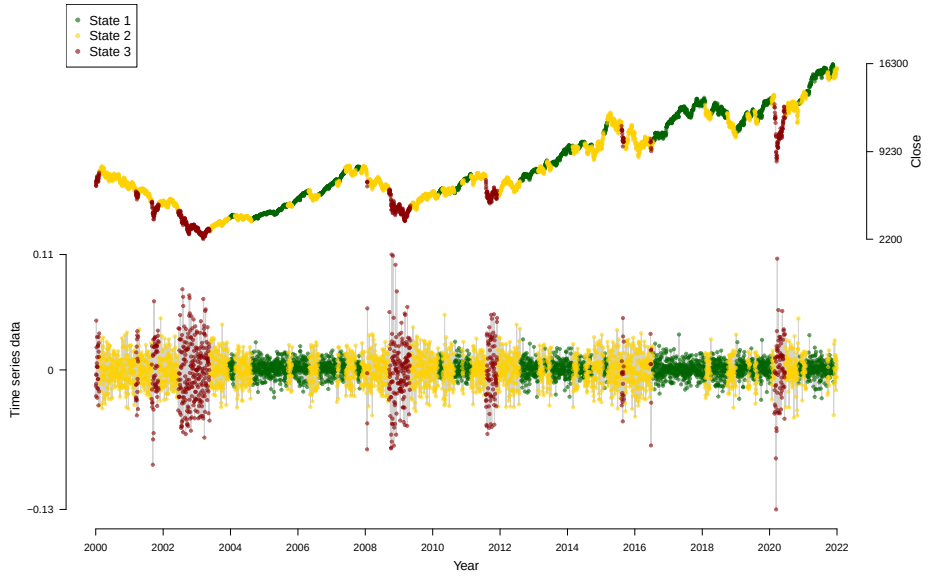
```
R> table(dax_model_3t$decoding)
```

```
   1    2    3
2177 2753  695
```

The decoded time series can then be visualized using the `plot()` method:[3]

```
R> plot(dax_model_3t)
```

[3]Notice that the model is invariant to permutations of the state labels. The **fHMM** package provides the option to switch labels after state decoding via the `reorder_states()` function. For example, `reorder_states(dax_model_3t, 3:1)` reverses the order.




After decoded the underlying states, we use the estimated transition probabilities to compute the state probabilities of the subsequent observations. Based on these probabilities and in combination with the estimated state-dependent distributions, the subsequent observations can be predicted, compare Zucchini *et al.* (2016):

```
R> predict(dax_model_3t, ahead = 10)
```

```
   state_1 state_2 state_3       lb estimate      ub
1  0.02170 0.97315 0.00515 -0.02190 -0.00023 0.02145
2  0.04225 0.94769 0.01006 -0.02179 -0.00020 0.02138
3  0.06170 0.92354 0.01477 -0.02168 -0.00018 0.02132
4  0.08011 0.90063 0.01926 -0.02158 -0.00016 0.02126
5  0.09754 0.87890 0.02356 -0.02148 -0.00014 0.02121
6  0.11403 0.85829 0.02767 -0.02140 -0.00012 0.02116
7  0.12965 0.83874 0.03161 -0.02131 -0.00010 0.02111
8  0.14442 0.82020 0.03537 -0.02124 -0.00008 0.02107
9  0.15841 0.80261 0.03898 -0.02116 -0.00007 0.02103
10 0.17164 0.78593 0.04243 -0.02110 -0.00005 0.02100
```

Columns 1 to 3 show the state probabilities for the next `ahead = 10` trading days. The values in columns 4 to 6 (the bounds of the 95%-confidence intervals and the point predictions) are log-returns, which obviously can be transformed to index values or relative returns.

## 7. Model checking

Checking whether a fitted model describes the data well is an essential part of any modeling process. In the HMM setting, this is typically done by analyzing the so-called pseudo-residuals (Zucchini *et al.* 2016). Since the observations are modeled by different distributions

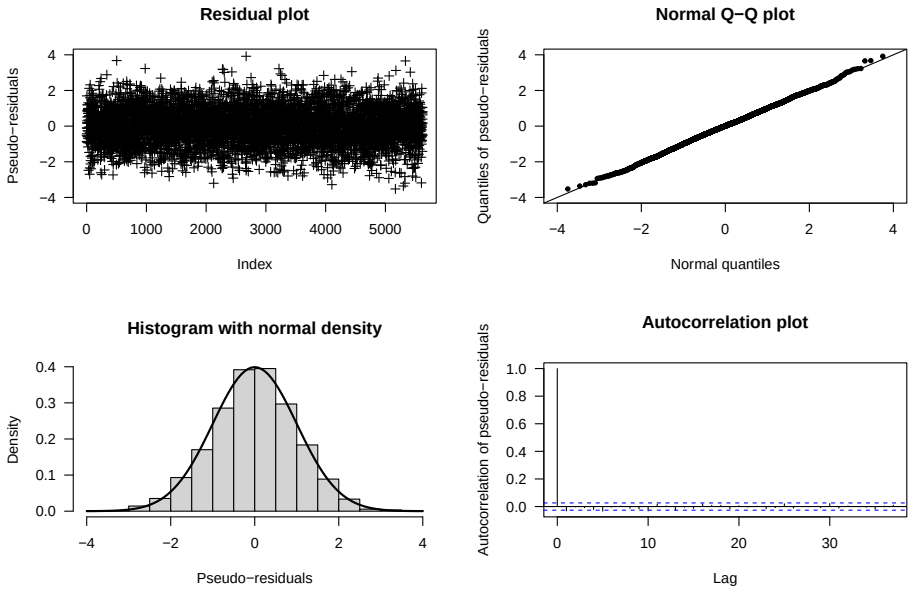
(depending on the active state), they have to be transformed on a common scale, which can be done as follows: If the observation $X_t$ has the invertible distribution function $F_{X_t}$, then $Z_t = \Phi^{-1}(F_{X_t}(X_t))$ is standard normally distributed, where $\Phi$ denotes the cumulative distribution function of the standard normal distribution. The observations are modeled well if the pseudo-residuals $(Z_t)_t$ are approximately standard normally distributed and exhibit no autocorrelation. In the hierarchical case, we first decode the coarse-scale state process using the Viterbi algorithm. Subsequently, we assign each coarse-scale observation its distribution function under the fitted model and perform the transformation described above. Using the Viterbi-decoded coarse-scale states, we then treat the fine-scale observations analogously.

**Example 1: DAX (cont.).** Via the `compute_residuals()` function, pseudo-residuals can be computed in **fHMM** (provided that the states have been decoded beforehand). The function updates the `dax_model_3t` object in the following line:

```
R> dax_model_3t <- compute_residuals(dax_model_3t)
```

The normality and independence of the pseudo-residuals can be verified visually:

```
R> plot(dax_model_3t, plot_type = "pr")
```




Alternatively, the residuals can be extracted from the model object for normality tests, for example a Jarque-Bera test (Jarque and Bera 1987):[4]

```
R> res <- dax_model_3t$residuals
R> tseries::jarque.bera.test(res)
```

[4]Here, the test is unable to reject the null hypothesis that the data is normally distributed.

```
        Jarque Bera Test

data:  res
X-squared = 2.6542, df = 2, p-value = 0.2652
```

## 8. Model selection

Model selection involves the choice of a family for the state-dependent distributions and the selection of the number of states. Common model selection tools are information criteria, such as the Akaike information criterion (AIC) (Akaike 1974) or the Bayesian information criterion (BIC) (Schwarz 1978). They are defined as

$$\begin{aligned} \text{AIC} &= -2 \log \mathcal{L}^{(\text{H})\text{HMM}} + 2p; \\ \text{BIC} &= -2 \log \mathcal{L}^{(\text{H})\text{HMM}} + \log(T)p, \end{aligned}$$

where $p$ denotes the number of model parameters and $T$ is the number of observations. Both criteria aim at finding a compromise between model fit and model complexity, where a model with a lower value is to be preferred. For an in-depth discussion of pitfalls, practical challenges, and pragmatic solutions regarding model selection, we refer to Pohle, Langrock, van Beest, and Schmidt (2017). The **fHMM** package provides the `compare_models()` function that takes (arbitrarily many) '`fHMM_model`' objects as input and returns a matrix of the number of parameters, the log-likelihood value, the AIC, and the BIC.

**Example 1: DAX (cont.).** We compare our 3-state HMM with state-dependent t-distributions fitted to the Dax data to an HMM with 2 states and normal state-dependent distributions. The competing model was estimated using the same data and can be accessed as object `dax_model_2n`. In this example, both AIC and BIC clearly prefer the more complex model:

```
R> data(dax_model_2n, package = "fHMM")
R> compare_models(dax_model_2n, dax_model_3t)

             parameters log-likelihood       AIC       BIC
dax_model_2n          6       16681.98 -33351.96 -33312.15
dax_model_3t         15       16913.33 -33796.65 -33697.13
```

## 9. Conclusions

The **fHMM** package aims at making HMMs accessible to R users with an interest in financial time series. It contains functions to download, prepare, and simulate data, to fit models, to decode the hidden states, to use a fitted model for state forecasting, to check the goodness of fit, and to perform model selection. The **fHMM** package has a user-friendly design: All model specifications are centralized in a list of controls, four different package objects can be seamlessly passed between functions, and its usage follows a clear workflow (see Figure 1). In this paper, we illustrated a typical workflow using three illustrating examples (an application

to stock market data from the DAX and VW as well as a model fitted to simulated data) that serve as a starting point for R users who want to apply HMMs and their extensions to their own data.

Current limitations of the **fHMM** package include (1) the confined set of available state-dependent distributions for the observations, (2) the restriction to a discrete state space, (3) the restriction to a discretized time dimension, (4) the limitation of a maximum number of two hierarchies in HHMMs, and (5) the need to specify the number of hidden states in advance of the model estimation. We plan to overcome these limitations and invite the community to suggest further features that we can implement in future package versions.

## Computational details

The results presented in this paper were obtained using R 4.1.1 with the **fHMM** 1.0.2 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/.

## References

Adam T (2019). *countHMM: penalized estimation of flexible hidden Markov models for time series of counts*. R package, version 0.1.0, URL https://CRAN.R-project.org/package=countHMM.

Adam T, Griffiths C, Leos-Barajas V, Meese E, Lowe C, Blackwell P, Righton D, Langrock R (2019a). "Joint modelling of multi-scale animal movement data using hierarchical hidden Markov models." *Methods in Ecology and Evolution*, **10**(9), 1536–1550.

Adam T, Langrock R, Kneib T (2019b). "Model-based clustering of time series data: a flexible approach using nonparametric state-switching quantile regression models." *Book of Short Papers of the 12th Scientific Meeting on Classification and Data Analysis*, pp. 8–11.

Adam T, Mayr A, Kneib T (2022). "Gradient boosting in Markov-switching generalized additive models for location, scale, and shape." *Econometrics and Statistics*, **22**, 3–16.

Adam T, Oelschläger L (2020). "Hidden Markov models for multi-scale time series: an application to stock market data." *Proceedings of the 35th International Workshop on Statistical Modelling*, **1**, 2–7.

Akaike H (1974). "A new look at the statistical model identification." **19**.

Bartolucci F, Pandolfi S, Pennoni F (2017). "LMest: an R package for latent Markov models for longitudinal categorical data." *Journal of Statistical Software*, **81**, 1–38.

Bulla J, Bulla I (2006). "Stylized facts of financial time series and hidden semi-Markov models." *Computational Statistics and Data Analysis*, **51**(4), 2192–2209. doi:10.1016/j.csda.2006.07.021.

Bulla J, Bulla I, Nenadić O (2010). "hsmm – An R package for analyzing hidden semi-Markov models." *Computational Statistics and Data Analysis*, **54**(3), 611–619.

Bulla J, Mergner S, Bulla I, Sesboüe A, Chesneau C (2011). "Markov-switching asset allocation: do profitable strategies exist?" *Journal of Asset Management*, **12**, 310–321. doi:10.1057/jam.2010.27.

Forney G (1973). "The viterbi algorithm." *Proceedings of the IEEE*, **61**(3), 268–278.

Gregoir S, Lenglart F (2000). "Measuring the probability of a business cycle turning point by using a multivariate qualitative hidden Markov model." *Journal of forecasting*, **19**(2), 81–102.

Himmelmann L (2010). *HMM – hidden Markov models*. R package, version 1.0, URL http://CRAN.R-project.org/package=HMM.

Jackson C (2011). "Multi-state models for panel data: the msm package for R." *Journal of Statistical Software*, **38**(8), 1–28. doi:10.18637/jss.v038.i08.

Janssen B, Rudolph B (1992). "Der Deutsche Aktienindex DAX." *Knapp Verlag*.

Jarque C, Bera A (1987). "A test for normality of observations and regression residuals." *International Statistical Review*, **55**, 163–172.

Kim CJ, Nelson C (1998). "Business cycle turning points, a new coincident index, and tests of duration dependence based on a dynamic factor model with regime switching." *Review of Economics and Statistics*, **80**(2), 188–201.

Langrock R, Adam T, Leos-Barajas V, Mews S, Miller D, Papastamatiou Y (2018). "Spline-based nonparametric inference in general state-switching models." *Statistica Neerlandica*, **72**(3), 179–200.

Lebedex S (2022). *hmmlearn: hidden Markov models in Python, with scikit-learn like API*. Python library, version 0.2.7, URL https://hmmlearn.readthedocs.io/.

Lihn S (2017). "Hidden Markov model for financial time series and its application to S&P 500 index." *Quantitative Finance (forthcoming)*.

McClintock B, Michelot T (2018). "momentuHMM: R package for generalized hidden Markov models of animal movement." *Methods in Ecology and Evolution*, **9**(6), 1518–1530.

Michelot T, Langrock R, Patterson T (2016). "moveHMM: an R package for the statistical modelling of animal movement data using hidden Markov models." *Methods in Ecology and Evolution*, **7**(11), 1308–1315.

Nguyen N (2018). "Hidden Markov model for stock trading." *International Journal of Financial Studies*, **6**(2).

Norris J (1997). "Markov Chains." *Cambridge University Press*.

Nystrup P, Madsen H, Lindström E (2015). "Stylised facts of financial time series and hidden Markov models in continuous time." *Quantitative Finance*, **15**(9), 1531–1541. doi:10.1080/14697688.2015.1004801.

Nystrup P, Madsen H, Lindström E (2018). "Dynamic portfolio optimization across hidden market regimes." *Quantitative Finance*, **18**(1), 83–95. doi:10.1080/14697688.2017.1342857.

O'Connell J, Højsgaard S (2011). "Hidden semi markov models for multiple observation sequences: the mhsmm package for R." *Journal of Statistical Software*, **39**, 1–22.

Oelschläger L (2019). "Detection of bearish and bullish markets in the DAX using hierarchical hidden Markov models." *Master's thesis at Bielefeld University*.

Oelschläger L, Adam T (2021). "Detecting bearish and bullish markets in financial time series using hierarchical hidden Markov models." *Statistical Modelling*. doi:10.1177/1471082X211034048.

Oelschläger L, Adam T, Michels R (2022). *fHMM: fitting hidden Markov models to financial data*. R package, version 1.0.2, URL https://loelschlaeger.de/fHMM/.

Platen E, Rendek R (2008). "Empirical evidence on Student-t log-returns of diversified world stock indices." *Journal of Statistical Theory and Practice*, **2**. doi:10.1080/15598608.2008.10411873.

Pohle J, Langrock R, van Beest F, Schmidt N (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), 270–293.

Schwarz G (1978). "Estimating the dimension of a model." *The Annals of Statistics*, **6**.

Team RC (2021). *R: a language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Turner R, Liu L (2014). *hmm.discnp: hidden Markov models with discrete non-parametric observation distributions*. R package, version 0.2–3, URL http://CRAN.R-project.org/package=hmm.discnp.

Visser I, Speekenbrink M (2010). "depmixS4: an R package for hidden Markov models." *Journal of Statistical Software*, **36**, 1–21.

Zucchini W, MacDonald I, Langrock R (2016). "Hidden Markov models for time series: an introduction using R, 2nd Edition." *Chapman and Hall/CRC*.

**Affiliation:**

Lennart Oelschläger
Department of Business Administration and Economics
Bielefeld University
Postfach 10 01 31, Germany
E-mail: lennart.oelschlaeger@uni-bielefeld.de

Timo Adam
School of Mathematics and Statistics
University of St Andrews
The Observatory, Buchanan Gardens, St Andrews KY16 9LZ, UK
E-mail: ta59@st-andrews.ac.uk

Rouven Michels
Department of Business Administration and Economics
Bielefeld University
Postfach 10 01 31, Germany
E-mail: r.michels@uni-bielefeld.de

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd