Computational issues in parameter estimation for hidden Markov models with Template Model Builder

Timothée Bacri[a], Geir D. Berentsen [b], Bård Støve[a], and Jan Bulla[a]

[a] Department of Mathematics, University of Bergen, Postbox 7803, 5007 Bergen, Norway
[b] Department of Business and Management Science, Norwegian School of Economics, 5045 Bergen, Norway

ARTICLE HISTORY
Compiled February 9, 2023

ABSTRACT
A popular way to estimate the parameters of a hidden Markov model (HMM) is direct numerical maximization (DNM) of the (log-)likelihood function. The advantages of employing the TMB [1] framework in R for this purpose were illustrated recently [2]. In this paper, we present extensions of these results in two directions.
First, we present a practical way to obtain uncertainty estimates in form of confidence intervals (CIs) for the so-called smoothing probabilities at moderate computational and programming effort via TMB. Our approach thus permits to avoid computer-intensive bootstrap methods. By means of several examples, we illustrate patterns present for the derived CIs.
Secondly, we investigate the performance of popular optimizers available in R when estimating HMMs via DNM. Hereby, our focus lies on the potential benefits of employing TMB. Investigated criteria via a number of simulation studies are convergence speed, accuracy, and the impact of (poor) initial values. Our findings suggest that all optimizers considered benefit in terms of speed from using the gradient supplied by TMB. When supplying both gradient and Hessian from TMB, the number of iterations reduces, suggesting a more efficient convergence to the maximum of the log-likelihood. Last, we briefly point out potential advantages of a hybrid approach.

KEYWORDS
Hidden Markov model; Template Model Builder; Smoothing probabilities; Confidence intervals; Maximum likelihood estimation; Robustness; Initial conditions

Supporting Information for this article is available from the author or online under `https://timothee-bacri.github.io/HMM_with_TMB`

1. Introduction

Hidden Markov models (HMMs) are a well-studied and popular class of models in many areas. While they have been used for speech recognition historically [see, e.g. 3–7], these models also became important in many other fields due to their flexibility. These are, to name only a few, biology and bioinformatics [8–10], finance [11], ecology [12], stochastic weather modeling [13,14], and engineering [15]. In short, HMMs are a class of models where the given data is assumed to follow varying distributions according to an underlying unknown Markov chain.
Parameter estimation for HMMs is typically achieved either by the Baum-Welch (BW) algorithm [5,16–19] - an Expectation-Maximization(EM)-type algorithm - or in a quite straightforward fashion by direct numerical maximization (DNM) of the (log-)likelihood function [see, e.g., 20,21]. A discussion of both approaches can be found in [22, p. 358]. Furthermore, [23] points out that it is challenging to fit complex models with the BW algorithm, [24, pp. 77-78] advises using DNM with HMMs due to the ease of fitting complex models, and

CONTACT Timothée Bacri. Email: timot@bacri.eu

[20,25] report a greater speed of DNM compared to the BW algorithm. Nevertheless, the BW is highly accepted and finds widespread application. However, we will only use DNM in the following due to the possibility of accelerating the estimation via the R-package `TMB` [1,2].

In this paper, we present a straightforward procedure to calculate uncertainty estimates of the *smoothing probabilities* through appropriate use of `TMB`. Hence, we quantify the uncertainty of the state classifications of the observations at a low computational cost. To the best of our knowledge, such results on confidence intervals for the aforementioned probabilities are not available in the literature.

Furthermore, the chosen optimization routine for DNM plays an important role in parameter estimation. While e.g. [24] focuses on the unconstrained optimizer `nlm`, alternatives include the popular `nlminb` [26] and many others, such as those provided by the `optim` function [27]. In this context, [28] compare different estimation routines for two-state HMMs: a Newton-Type algorithm [29,30], the Nelder-Mead algorithm [31], BW, and a hybrid algorithm successively executing BW and DNM. To complement these studies, we investigate the speed of several optimization algorithms, many of which allow the gradient and Hessian of the objective function as input. Particular focus lies on the ability of the `TMB` framework, which allows for easy retrieval of the exact gradient and Hessian (up to machine precision). This enriches the existing literature on parameter estimation for HMMs.

Among others, [24] report that “Depending on the starting values, it can easily happen that the algorithm (i.e. DNM) identifies a local, but not the global, maximum”. The literature on how to tackle this problem is rich, and includes among others: the use of artificial neural networks to guess the initial values [32], model induction [33], genetic algorithms in combination with HMMs [34], hybridizing both the BW and the DNM algorithms [23,28,35], specific assumptions on the parameters [36], or educated guesses [24, p. 53]. We investigate how stable the various considered optimization routines are towards poor initial values. Again, this part takes under consideration the special features of `TMB`, which have not been investigated yet.

The paper is organized as follows. In Section 2, we provide a brief overview of parameter estimation for HMMs, including inference for CIs. In Section 3 we show how uncertainty estimates of the smoothing probabilities can be computed via `TMB`. Then, we apply our results to a couple of data sets with different characteristics. We perform simulation studies in Sections 4 and 5. Therein, we compare measures of performance and accuracy of various optimizers and check how well these optimizers perform in the presence of poor initial values. Section 6 provides some concluding remarks. All code necessary to reproduce our results is available in the supporting information.

2. Basics on hidden Markov models

The HMMs considered here are fit on observed time series $\{X_t : t = 1, \dots, T\}$ where t denotes the (time) index ranging from one to T. In this setting, a mixture of conditional distributions is assumed to be driven by an unobserved (hidden) homogeneous Markov chain, whose states will be denoted as $\{C_t : t = 1, \dots, T\}$. We will use different conditional distributions in the paper. First, we specify an m-state Poisson HMM, i.e. with the conditional distribution

$$p_i(x) = \mathrm{P}(X_t = x | C_t = i) = \frac{e^{-\lambda_i} \lambda_i^x}{x!}$$

with parameters $\lambda_i, i = 1, ..., m$. Secondly, we consider Gaussian HMMs with conditional distribution specified by

$$p_i(x) = \mathrm{P}(X_t = x | C_t = i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_i}\right)^2},$$

with parameters $(\mu_i, \sigma_i), i = 1, \ldots, m$. In addition, $\mathbf{\Gamma} = \{\gamma_{ij}\}, i, j = 1, ..., m$ denotes the transition probability matrix (TPM) of the HMM's underlying Markov chain, and $\boldsymbol{\delta}$ is a vector of length m collecting the corresponding stationary distribution. We assume that the Markov chains underlying our HMMs are irreducible and aperiodic. This ensures the existence and uniqueness of a stationary distribution as the limiting distribution [37]. However, these results are of limited relevance for most estimation algorithms, because the elements of $\mathbf{\Gamma}$ are generally strictly positive. Nevertheless, one should be careful when manually fixing selected elements of $\mathbf{\Gamma}$ to zero.

The basis for our estimation procedure is the (log-)likelihood function. We denote the "history" of observations x_t and the observed process X_t up to time t by $\boldsymbol{x^{(t)}} = \{x_1, \ldots, x_t\}$ and $\boldsymbol{X^{(t)}} = \{X_1, \ldots, X_t\}$, respectively. In addition, let $\boldsymbol{\theta}$ be the vector of model parameters. As explained, e.g., by [24, p. 37], the likelihood function can then be represented as a product of matrices:

$$L(\boldsymbol{\theta}) = \boldsymbol{P}(\boldsymbol{X^{(T)}} = \boldsymbol{x^{(T)}}) = \boldsymbol{\delta P}(x_1)\mathbf{\Gamma}\boldsymbol{P}(x_2)\mathbf{\Gamma}\boldsymbol{P}(x_3)\ldots\mathbf{\Gamma}\boldsymbol{P}(x_T)\mathbf{1}', \tag{1}$$

where the m conditional probability density functions evaluated at x (we use this term for discrete support as well) are collected in the diagonal matrix

$$\boldsymbol{P}(x) = \begin{pmatrix} p_1(x) & & & 0 \\ & p_2(x) & & \\ & & \ddots & \\ 0 & & & p_m(x) \end{pmatrix},$$

and $\mathbf{1}'$ and $\boldsymbol{\delta}$ denote a transposed vector of ones and the stationary distribution, respectively. That is, we assume that the initial distribution corresponds to $\boldsymbol{\delta}$. The likelihood function given in Equation 1 can be efficiently evaluated by a so-called forward pass through the observations, as illustrated in Appendix A. Consequently, it is possible to obtain $\hat{\boldsymbol{\theta}}$, the ML estimates of $\boldsymbol{\theta}$, by - in our case - unconstrained DNM. Since several of our parameters are constrained, we rely on known re-parametrization procedures (see Appendix B for details).

Confidence intervals (CIs) for the estimated parameters of HMMs can be derived via various approaches. The most common ones are Wald-type, profile likelihood, and bootstrap-based CIs. [2] shows that `TMB` can yield valid Wald-type CIs in a fraction of the time required by classical bootstrap-based methods as investigated e.g. by [24,28]. Furthermore, the likelihood profile method may fail to provide CIs [2]. Therefore, we rely on Wald-type confidence intervals derived via `TMB`. For details, see Appendix C.

3. Uncertainty of smoothing probabilities

When interpreting the estimation results of an HMM, the so-called smoothing probabilities often play an important role. These quantities, denoted by $p_{it}(\boldsymbol{\theta})$ in the following, correspond to the probability of being in state i at time t given the observations, i.e.,

$$p_{it}(\boldsymbol{\theta}) = \mathrm{P}_{\boldsymbol{\theta}}(C_t = i | X^{(T)} = x^{(T)}) = \frac{\alpha_t(i)\beta_t(i)}{L(\boldsymbol{\theta})},$$

and can be calculated for $i = 1, \ldots, m$ and $t = 1, \ldots, T$ [see, e.g., 24, p. 87]. The involved quantities $\alpha_t(i)$ and $\beta_t(i)$, which also depend on $\boldsymbol{\theta}$, result directly from the forward- and backward algorithm, respectively, as illustrated in Appendix A. Estimates of $p_{it}(\boldsymbol{\theta})$ are obtained by $p_{it}(\hat{\boldsymbol{\theta}})$ where $\hat{\boldsymbol{\theta}}$ is the ML estimate of $\boldsymbol{\theta}$.

An important feature of `TMB` is that it not only permits obtaining standard errors for $\hat{\boldsymbol{\theta}}$ (and CIs for $\boldsymbol{\theta}$), but in principle also for any other quantity depending on the parameters. This is achieved by combining the delta method with automatic differentiation (see [1] for details). A minor requirement for deriving standard deviations for $p_{it}(\hat{\boldsymbol{\theta}})$ via the `ADREPORT` function provided by `TMB` consists in implementing the function $p_{it}(\boldsymbol{\theta})$ in `C++`. Then, this part has to be integrated into the scripts necessary for the parameter estimation procedure. It is noteworthy that, once implemented, the procedures related to inference of the smoothing probabilities do not need to change when the HMM considered changes, because the input vector for $p_{it}(\boldsymbol{\theta})$ remains the same. The `C++` code in the supporting information illustrates how to complete these tasks. Note that population value we are interested in constructing a CI for is $p_{it}(\boldsymbol{\theta})$, which should not be confused with $E_{X^{(T)}}\left(P_{\hat{\boldsymbol{\theta}}(X^{(T)})}(C_t = i \mid X^{(T)})\right)$. That is, we treat $p_{it}(\boldsymbol{\theta})$ as the probability of the event $C_t = i$ conditional on $X^{(T)}$ being equal to the particular sequence of observations $x^{(T)}$. After all, it is this sequence and the corresponding smoothing probabilities which are of interest to the researcher.

Quantifying the uncertainty in the estimated smoothing probabilities can be of interest in multiple fields because the underlying states are often linked to certain events of interest (e.g., the state of an economy or the behavior of a customer). In the following, we illustrate the results of our approach through a couple of examples.

3.1. Track Your Tinnitus

The "Track Your Tinnitus" (TYT) mobile application gathered a large data set, a description of which is detailed in [38] and [39]. The data plotted in Figure 1 shows the so-called "arousal" variable reported for an individual over 87 consecutive days. This variable takes high values when a high level of excitement is achieved and low values when the individual is in a calm emotional state. The values are measured on a discrete scale; we refer to [40,41] for details.

We estimated Poisson HMMs with varying number of states by `nlminb` with `TMB`'s gradient and Hessian functions passed as arguments (this approach was chosen for all estimated models in our examples). The preferred HMM in terms of AIC and BIC is a two-state model, see Table D1 in Appendix D.

Figure 2 displays the corresponding smoothing probabilities with 95% Wald-type CIs, constructed using the standard error provided by `TMB`. Intuitively, one might expect the uncertainty to be low when a smoothing probability takes values close to zero or one, whereas higher uncertainty should be inherent to smoothing probabilities further away from these bounds. The CIs illustrated in Figure 2 follow this pattern. However, as pointed out by [2], this data set is atypically short for fitting HMMs. As we will see in the following, different patterns will emerge for longer sequences of observations.

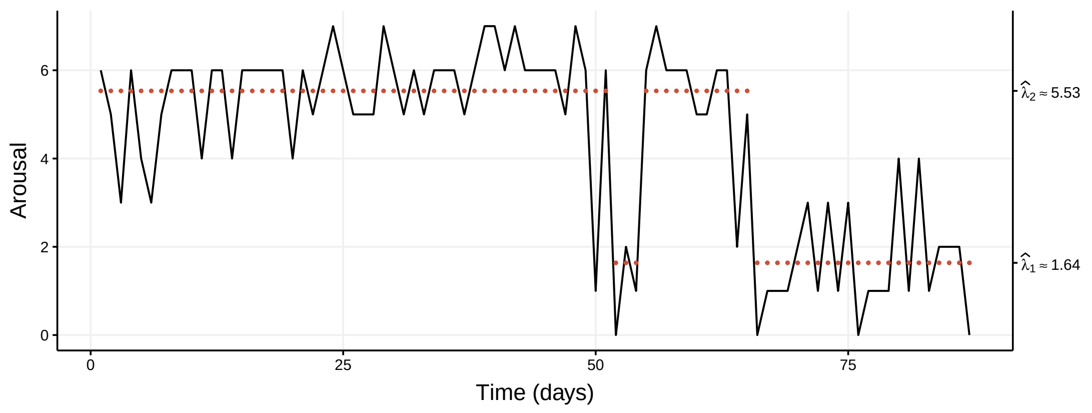

Figure 1. Plot of a two-state Poisson HMM fitted to the TYT data (of size 87). The colored dots correspond to the conditional mean of the inferred state at each time. Model estimates are displayed in Table D1.

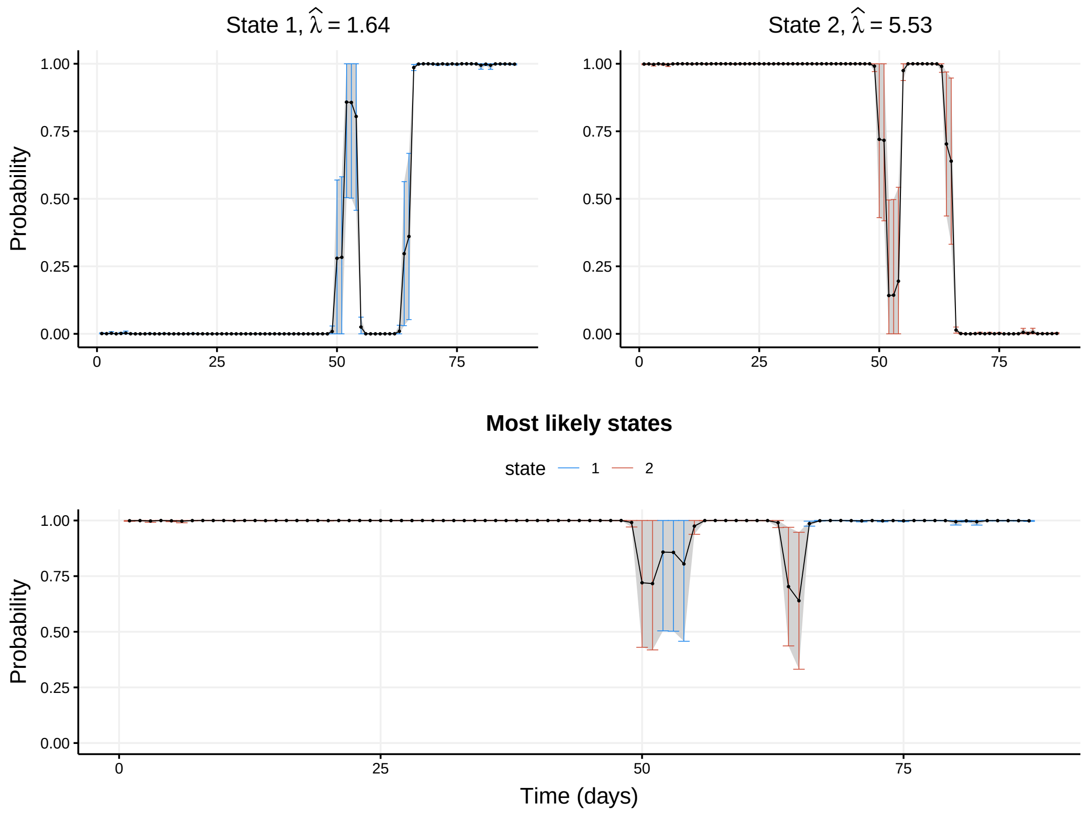

Figure 2. Smoothing probabilities and confidence intervals of a two-state Poisson HMM fitted to the TYT data set. The solid line shows the smoothing probability estimates and the 95% CIs are represented by vertical lines. The bottom graph displays smoothing probabilities for the most likely state estimated for each data point. The vertical confidence interval lines are colored differently per hidden state. Further details on the model estimates are available in Table D1.

3.2. Soap

In the following, we consider a data set of weekly sales of a soap in a supermarket. The data were provided by the Kilts Center for Marketing, Graduate School of Business of the University of Chicago and are available in the database at `https://research.chicagobooth.edu/kilts/marketing-databases/dominicks`. The data are displayed in Figure 3. Similarly to the previous section, we fitted multiple Poisson HMMs, and the preferred HMM in terms of AIC and BIC is a two-state model as well (see Table D1 in Appendix D). Figure 4 displays the smoothing probabilities resulting from this model. The patterns of the CIs described in the previous section are, in principle, also present here. Nevertheless, an exception is highlighted in the bottom panel of Figure 4: at $t = 152$, the smoothing probability of State 2 is closer to the upper boundary of one than the corresponding probability at $t = 147$. Yet, the uncertainty is higher at $t = 152$.

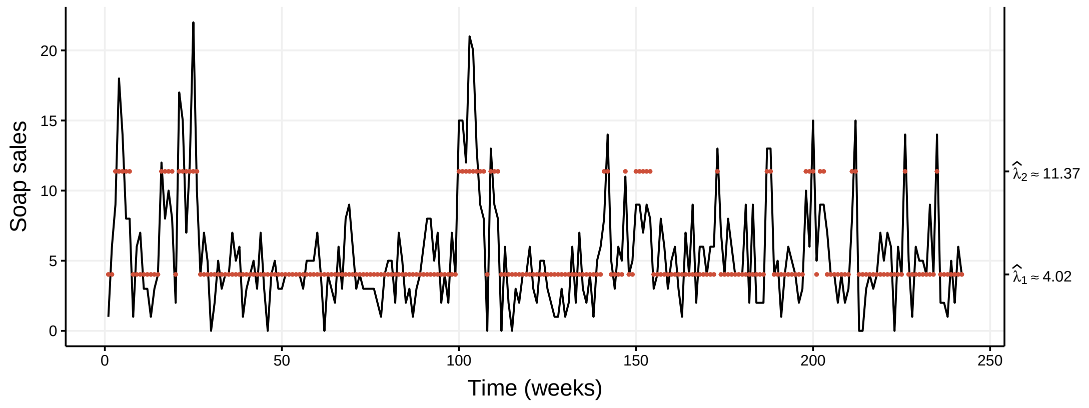

Figure 3. Plot of the soap data (of size 242), with fitted two-state Poisson HMM. The colored dots correspond to the conditional mean of the inferred state at each time. Model estimates are displayed in Table D1.

3.3. Weekly returns

The data set considered in this section are 2230 weekly log-returns based on the adjusted closing share price of the S&P 500 stock market index retrieved from Yahoo Finance, between January 1^{st} 1980 and September 30^{th} 2022. The returns are expressed in per cent to facilitate reading and interpreting the estimates. As shown, e.g., by [42], Gaussian HMMs reproduce well certain stylized facts of financial returns. We thus estimated such models with varying number of states. A three-state model is preferred by the BIC, whereas the AIC is almost identical for three and four states (see Table D1). The estimates show a decrease in conditional standard deviation with increasing conditional mean. This is a well-known property of many financial returns (see, e.g., [43–45]) related to the behavior of market participants in crisis and calm periods. Figure 5 shows the first 200 data points plotted along with conditional means from the preferred model.

The subsequent Figure 6 shows inferred smoothing probabilities together with their CIs. In addition to the previous applications, where smoothing probabilities close to the boundary seemed to be linked to lower uncertainty, this data set shows also some clear exceptions from this pattern. For example, the smoothing probabilities of State 3 inferred at $t = 30$ and $t = 83$ take the relatively close values 0.51 and 0.53, respectively. However, the associated uncertainty at $t = 30$ is visibly higher than the corresponding uncertainty at $t = 83$ (the estimated standard errors are 0.205 and 0.045). This may be explained by the fact that the inferred states closely around $t = 30$ oscillated between State 2 and 3, thus indicating a high uncertainty of the state classification during this period. On the contrary, around $t = 83$ a quick and persistent change from State 2 to 3 takes place.

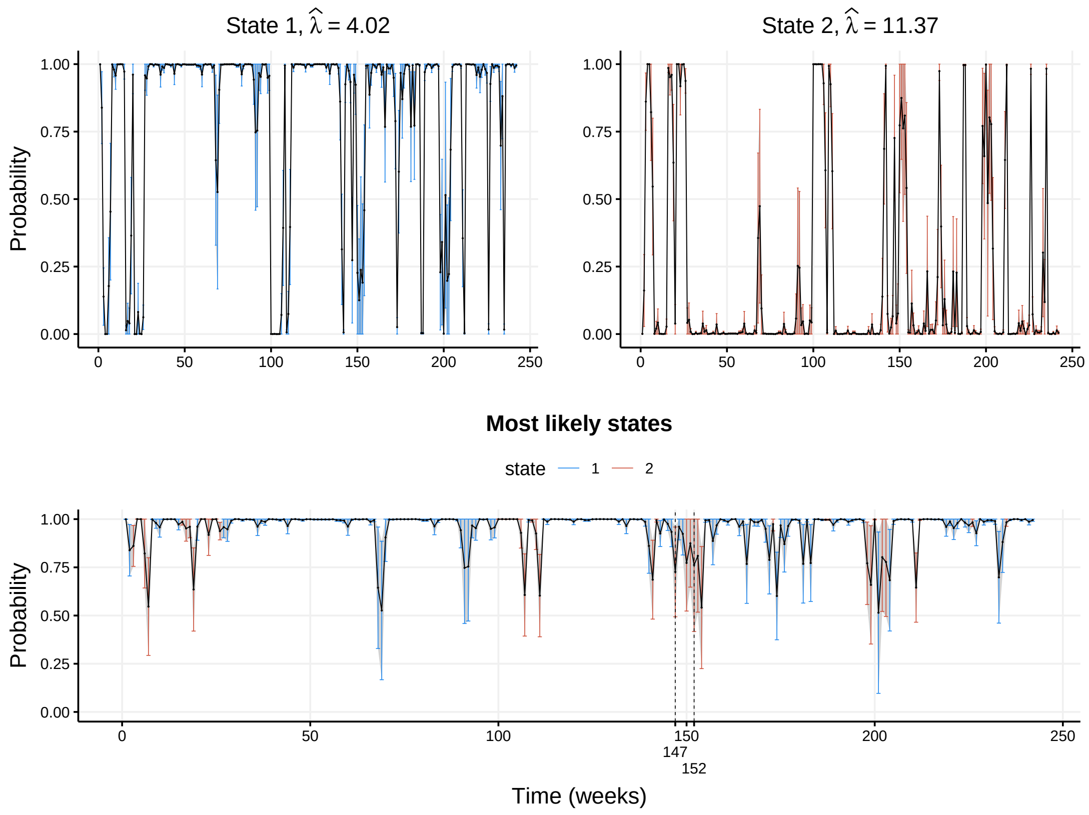

Figure 4. Smoothing probabilities and confidence intervals of a two-state Poisson HMM fitted to the soap data set. The solid line shows the smoothing probability estimates and the 95% CIs are represented by vertical lines. The bottom left graph displays smoothing probabilities for the most likely state estimated for each data point. The vertical confidence interval lines are colored differently per hidden state. Further details on the model estimates are available in Table D1.

3.4. Hospital

Basis for our last example is a data set issued by the hospital ”Hôpital Lariboisière” from Assistance Publique – Hôpitaux de Paris (a french hospital trust). This data set consists of the hourly number of patient arrivals to the emergency ward during a period of roughly 10 years. A subset of the data (over one week) is displayed in Figure 7. We examine this due to several reasons. First, with 87648 observations, this data set is much larger than the ones examined previously. Secondly, the medical staff noticed differences between the rates of patient arrivals at day and night, respectively, which motivates the use of, e.g., an HMM. Last, a Poisson HMM is a natural candidate for the observed count-type data. The preferred model by both AIC and BIC has five states (see Table D2 in Appendix D) and confirms the impression of the hospital employees: State 5 is mainly visited during core operating hours during day time. The fourth and third states mainly occur late afternoon, early evening, and early morning. Last, State 1 and 2 correspond to night time observations.

Even for a data set of this size, the smoothing probabilities and corresponding CIs can be derived with moderate computational effort and time: Figure 8 displays the results. Overall, the inferred CIs are relatively small, in particular in comparison with those obtained for the stock return data discussed in the previous example. The low uncertainty in the state classification may most likely result from the clear, close to periodic transition patterns.

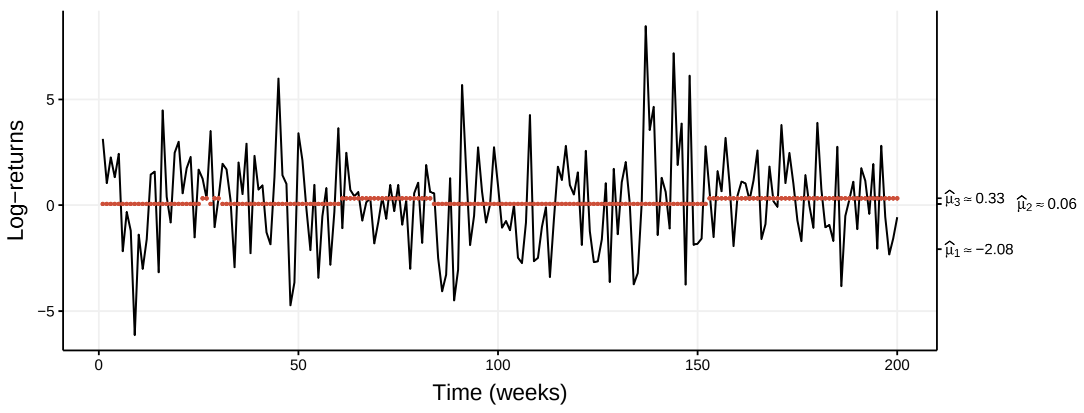

Figure 5. Plot of the weekly returns (of size 2230). The colored dots correspond to the conditional mean of the inferred state from a fitted 3-state Gaussian HMM (see Table D1). For readability, only the first 200 data are plotted.

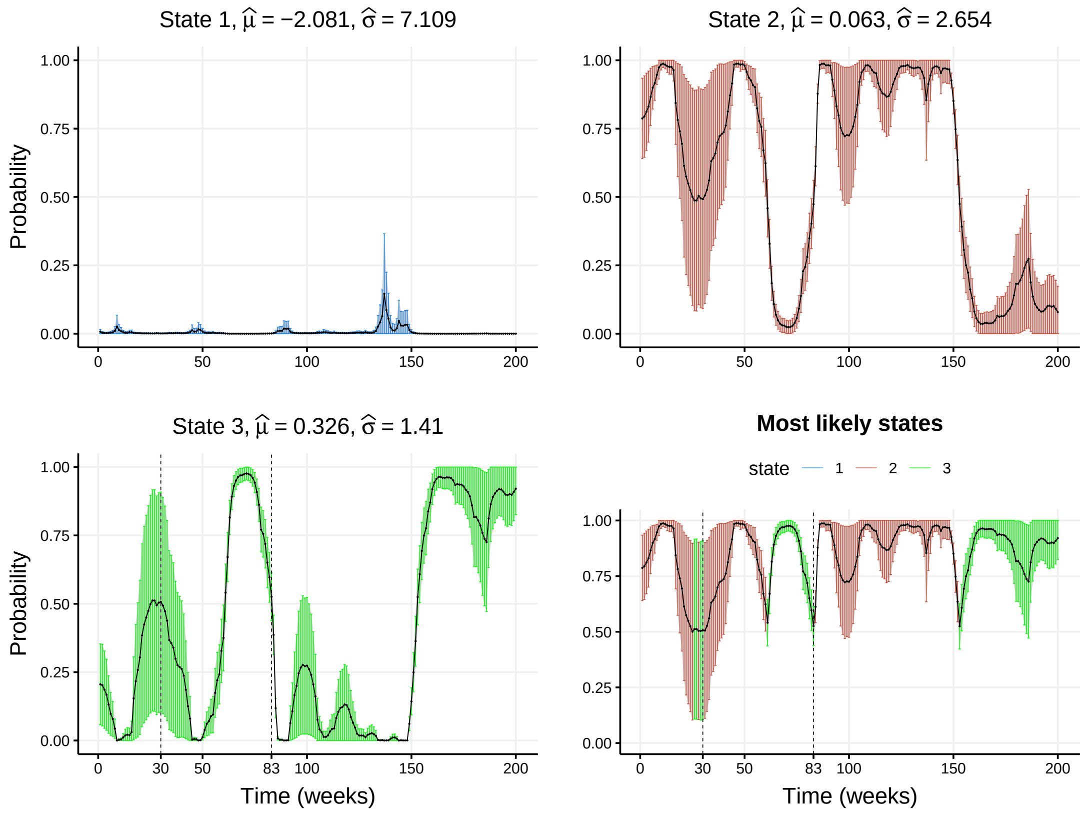

Figure 6. Smoothing probabilities and confidence intervals of a three-state Gaussian HMM fitted to the weekly returns data. The solid line shows the smoothing probability estimates and the 95% CIs are represented by vertical lines. The bottom right graph displays smoothing probabilities for the most likely state estimated for each data point. The vertical confidence interval lines are colored differently per hidden state. Only the first 200 out of the 2230 values are shown for readability purposes. Further details on the model estimates are available in Table D1.

4. Performance and accuracy of different optimizers

This section compares the speed and accuracy of different optimizers in R using `TMB` in a HMM estimation setting with DNM by several simulation studies with different settings.

In the first setting, we simulate time series consisting of 87 observations from a two-state

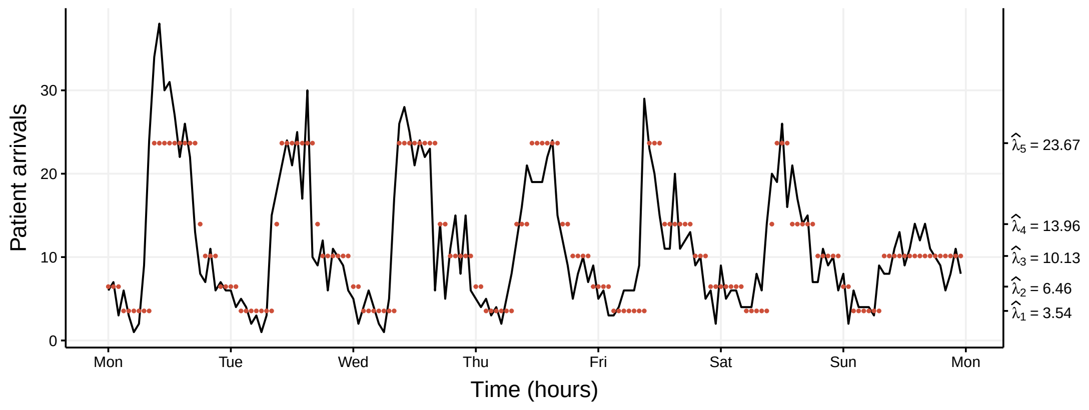

Figure 7. Plot of the hospital data (of size 87648). The colored dots correspond to the conditional mean of the inferred state at each time. For readability, only the first week starting on Monday (from Monday January 4^{th} 2010 at 00:00 to Sunday January 10^{th} 2010 at 23:59) is plotted instead of the entire 10 years. Model estimates are displayed in Table D2.

Poisson HMM fitted to the TYT data, as we want to examine the different optimizers performance on data sets with relatively few observations. In the second setting, we simulate time series of length 200 from a two-state Poisson HMM, and in the third setting, time series of length 200 from a two-state Gaussian HMM are simulated. The specific choice of parameters is described in more detail in the forthcoming sections.

The comparisons that focus on computational speed and iterations of the different optimizers, are based on 200 replications from the different models, while the studies focusing on the accuracy of the optimizers are based on 1000 replications from the different models. Hence, the Monte-Carlo simulation setup in this paper closely resembles the setup of [2].

4.1. *Computational setup*

The results presented in this paper required installing and using the `R`-package `TMB` and the software `Rtools`, where the latter is needed to compile `C++` code. Scripts were coded in the interpreted programming language `R` [46] using its eponymous software environment `RStudio`. For the purpose of making our results reproducible, the generation of random numbers is handled with the function `set.seed` under `R` version number 3.6.0. A seed is set multiple times in our scripts, ensuring that smaller parts can be easily duplicated without executing lengthy prior code. A workstation with 4 Intel(R) Xeon(R) Gold 6134 processors (3.7 GHz) running under the Linux distribution Ubuntu 18.04.6 LTS (Bionic Beaver) with 384 GB RAM was used to execute our scripts.

4.2. *Selected optimizers and additional study design*

There exist a large number of optimizers for use within the `R` ecosystem, see e.g. [47]. In a preliminary phase, multiple optimization routines were attempted, and the ones failing to converge on a few selected test data sets from the designs mentioned above were excluded (details available from the author upon request). Hence, the following optimizers were selected for the simulation studies:

- `optim` from the `stats` package to test the BFGS [48–51], L-BFGS-B [52], CG [53], and Nelder-Mead [31] algorithms.
- `nlm` [54] and `nlminb` [26] from the same package to test popular non-linear uncon-

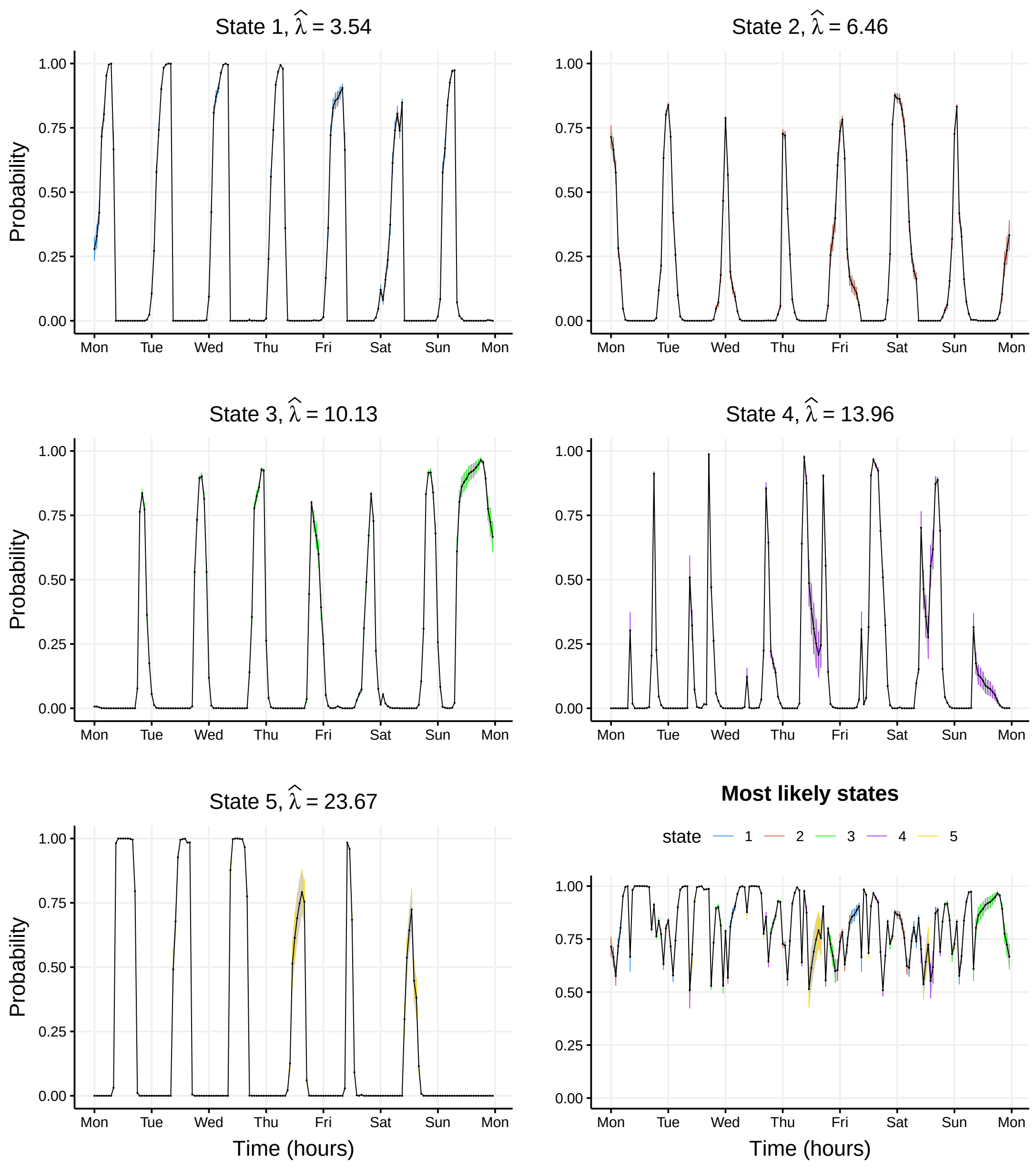

Figure 8. Smoothing probabilities and confidence intervals of a five-state Gaussian HMM fitted to the hospital data set. The solid line shows the smoothing probability estimates and the 95% CIs are represented by vertical lines. The bottom right graph displays smoothing probabilities for the most likely state estimated for each data point. The vertical confidence interval lines are colored differently per hidden state. For readability, only the first week starting on Monday (from Monday January 4^{th} 2010 at 00:00 to Sunday January 10^{th} 2010 at 23:59) is plotted instead of the entire 10 years. Further details on the model estimates are available in Table D2.

strained optimization algorithms.

- `hjn` from the `optimr` package [27] to test the so-called Hooke and Jeeves Pattern Search Optimization [55].
- `marqLevAlg` from the `marqLevAlg` package [56] to test a general-purpose optimization based on the Marquardt-Levenberg algorithm [57,58].
- `ucminf` from the `ucminf` package [59] to test a general-purpose optimization based on the Marquardt-Levenberg algorithm [60].

- `BBoptim` from the `BB` package [61] to test a spectral projected gradient method for large scale optimization with simple constraints [62].
- `newuoa` from the `minqa` package [63] to test a large-scale nonlinear optimization algorithm [64].

By also providing the Hessian and/or gradient to the optimizers that support such inputs, 22 optimization procedures are examined for each study design. For the study of the computational speed and number of iterations, we applied the `microbenchmark` package [65] that allows us to obtain precise durations for each of the simulated data sets. Furthermore, HMM ML estimates were reordered by increasing Poisson rates or Gaussian means to avoid a random order of the states.

It is important to note that we discarded all those samples for which the simulated state sequence did not sojourn at least once in each state. This was to avoid identifiability and convergence issues when trying to estimate a m-state HMM on a data set where only $m - 1$ states are visited. Further, samples where at least one optimizer failed to converge were also discarded to ensure comparability. For the same reason, we set the maximum number of iterations to 10^4 whenever possible. Note that some optimizers (`BFGS, L-BFGS-B, CG, Nelder-Mead, hjn, newuoa`) do not report a number of iterations and are therefore missing from our comparisons of iterations. Finally, we use the true values as initial parameters. For the TYT data, initial values are calculated as medians from the fitted models resulting from all optimizers listed above.

4.3. Results

In this section, we report the results from the simulation studies focusing on the performance of different optimizers in terms of the computational speed and the lack or presence of unreasonable estimates. This section uses `R` scripts that may be of interest to users investigating their own HMM settings, and are available in the supporting information. We begin by reporting the speed of all optimization routines and thereafter the accuracy. The routine names contain the optimizer names and may be followed by either "_gr" to indicate that the routine uses `TMB`'s provided gradient function, "_he" to indicate the use of `TMB`'s provided Hessian function, or "_grhe" to indicate the use of both gradient and Hessian functions provided by `TMB`. Note that some optimizers cannot make use of these functions and hence bear no extension at the end of their names.

4.3.1. Simulation study: TYT setting

Figure 9 shows the time required by each optimization routine, measured over the 200 replication. The number of iterations required by each optimization routine is also reported where available. Times range from 1 to 1340 milliseconds. As shown by the large medians and wide CIs, `CG` and `CG_gr` require substantially more time to estimate a HMM, compared to the other optimizers. Furthermore, optimizers that take a long time to estimate HMMs require a high amount of iterations as well, as expected. Notably, optimization routines almost always take longer when the Hessian is passed. This likely results from a higher computational burden due to handling and evaluating Hessian matrices via `TMB`.

Looking at the details, the optimizer `BFGS_gr` is among the fastest optimizers, and this is partly explained by a relatively low number of iterations, and by the fact that it does not make use of the Hessian. However, we note that `nlm` and `nlmimb` families of optimizers are slower when the gradient and Hessian are not passed as argument. The optimizer `BBoptim` is comparatively slow, and `nlminb_grhe` fails to converge comparably often.

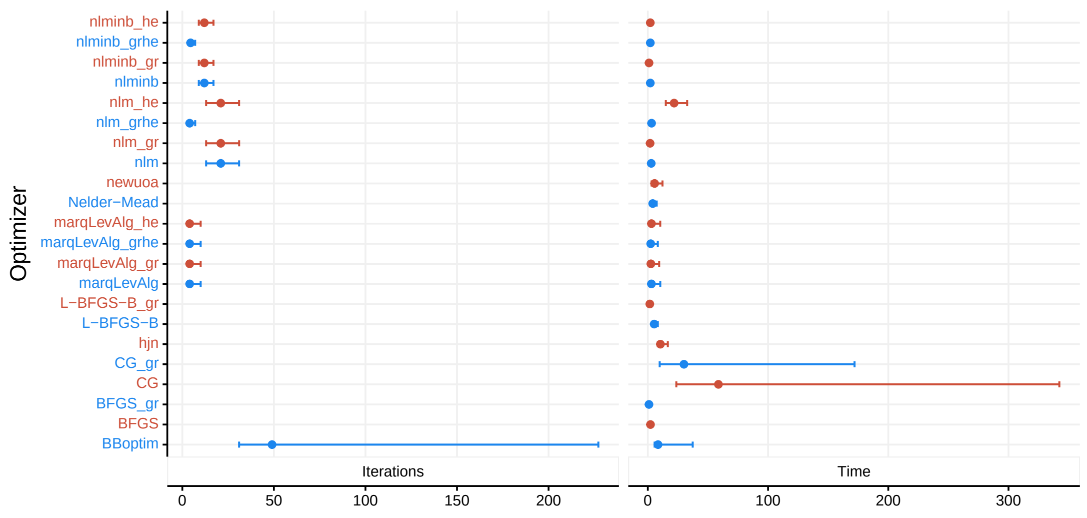

Figure 9. Median duration (in milliseconds) and median number of iterations together with 2.5%- and 97.5%-quantile when fitting two-state Poisson HMMs to 200 replications in the first setting.

In addition to time durations, we investigate the accuracy of the optimizers, based on another 1000 replications from the two-state Poisson HMM in the first setting. The main conclusion from this simulation is that the medians and empirical quantiles calculated are in fact almost identical across all optimizers, and very close to the true parameter values used in the simulations. Results are reported in Figure E1 in Appendix E. These result imply that the different optimizers are equally accurate. Note, however, that the variation is quite high for some of the estimated parameters and for the nll, which results most likely from the limited number of observations.

4.3.2. Simulation study: two-state Poisson HMM

In this second simulation setting, the parameters of our two-state Poisson HMM are given by

$$\boldsymbol{\lambda} = (1, 7), \quad \boldsymbol{\Gamma} = \begin{pmatrix} 0.95 & 0.05 \\ 0.15 & 0.85 \end{pmatrix},$$

and the sample size is fixed at 200. Figure 10 illustrates our results, which are in line with those obtained in the first setting. More precisely, `BBoptim` still dominates the number of iterations, and `CG` and `CG_gr` require more computational time than the others. Overall, most optimizers need more time compared to the previous setting as there is more data to process, but the time increase for `nlm_he` and `hjn` is much more substantial than for other optimizers. Moreover, although the data set is larger, some optimizers (e.g. `BBoptim` and `BFGS_gr`) perform as fast as or faster than in the previous setting. Furthermore, passing `TMB`'s Hessian to optimizers does not always slow down the optimization. This can be seen, e.g., from `nlminb_he` which exhibits a lower median time than `nlminb`. A last notable pattern concerns the popular `nlm` optimizer. When only the Hessian but not the gradient is supplied from `TMB`, the computational time increases substantially compared to `nlm_gr` and `nlm_grhe`. Interestingly, the equally popular optimizer `nlminb` is not affected in the same way. Thus, one should be careful when only supplying the Hessian from `TMB` while an optimizer carries out the gradient approximation. In any case, the low number of iterations required by both nlminb_grhe and nlm_grhe indicate a preferable convergence behavior when supplying both quantities.

In terms of accuracy of the parameter estimates and obtained nlls, our results are highly similar to those described in the first setting (for details, see Figure E2 in Appendix E).

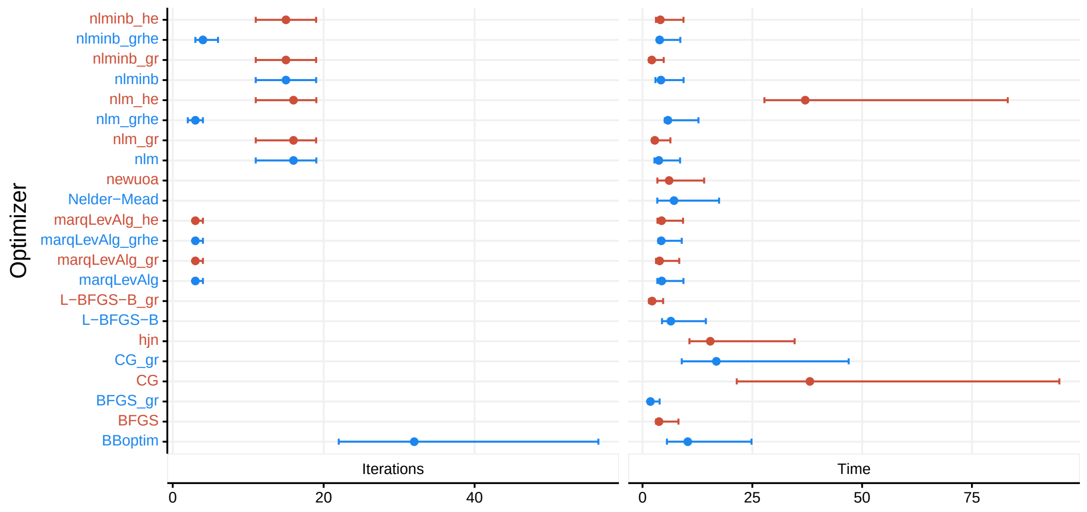

Figure 10. Median duration (in milliseconds) and median number of iterations together with 2.5%- and 97.5%-quantile when fitting two-state Poisson HMMs to 200 replications in the second setting.

4.3.3. *Simulation study: two-state Gaussian HMM*

In our third and last simulation setting, the parameters of a two-state Gaussian HMM are given by

$$\mathbf{\Gamma} = \begin{pmatrix} 0.95 & 0.05 \\ 0.15 & 0.85 \end{pmatrix}, \quad \boldsymbol{\mu} = (-5, 5), \quad \boldsymbol{\sigma} = (1, 5),$$

and the sample size is fixed at 200. In terms of computational time and the number of iterations, the results observed for this setup are similar to the previously studied Poisson HMMs (see Figure 11). A minor aspect is an increase in both iterations and computational time, which is not surprising given the higher number of parameters.

Regarding estimation accuracy of the parameter estimates and obtained nlls, our results are again highly similar to those obtained in the previous two settings (illustrated graphically by Figure E3 in Appendix E).

5. Robustness to initial value selection

In this section, we examine the impact of initial values of the parameters - including poorly selected ones - on the convergence behavior and accuracy of different optimizers. We investigate one real and several simulated data sets. In order to “challenge” the optimizers, all data sets are of comparable small size.

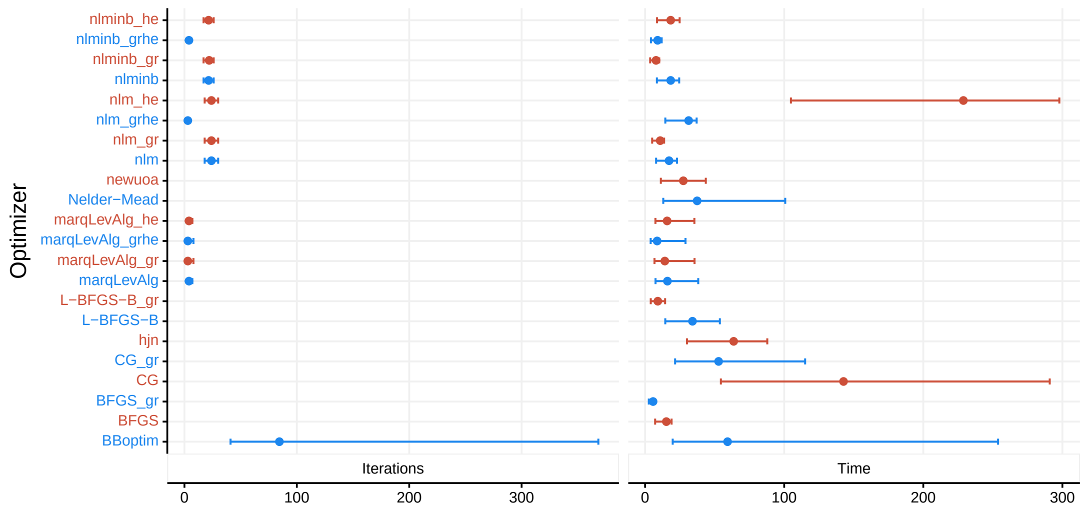

Figure 11. Median duration (in milliseconds) and median number of iterations together with 2.5%- and 97.5%-quantile when fitting two-state Gaussian HMMs to 200 replications in the third setting.

5.1. *Study design*

We again consider three settings in the following. In the first, we investigate the TYT data set. In the second and third setting, we focus on simulated data from Poisson and Gaussian HMMs, respectively. More specifically, for the Poisson HMMs we generate one data set of sample size 200 for each of the HMMs considered. These HMMs are defined by all possible combinations of the following Poisson rates and TPMs:

$$\boldsymbol{\lambda} \in \{(1,4),(5,7)\} \text{ and}$$
$$\boldsymbol{\Gamma} \in \left\{ \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{pmatrix}, \begin{pmatrix} 0.7 & 0.3 \\ 0.8 & 0.2 \end{pmatrix}, \begin{pmatrix} 0.1 & 0.9 \\ 0.8 & 0.2 \end{pmatrix}, \begin{pmatrix} 0.55 & 0.45 \\ 0.45 & 0.55 \end{pmatrix} \right\}.$$

This setup leads to the generation of 2 x 4 data sets. For the Gaussian HMMs, we follow the same approach, where the Gaussian means and standard deviations are selected from the sets

$$\boldsymbol{\mu} \in \{(-2,2),(-1,4)\} \text{ and}$$
$$\boldsymbol{\sigma} \in \{(1.5,2.5),(0.5,1.5)\}.$$

This setting thus generates 2 x 2 x 4 data sets.

For each data set, we pass a large range of initial values to the same optimizers considered in the previous section. In this way, we investigate the resistance of each optimizer to potentially poor initial values when fitting HMMs. To generate sets of initial values, we consider the following potential candidates for Poisson rates $\boldsymbol{\lambda}$, Gaussian means $\boldsymbol{\mu}$, Gaussian standard deviations $\boldsymbol{\sigma}$, and TPMs $\boldsymbol{\Gamma}$:

$$
\begin{aligned}
&\lambda_1, \lambda_2 \in \{M, M+0.5, M+1, 1.5, \ldots, x_{max}\}, \text{ where } \lambda_1 < \lambda_2,\\
&\mu_1, \mu_2 \in \{x_{min}, x_{min}+0.5, x_{min}+1, \ldots, x_{max}-0.5, x_{max}\}, \text{ where } \mu_1 < \mu_2,\\
&\sigma_1, \sigma_2 \in \left\{ \text{10 equidistant points going from } \sqrt{\frac{(x_{max}-x_{min})^2}{2T}} \text{ to } \sqrt{(x_{max}-\widehat{\mu})(\widehat{\mu}-x_{min})} \right\},\\
&\text{and } \boldsymbol{\Gamma} \in \Bigg\{ \begin{pmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix}, \begin{pmatrix} 0.1 & 0.9 \\ 0.8 & 0.2 \end{pmatrix}, \ldots, \begin{pmatrix} 0.1 & 0.9 \\ 0.1 & 0.9 \end{pmatrix},\\
&\qquad \begin{pmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{pmatrix}, \begin{pmatrix} 0.2 & 0.8 \\ 0.8 & 0.2 \end{pmatrix}, \ldots, \begin{pmatrix} 0.2 & 0.8 \\ 0.1 & 0.9 \end{pmatrix},\\
&\qquad\qquad \ldots \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \Bigg\},
\end{aligned}
$$

where $x_{min} = \min(x_1, x_2, \ldots, x_T)$, $x_{max} = \max(x_1, x_2, \ldots, x_T)$, $\widehat{\mu} = \frac{1}{T}\sum_{i=1}^{T} x_i$ and $M = \max(0.5, x_{min})$. The motivation for this selection is as follows. First, Poisson means have to be greater than zero, so we set their lower boundary to M. Secondly, the $\widehat{\lambda}_i$'s have to belong to the interval (x_{min}, x_{max}), cfr. [66]. This applies to the Gaussian means as well. Thirdly, the upper and lower limit of σ_1, σ_2 is motivated by the Bhatia-Davis [67] and the von Szokefalvi Nagy inequalities [68].

For the Poisson HMMs fitted, we consider all possible combination of the parameters described above. For the Gaussian HMMs, however, we sample 12500 initial parameters for each of the 16 data sets to reduce computation time (the total amount reduces from 39066300 to 200000). As in the previous Section 4, we limit the maximal number of iterations to 10^4 whenever possible. Other settings were left as default.

To evaluate the performance of the optimizers considered, we rely on two criteria. First, we register whether the optimizer converges (all errors are registered regardless of their type). Secondly, when an optimizer converges, we also register if the nll is equal to the "true" nll (with a gentle margin of ± 5%). We calculate this "true" nll as the median of all nlls derived with all optimizers initiated with the true parameter values.

5.2. Results

Table 1 shows the results for the two-state Poisson HMMs fitted to the TYT data. The four `marqLevAlg` follow closely. We note that `hjn` has the best performance: it basically does not fail and converges almost always. Moreover, `nlminb_grhe` fails to converge much more often than other optimizers, but is the most likely to find the global maximum when it converges. For the other optimizers, the failure rate is generally low (less than 5%) and the global maximum is found in more than 65% of the cases.

Table 2 reports the results from the second setting, also with two-state Poisson HMMs. The results show that all optimizers find the global maximum in the majority of cases ($> 96\%$) when converging. Moreover, the failure rates are relatively low for all optimizers with the exception of `CG`, which stands out with comparably high failure rates.

Finally, Table 3 presents the results from the third setting with Gaussian HMMs. The results regarding the failure rates point in the same direction as those from the Poisson HMM setting: `CG` attains the highes failure rate, followed by `newuoa` and `L-BFGS-B`. The remaining optimizers perform satisfactorily. Note, however, that `nlminb` does not benefit from being supplied with the gradient and Hessian from `TMB`. Regarding the global maximum found, many

Table 1. Performance of multiple optimizers estimating Poisson HMMs from the TYT data (first setting) over 7371 different sets of initial parameters. The first column lists all optimizers. The second column shows how often optimizers fail to converge successfully. The third column displays how often optimizers successfully found the global maximum of the nll when converging.

	Failures (%)	Global maximum found (%)
BFGS	0.00	66.83
BFGS_gr	0.00	66.83
L-BFGS-B	3.43	71.38
L-BFGS-B_gr	3.46	71.22
CG	4.90	71.48
CG_gr	4.91	71.49
Nelder-Mead	0.00	69.88
nlm	0.00	65.19
nlm_gr	0.00	65.22
nlm_he	0.00	65.19
nlm_grhe	0.00	72.99
nlminb	0.00	74.96
nlminb_gr	0.00	74.94
nlminb_he	0.00	74.96
nlminb_grhe	24.41	99.96
hjn	0.00	97.40
marqLevAlg	1.64	88.43
marqLevAlg_gr	1.99	88.64
marqLevAlg_he	1.64	88.43
marqLevAlg_grhe	1.94	88.60
newuoa	0.00	71.67
BBoptim	0.00	72.11

algorithms reach success rates of about 95% or more. Notable exceptions are `Nelder-Mead` and `nlm` when not both gradient and Hessian are provided by `TMB`. As observed previously for the TYT data, `nlm` benefits strongly from the full use of `TMB`.

Table 2. Performance of multiple optimizers estimating Poisson HMMs from the second setting over 194481 different sets of initial parameters. The first column lists all optimizers. The second column shows how often optimizers fail to converge successfully. The third column displays how often optimizers successfully found the global maximum of the nll when converging.

	Failures (%)	Global maximum found (%)
BFGS	0.00	96.68
BFGS_gr	0.00	96.68
L-BFGS-B	0.30	100.00
L-BFGS-B_gr	0.29	100.00
CG	14.23	99.49
CG_gr	14.47	99.53
Nelder-Mead	0.00	99.91
nlm	0.07	98.42
nlm_gr	0.02	98.42
nlm_he	0.07	98.42
nlm_grhe	0.00	99.50
nlminb	0.00	100.00
nlminb_gr	0.00	100.00
nlminb_he	0.00	100.00
nlminb_grhe	4.62	100.00
hjn	0.00	99.99
marqLevAlg	3.19	99.98
marqLevAlg_gr	3.20	100.00
marqLevAlg_he	3.19	99.98
marqLevAlg_grhe	3.20	100.00
newuoa	0.34	99.77
BBoptim	4.20	100.00

Table 3. Performance of multiple optimizers estimating Gaussian HMMs from the third setting over 200000 different sets of initial parameters randomly drawn from 39066300 different candidate sets of initial values. The first column lists all optimizers. The second column shows how often optimizers fail to converge successfully. The third column displays how often optimizers successfully found the global maximum of the nll when converging.

	Failures (%)	Global maximum found (%)
BFGS	0.00	84.24
BFGS_gr	0.00	84.24
L-BFGS-B	6.58	94.40
L-BFGS-B_gr	6.75	94.45
CG	8.62	98.68
CG_gr	8.82	98.81
Nelder-Mead	0.03	80.55
nlm	0.39	87.77
nlm_gr	0.11	87.73
nlm_he	0.39	87.77
nlm_grhe	0.32	97.21
nlminb	0.22	97.18
nlminb_gr	0.26	97.20
nlminb_he	0.22	97.18
nlminb_grhe	5.69	97.97
hjn	0.53	95.21
marqLevAlg	2.55	98.18
marqLevAlg_gr	2.66	98.34
marqLevAlg_he	2.55	98.18
marqLevAlg_grhe	2.64	98.33
newuoa	7.58	97.44
BBoptim	1.13	98.61

5.3. Hybrid algorithm

Last, we ran a small simulation study investigating a hybrid algorithm within the `TMB` framework in the spirit of [28]. The hybrid algorithm starts with the `Nelder-Mead` optimizer and switches to `nlminb` after a certain number of iterations. The idea is to benefit from both `Nelder-Mead`'s rapid departure from poor initial values and from the high convergence speed of `nlminb`. Our study is performed on 1000 random sets of initial values in a similar fashion to Section 5, using the weekly returns data set. For every set of initial values, we sequentially increase the number of iterations carried out by `Nelder-Mead` by ten (starting at one) until convergence or until 10000 iterations has been reached, in which case we classify the attempt as a failure. For comparison, we also run an optimization only with `nlminb` starting directly with the same sets of inital values. In both cases, `nlminb` benefits from the gradient and Hessian provided by `TMB`.

Figure 12 reports the results of our study. The left most columns show that only one iteration was required to converge in 827 of the 1000 sets of initial values with the hybrid algorithm, while `nlminb` converges in only 698 out of these 827 sets. Furthermore, for 39 additional sets of initial values, the hybrid algorithm converges with 10 iterations carried out by the `Nelder-Mead` optimizer, whereas `nlminb` fails 26 times on these sets. Increasing the number of initial iterations carried out by `Nelder-Mead` for the hybrid algorithm leads to similar patterns. Finally, both optimization routines fail to converge for 18 sets of initial values. Thus, in total, the hybrid algorithm with one or more `Nelder-Mead` iterations failed in only 1.8% of the cases. In comparison, direct use of `nlminb` failed in around 12% of the cases. This indicates that the hybrid algorithm requires rather few iterations of `Nelder-Mead` to improve the convergence rate substantially.

6. Concluding remarks

This paper addresses a couple of aspects concerning parameter estimation for HMMs via `TMB` and `R` using direct numerical maximization (DNM) of the likelihood. The advantage of `TMB` is that it permits to quantify the uncertainty of any quantity depending on the estimated parameters. This is achieved by combining the delta method with automatic differentiation. Moreover, `TMB` provides exact calculations of first- and second-order derivatives of the (log-)likelihood of a model by automatic differentiation. This allows for efficient gradient- and/or Hessian-based optimization of the likelihood.

In the first part of the paper, we propose a straightforward technique to quantify the uncertainty of smoothing probabilities via the calculation of Wald-type CIs. The advantage of this approach is that one avoids computationally intensive bootstrap methods. By means of several examples, we illustrate how such CIs provide additional insight into the commonly used state classification via smoothing probabilities. For a practitioner working with HMMs, the presented uncertainty quantification constitutes a new tool for obtaining a better understanding of the dynamics of the hidden process.

Subsequently, we examine speed and accuracy of the different optimizers in three simulation settings. Our results show a slight variation in both number of iterations and computational speed. In particular, the optimizers `nlminb`, `nlm`, `marqLevAlg`, and `BFGS` usually possess the highest computational speed. The computing time required by all optimizers reduces when using the gradient provided by `TMB`. The number of iterations reduces in particular for `nlminb` and `nlm` when employing both gradient and Hessian from `TMB`. This suggests a more direct path towards the (global) likelihood maximum compared to optimizer-internal approximation routines.

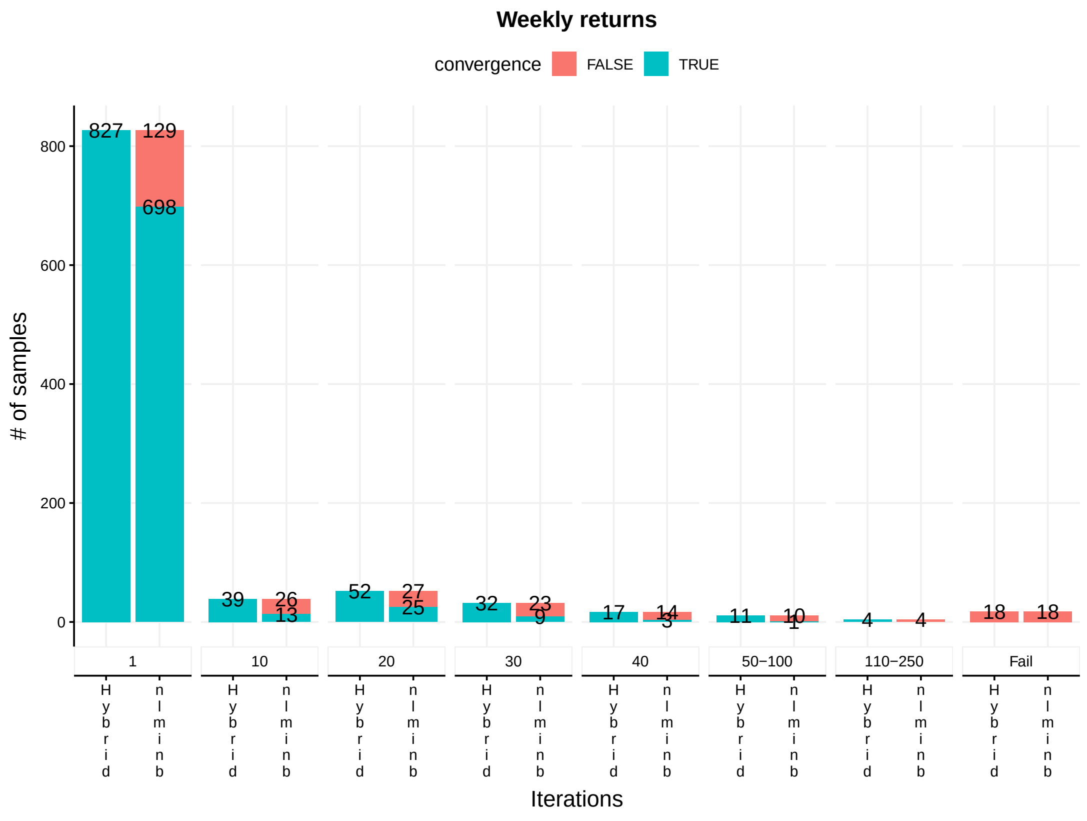

Figure 12. Convergence counts of `nlminb` and a hybrid algorithm (first `Nelder-Mead`, then `nlminb`). The hybrid algorithm uses 1, 10, 20,. . . iterations for the `Nelder-Mead`, then passes the estimates to `nlminb` as initial values. We randomly picked 1000 sets of initial values out of 1458000 potential candidates.

Regarding the accuracy, measured in terms of attained negative log-likelihood and parameter estimates (and their variability) the results show little variations across the different optimizers. This is indeed a positive finding, indicating that the performance of the optimizers is relatively equal in terms of accuracy - however, this statment only holds true when using optimal initial value. Then, we examine robustness towards initial values across different optimizers. In three settings, we measure a) how often optimizers fail to converge and b) how often they successfully reach the global maximum of the log-likelihood function when starting from a wide range of sets of initial values. In particular, `nlm`, `nlminb`, and `hjn` show an overall good performance. Notably, `nlm` benefits strongly from employing both gradient and Hessian provided by `TMB`, which is not the case for `nlminb`. Altogether, we observe a trade-off between failure rates and convergence to the global maximum. Nevertheless, none of the optimizers shows an exceptionally bad performance.

Finally, we illustrate that a hybrid algorithm starting with the `Nelder-Mead` optimizer and switching to `nlminb` after a certain number of iterations converges more often than `nlminb`. Notably an effect is visible even with a single initial iteration carried out by `Nelder-Mead`.

Acknowledgements

We thank the University of Regensburg and European School for Interdisciplinary Tinnitus Research (ESIT) for providing access to the TYT data. Special thanks go to J. Simões for data preparation, and W. Schlee and B. Langguth for helpful comments and suggestions. We also thank Anthony Chauvin and Bertrand Galichon for providing access to the hospital data set along with the necessary authorizations. J. Bulla's profound thanks for the continued moral support go to the highly skilled, very helpful, and always friendly employees of Vestbrygg AS (Org. ID 912105954).

Disclosure statement

The authors have declared no conflict of interest.

Funding

The work of J. Bulla was supported by the GENDER-Net Co-Plus Fund (GNP-182). This work was also supported by the Financial Market Fund (Norwegian Research Council project no. 309218).

References

[1] Kristensen K, Nielsen A, Berg C, et al. TMB: Automatic differentiation and laplace approximation. Journal of Statistical Software, Articles. 2016;70(5):1–21.

[2] Bacri T, Berentsen GD, Bulla J, et al. A gentle tutorial on accelerated parameter and confidence interval estimation for hidden Markov models using Template Model Builder. Biometrical Journal. 2022 May;.

[3] Juang BH, Rabiner LR. Hidden Markov Models for Speech Recognition. Technometrics. 1991 Aug;33(3):251–272.

[4] Baum LE, Petrie T. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. The Annals of Mathematical Statistics. 1966 Dec;37(6):1554–1563.

[5] Rabiner L. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE. 1989 Feb;77(2):257–286.

[6] Rabiner L, Juang B. An introduction to hidden Markov models. IEEE ASSP Magazine. 1986 Jan; 3(1):4–16.

[7] Fredkin DR, Rice JA. Bayesian Restoration of Single-Channel Patch Clamp Recordings. Biometrics. 1992;48(2):427–448.

[8] Schadt EE, Sinsheimer JS, Lange K. Computational Advances in Maximum Likelihood Methods for Molecular Phylogeny. Genome Research. 1998 Jan;8(3):222–233.

[9] Durbin R. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press; 1998.

[10] Eddy SR. Profile hidden Markov models. Bioinformatics. 1998 Jan;14(9):755–763.

[11] Hamilton JD. A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. Econometrica. 1989;57(2):357–384.

[12] McClintock BT, Langrock R, Gimenez O, et al. Uncovering ecological state dynamics with hidden Markov models. Ecology Letters. 2020;23(12):1878–1903.

[13] Lystig TC, Hughes JP. Exact Computation of the Observed Information Matrix for Hidden Markov Models. Journal of Computational and Graphical Statistics. 2002 Sep;11(3):678–689.

[14] Ailliot P, Allard D, Monbet V, et al. Stochastic weather generators: an overview of weather type models. Journal de la société française de statistique. 2015;156(1):101–113.

[15] Mor B, Garhwal S, Kumar A. A Systematic Review of Hidden Markov Models and Their Applications. Archives of Computational Methods in Engineering. 2021 May;28(3):1429–1448.

[16] Baum LE, Petrie T, Soules G, et al. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. The Annals of Mathematical Statistics. 1970; 41(1):164–171.

[17] Dempster AP, Laird NM, Rubin DB. Maximum Likelihood from Incomplete Data Via the EM Algorithm. Journal of the Royal Statistical Society: Series B (Methodological). 1977;39(1):1–22.

[18] Liporace L. Maximum likelihood estimation for multivariate observations of Markov sources. IEEE Transactions on Information Theory. 1982 Sep;28(5):729–734.

[19] Wu CFJ. On the Convergence Properties of the EM Algorithm. The Annals of Statistics. 1983; 11(1):95–103.

[20] Turner R. Direct maximization of the likelihood of a hidden Markov model. Computational Statistics & Data Analysis. 2008 May;52(9):4147–4160.

[21] MacDonald IL, Zucchini W. Hidden Markov and other models for discrete-valued time series. (Monographs on Statistics and Applied Probability; Vol. 70). London: Chapman & Hall; 1997.

[22] Cappé O, Moulines E, Ryden T. Inference in Hidden Markov Models. Springer Science & Business Media; 2006.

[23] Lange K, Weeks DE. Efficient computation of lod scores: Genotype elimination, genotype redefinition, and hybrid maximum likelihood algorithms. Annals of Human Genetics. 1989;53(1):67–83.

[24] Zucchini W, MacDonald I, Langrock R. Hidden markov models for time series: An introduction using r, second edition. CRC Press; 2016. Chapman & Hall/CRC Monographs on Statistics & Applied Probability.

[25] Altman RM, Petkau AJ. Application of hidden Markov models to multiple sclerosis lesion count data. Statistics in Medicine. 2005;24(15):2335–2344.

[26] Gay DM. Usage summary for selected optimization routines. Computing science technical report. 1990;153:1–21.

[27] Nash JC, Varadhan R. optimr: A replacement and extension of the optim function; 2019. R package version 2019-12.16; Available from: `https://CRAN.R-project.org/package=optimr`.

[28] Bulla J, Berzel A. Computational issues in parameter estimation for stationary hidden Markov models. Computational Statistics. 2008 Jan;23(1):1–18.

[29] Dennis J John E, Moré JJ. Quasi-Newton Methods, Motivation and Theory. SIAM Review. 1977 Jan;19(1):46–89.

[30] Schnabel RB, Koonatz JE, Weiss BE. A modular system of algorithms for unconstrained minimization. ACM Transactions on Mathematical Software. 1985 Dec;11(4):419–440.

[31] Nelder JA, Mead R. A Simplex Method for Function Minimization. The Computer Journal. 1965 Jan;7(4):308–313.

[32] Hassan MR, Nath B, Kirley M. A fusion model of HMM, ANN and GA for stock market forecasting. Expert Systems with Applications. 2007 Jul;33(1):171–180.

[33] Stolcke A, Omohundro S. Hidden Markov model induction by Bayesian model merging. Advances in neural information processing systems. 1993;:11–11.

[34] Oudelha M, Ainon RN. HMM parameters estimation using hybrid Baum-Welch genetic algorithm. In: 2010 International Symposium on Information Technology; Vol. 2; Jun.; 2010. p. 542–545.

[35] Redner RA, Walker HF. Mixture Densities, Maximum Likelihood and the EM Algorithm. SIAM

Review. 1984 Apr;26(2):195–239.

[36] Kato J, Watanabe T, Joga S, et al. An HMM-based segmentation method for traffic monitoring movies. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002 Sep;24(9):1291–1296.

[37] Feller W. An introduction to probability theory and its applications. Wiley; 1968.

[38] Pryss R, Reichert M, Langguth B, et al. Mobile Crowd Sensing Services for Tinnitus Assessment, Therapy, and Research. In: 2015 IEEE International Conference on Mobile Services; Jun. IEEE; 2015. p. 352–359.

[39] Pryss R, Reichert M, Herrmann J, et al. Mobile Crowd Sensing in Clinical and Psychological Trials – A Case Study. In: 2015 IEEE 28th International Symposium on Computer-Based Medical Systems; Jun. IEEE; 2015. p. 23–24.

[40] Probst T, Pryss R, Langguth B, et al. Emotion dynamics and tinnitus: Daily life data from the “TrackYourTinnitus” application. Scientific Reports. 2016 Aug;6(1):31166.

[41] Probst T, Pryss RC, Langguth B, et al. Does Tinnitus Depend on Time-of-Day? An Ecological Momentary Assessment Study with the “TrackYourTinnitus” Application. Frontiers in Aging Neuroscience. 2017 Aug;9:253.

[42] Rydén T, Teräsvirta T, Åsbrink S. Stylized Facts of Daily Return Series and the Hidden Markov Model. Journal of Applied Econometrics. 1998;13(3):217–244.

[43] Schwert GW. Why does stock market volatility change over time. Journal of Finance. 1989; 44(5):1115–1153.

[44] maheu JM, McCurdy TH. Identifying Bull and Bear Markets in Stock Returns. Journal of Business & Economic Statistics. 2000 Jan;18(1):100–112.

[45] Guidolin M, Timmermann A. Economic Implications of Bull and Bear Regimes in UK Stock and Bond Returns. The Economic Journal. 2005 Jan;115(500):111–143.

[46] R Core Team. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; 2021.

[47] Schwendinger F, Borchers HW. CRAN Task View: Optimization and Mathematical Programming [Https://cran.r-project.org/view=optimization]; 2022.

[48] Broyden CG. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. IMA Journal of Applied Mathematics. 1970 Mar;6(1):76–90.

[49] Fletcher R. A new approach to variable metric algorithms. The Computer Journal. 1970 Jan; 13(3):317–322.

[50] Goldfarb D. A family of variable-metric methods derived by variational means. Mathematics of Computation. 1970;24(109):23–26.

[51] Shanno DF. Conditioning of quasi-Newton methods for function minimization. Mathematics of Computation. 1970;24(111):647–656.

[52] Byrd RH, Lu P, Nocedal J, et al. A Limited Memory Algorithm for Bound Constrained Optimization. SIAM Journal on Scientific Computing. 1995 Sep;16(5):1190–1208.

[53] Fletcher R. Practical Methods of Optimization. John Wiley & Sons; 2013.

[54] Dennis JE, Schnabel RB. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for Industrial and Applied Mathematics; 1996. Classics in Applied Mathematics.

[55] Hooke R, Jeeves TA. “ Direct Search” Solution of Numerical and Statistical Problems. Journal of the ACM. 1961 Apr;8(2):212–229.

[56] Philipps V, Proust-Lima C, Prague M, et al. marqlevalg: A parallelized general-purpose optimization based on marquardt-levenberg algorithm; 2022. R package version 2.0.7; Available from: `https://CRAN.R-project.org/package=marqLevAlg`.

[57] Levenberg K. A method for the solution of certain non-linear problems in least squares. Quarterly of Applied Mathematics. 1944;2(2):164–168.

[58] Marquardt DW. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. Journal of

the Society for Industrial and Applied Mathematics. 1963 Jun;11(2):431–441.

[59] Nielsen HB, Mortensen SB. ucminf: General-purpose unconstrained non-linear optimization; 2022. R package version 1.1-4.1; Available from: `https://CRAN.R-project.org/package=ucminf`.

[60] Nielsen HB. UCMINF - an algorithm for unconstrained, nonlinear optimization. Informatics and Mathematical Modelling, Technical University of Denmark, DTU; 2000.

[61] Varadhan R, Gilbert P. Bb: Solving and optimizing large-scale nonlinear systems; 2019. R package version 2019.10-1; Available from: `http://www.jhsph.edu/agingandhealth/People/Faculty_personal_pages/Varadhan.html`.

[62] Varadhan R, Gilbert P. BB: An R Package for Solving a Large System of Nonlinear Equations and for Optimizing a High-Dimensional Nonlinear Objective Function. Journal of Statistical Software. 2010;32:1–26.

[63] Bates D, Mullen KM, Nash JC, et al. minqa: Derivative-free optimization algorithms by quadratic approximation; 2022. R package version 1.2.5; Available from: `http://optimizer.r-forge.r-project.org`.

[64] Powell MJD. The NEWUOA software for unconstrained optimization without derivatives. In: Di Pillo G, Roma M, editors. Large-Scale Nonlinear Optimization. Boston, MA: Springer US; 2006. Nonconvex Optimization and Its Applications; p. 255–297.

[65] Mersmann O. microbenchmark: Accurate timing functions; 2021. R package version 1.4.9; Available from: `https://github.com/joshuaulrich/microbenchmark/`.

[66] Böhning D. Computer-assisted analysis of mixtures and applications: Meta-analysis, disease mapping and others. Vol. 81. CRC press; 1999.

[67] Bhatia R, Davis C. A Better Bound on the Variance. The American Mathematical Monthly. 2000 Apr;107(4):353–357.

[68] Nagy J. Über algebraische Gleichungen mit lauter reellen Wurzeln. Jahresbericht der Deutschen Mathematiker-Vereinigung. 1918;27:37–43.

[69] Visser I, Raijmakers MEJ, Molenaar PCM. Confidence intervals for hidden Markov model parameters. British Journal of Mathematical and Statistical Psychology. 2000;53(2):317–327.

[70] Efron B, Tibshirani RJ. An introduction to the bootstrap. New York, N.Y.; London: Chapman & Hall; 1993.

[71] Härdle W, Horowitz J, Kreiss JP. Bootstrap Methods for Time Series. International Statistical Review. 2003;71(2):435–459.

Appendix

Appendix A. Forward algorithm and backward algorithm

The pass through observations can be efficiently computed piece by piece, from left (i.e., time $s = 1$) to right (i.e., time $s = t$) or right to left by the so-called "forward algorithm" and "backward algorithm" respectively. With these two algorithms, the likelihood can be computed recursively. To formulate the forward algorithm, let us define the vector α_t for $t = 1, 2, \dots, T$ so that

$$\begin{aligned}\boldsymbol{\alpha}_t &= \boldsymbol{\delta}\boldsymbol{P}(x_1)\boldsymbol{\Gamma}\boldsymbol{P}(x_2)\boldsymbol{\Gamma}\boldsymbol{P}(x_3)\dots\boldsymbol{\Gamma}\boldsymbol{P}(x_t)\\ &= \boldsymbol{\delta}\boldsymbol{P}(x_1)\prod_{s=2}^{t}\boldsymbol{\Gamma}\boldsymbol{P}(x_s)\\ &= (\alpha_t(1), \dots, \alpha_t(m))\end{aligned}$$

for $t = 1, 2, \dots, T$. The name of the algorithm comes from computing it via the recursion

$$\begin{gathered}\boldsymbol{\alpha}_0 = \boldsymbol{\delta}\boldsymbol{P}(x_1),\\ \boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}\boldsymbol{\Gamma}\boldsymbol{P}(x_t) \text{ for } t = 1, 2, \dots, T.\end{gathered}$$

After passing through all observations until time T, the likelihood is derived from

$$L(\boldsymbol{\theta}) = \boldsymbol{\alpha}_T\mathbf{1}'.$$

The backward algorithm is very similar to the forward algorithm. The sole difference lies in starting from the last instead of the first observation. To specify the backward algorithm, let us define the vector $\boldsymbol{\beta}_t$ for $t = 1, 2, \dots, T$ so that

$$\begin{aligned}\boldsymbol{\beta}_t' &= \boldsymbol{\Gamma}\boldsymbol{P}(x_{t+1})\boldsymbol{\Gamma}\boldsymbol{P}(x_{t+2})\dots\boldsymbol{\Gamma}\boldsymbol{P}(x_T)\dots\mathbf{1}'\\ &= \left(\prod_{s=t+1}^{T}\boldsymbol{\Gamma}\boldsymbol{P}(x_s)\right)\mathbf{1}'\\ &= (\beta_t(1), \dots, \beta_t(m)).\end{aligned}$$

The name backward algorithm results from the way of recursively calculating $\boldsymbol{\beta}_t$ by

$$\begin{gathered}\boldsymbol{\beta}_T = \mathbf{1}'\\ \boldsymbol{\beta}_t = \boldsymbol{\Gamma}\boldsymbol{P}(x_{t+1})\boldsymbol{\beta}_{t+1} \text{ for } t = T-1, T-2, \dots, 1.\end{gathered}$$

As before, the likelihood can be obtained after a pass through all observations by

$$L(\boldsymbol{\theta}) = \boldsymbol{\delta}\boldsymbol{\beta}_1.$$

Usually, only the forward algorithm is used for parameter estimation. Nonetheless, either or both algorithms may be necessary to derive certain quantities of interest, e.g., smoothing probabilities (as we explore in Section 3) or forecast probabilities. For practical implementation, attention needs to be paid to underflow errors which can quickly occur in both algorithms (due to the elements of the TPM and / or conditional probabilties taking values in $[0, 1]$). Therefore, we implemented a scaled version of the forward algorithm as suggested by [24, p. 48]. This version directly provides the negative log-likelihood (nll) as result.

Appendix B. Reparameterization of the likelihood function

Along with the data, $\boldsymbol{\theta}$ serves as input for calculating the likelihood by the previously illustrated forward algorithms. Some constraints need to be imposed on $\boldsymbol{\theta}$:

(i) In the case of the Poisson HMM, the parameter vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)$ necessary for computing the conditional distribution has to consist of strictly positive elements.
(ii) The elements γ_{ij} of the TPM $\boldsymbol{\Gamma}$ must be non-negative, and each row of $\boldsymbol{\Gamma}$ needs to sum up to to one.

Tacking these constraints by means of constrained optimization routines when maximizing the likelihood can lead to several difficulties. For example, constrained optimization routines may require a significantly amount of fine-tuning. Moreover, the number of available optimization routines is significantly reduced. A common alternative works by reparametrizing the log-likelihood as a function of unconstrained, so-called “working” parameters $\{\mathbf{T}, \boldsymbol{\eta}\} = \mathbf{g^{-1}}(\boldsymbol{\Gamma}, \boldsymbol{\lambda})$, as follows. One possibility to reparametrize $\boldsymbol{\Gamma}$ is given by

$$\tau_{ij} = \log\left(\frac{\gamma_{ij}}{1 - \sum_{k \neq i} \gamma_{ik}}\right) = \log(\gamma_{ij}/\gamma_{ii}) \text{ for } i \neq j.$$

where $\tau_{ij} \in \mathbb{R}$ are $m(m-1)$ unconstrained elements of an $m \times m$ matrix $\mathbf{T}$ with missing diagonal elements. The matching reverse transformation is

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

and the diagonal elements of $\boldsymbol{\Gamma}$ follow from $\sum_j \gamma_{ij} = 1$, $i = 1, ..., m$ (see [24], p. 51).

For a Poisson HMM, a simple method to reparametrize the conditional Poisson means $\boldsymbol{\lambda}$ into $\boldsymbol{\eta}$ is given by $\eta_i = \log(\lambda_i)$ for $i = 1, \dots, m$. Consequently, the constrained “natural” parameters are obtained via $\lambda_i = \exp(\eta_i)$. Estimation of the natural parameters $\{\boldsymbol{\Gamma}, \boldsymbol{\lambda}\}$ can therefore be obtained by maximization of the reparametrized likelihood with respect to $\{\mathbf{T}, \boldsymbol{\eta}\}$ then by a transformation of the estimated working parameters back to natural parameters via the above transformations, i.e. $\{\hat{\boldsymbol{\Gamma}}, \hat{\boldsymbol{\lambda}}\} = g(\hat{\boldsymbol{P}}, \hat{\boldsymbol{\eta}})$. In general, the above procedure requires that the function g is one-to-one.

With a Gaussian HMM, the means are already unconstrained and do not require any transformation. However, the standard deviations can be transformed similarly to the Poisson rates via $\eta_i = \log(\sigma_i)$ for $i = 1, \dots, m$, and the “natural” parameters are then obtained via the corresponding reverse transformation $\sigma_i = \exp(\eta_i)$.

Appendix C. Confidence intervals

When handling statistical models, confidence intervals (CIs) for the estimated parameters can be derived via various approaches. One common such approach bases on finite-difference approximations of the Hessian. However, as [69] points out, there are better alternatives when dealing with most HMMs. Preferred are profile likelihood-based CIs or bootstrap-based CIs, where the latter are now widely used despite the potentially large computational load [24,28]. With common Poisson HMMs, [2] shows that `TMB` can yield valid Wald-type CIs in a fraction of the time required by classical bootstrap-based methods.

In this section, we describe the Wald-type and the bootstrap-based approach. The likelihood profile method frequently fails to yield CIs (see [2]) and is therefore not detailed.

C.1. Wald-type confidence intervals

Evaluating the Hessian at the optimum found by numerical optimization procedures is basis for Wald-tye CIs. As illustrated above, the optimized negative log-likelihood function of a HMM typically depends on the working parameters. However, drawing inference on the natural parameters is by far more relevant than inference on the working parameters. Since the Hessian $\nabla^2 \log L(\{\hat{\boldsymbol{T}}, \hat{\boldsymbol{\eta}}\})$ relies on the working parameters $\hat{\boldsymbol{\eta}}$, an estimate of the covariance matrix of $\{\hat{\boldsymbol{\Gamma}}, \hat{\boldsymbol{\lambda}}\}$ can be obtained via the delta method through

$$\Sigma_{\hat{\boldsymbol{\Gamma}},\hat{\boldsymbol{\lambda}}} = -\nabla g(\hat{\boldsymbol{T}}, \hat{\boldsymbol{\eta}}) \left(\nabla^2 \log L(\hat{\boldsymbol{T}}, \hat{\eta})\right)^{-1} \nabla g(\hat{\boldsymbol{T}}, \hat{\boldsymbol{\eta}})', \tag{C1}$$

with $\{\hat{\boldsymbol{\Gamma}}, \hat{\boldsymbol{\lambda}}\} = g(\hat{\boldsymbol{T}}, \hat{\boldsymbol{\eta}})$ as defined in [24, Ch. 3.3.1] and [2]. From this, `TMB` deduces the standard errors for the working and natural parameters, from which 95% CIs can be formed via $\hat{\boldsymbol{\lambda}} \pm q_{0.975} \cdot \text{SE}(\hat{\boldsymbol{\lambda}})$ where $q_{0.975} \approx 1.96$ is the 97.5th percentile from the standard normal distribution, and $\text{SE}(\hat{\boldsymbol{\lambda}})$ denotes the standard error of $\hat{\boldsymbol{\lambda}}$.

C.2. Bootstrap-based confidence intervals

The bootstrap method was described by [70] in their seminal article, and is widely used by many practitioners. Many bootstrap techniques have been developed since then, and have been extensively applied in the scientific literature. This paper will not review these techniques, but will instead use one of them: the so-called parametric bootstrap. As [71] points out, the parametric bootstrap is suitable for time series, and hence motivates this choice. For details on the parametric bootstrap implementation in `R`, we refer to [24, Ch. 3.6.2, pp. 58-60].

At its heart, bootstrapping requires some form of re-sampling to be carried out. Then, the chosen model is re-estimated on each new sample, and eventually some form of aggregation of the results takes place. We make use of `TMB` to accelerate the re-estimation procedure, then look at the resulting median along with the 2.5th and 97.5th empirical percentile to infer a 95% CI.

Appendix D. Estimated models for real data sets

Table D1. Estimates, Wald-type 95% CIs, for a two-state Poisson HMM fitted to the TYT data, a two-state Poisson HMM fitted to the soap data, and a three-state Gaussian HMM fitted to the weekly returns data. From left to right, the columns contain: the parameter name, parameter estimate, and lower (L.) and upper (U.) bound of the corresponding 95% CI derived via the Hessian provided by TMB, repeated for the three HMMs. As standard deviations and Poisson rates must be non-negative, the CIs are adjusted when necessary. Similarly, the TPM elements and the stationary distribution must take values between 0 and 1 and the CIs are adjusted accordingly when necessary. AIC and BIC scores are displayed under for corresponding models with two, three, four, and five states (when the model estimation converges). Model estimates are derived by `nlminb` with `TMB`'s gradient and Hessian functions passed as arguments.

TYT data set				Soap data set				Weekly returns data set			
Par.	Est.	L.	U.	Par.	Est.	L.	U.	Par.	Est.	L.	U.
λ_1	1.64	1.09	2.18	λ_1	4.02	3.66	4.38	μ_1	-2.08	-4.3	0.14
λ_2	5.53	4.91	6.16	λ_2	11.37	9.94	12.80	μ_2	0.06	-0.13	0.25
γ_{11}	0.95	0.86	1.00	γ_{11}	0.91	0.86	0.96	μ_3	0.33	0.24	0.41
γ_{12}	0.05	0	0.14	γ_{12}	0.09	0.04	0.14	σ_1	7.11	5.25	8.97
γ_{21}	0.03	0	0.07	γ_{21}	0.37	0.2	0.54	σ_2	2.65	2.47	2.84
γ_{22}	0.97	0.93	1.00	γ_{22}	0.63	0.46	0.80	σ_3	1.41	1.34	1.48
δ_1	0.34	0	0.79	δ_1	0.81	0.71	0.91	γ_{11}	0.98	0.97	0.99
δ_2	0.66	0.21	1.00	δ_2	0.19	0.09	0.29	γ_{12}	0	0	0.00
								γ_{13}	0.02	0.01	0.03
								γ_{21}	0	0	0.00
								γ_{22}	0.81	0.63	0.99
								γ_{23}	0.19	0.01	0.37
								γ_{31}	0.03	0.01	0.04
								γ_{32}	0.01	0	0.02
								γ_{33}	0.96	0.94	0.98
								δ_1	0.42	0.4	0.45
								δ_2	0.55	0.42	0.69
								δ_3	0.02	0	0.17
m	AIC	BIC		m	AIC	BIC		m	AIC	BIC	
2	345	355		2	1245	1259		2	9563	9597	
3	355	377		3	1239	1270		3	9476	9544	
				4	1241	1297		4	9475	9590	

Table D2. Estimates, Wald-type 95% CIs, for a five-state Poisson HMM fitted to the hospital data. From left to right, the columns contain: the parameter name, parameter estimate, and lower (L.) and upper (U.) bound of the corresponding 95% CI derived via the Hessian provided by TMB, repeated for the three HMMs. As standard deviations and Poisson rates must be non-negative, the CIs are adjusted when necessary. Similarly, the TPM elements and the stationary distribution must take values between 0 and 1 and the CIs are adjusted accordingly when necessary. AIC and BIC scores are displayed under for corresponding models with two, three, four, and five states (when the model estimation converges). Model estimates are derived by `nlminb` with `TMB`'s gradient and Hessian functions passed as arguments.

Hospital data set (1)				Hospital data set (2)			
Par.	Est.	L.	U.	Par.	Est.	L.	U.
λ_1	3.54	3.51	3.58	γ_{34}	0	0	0
λ_2	6.46	6.33	6.59	γ_{35}	0	0	0
λ_3	10.13	10.03	10.24	γ_{41}	0	0	0
λ_4	13.96	13.84	14.07	γ_{42}	0	0	0
λ_5	23.67	23.38	23.95	γ_{43}	0.13	0.12	0.13
γ_{11}	0.82	0.81	0.83	γ_{44}	0.87	0.86	0.87
γ_{12}	0	0	0	γ_{45}	0.01	0.01	0.01
γ_{13}	0.03	0.02	0.03	γ_{51}	0	0	0
γ_{14}	0.13	0.12	0.14	γ_{52}	0	0	0
γ_{15}	0.02	0.02	0.02	γ_{53}	0	0	0
γ_{21}	0.28	0.27	0.3	γ_{54}	0.19	0.18	0.21
γ_{22}	0.72	0.7	0.73	γ_{55}	0.8	0.79	0.82
γ_{23}	0	0	0	δ_1	0.24	0.23	0.25
γ_{24}	0	0	0	δ_2	0.15	0.15	0.16
γ_{25}	0	0	0	δ_3	0.28	0.27	0.29
γ_{31}	0	0	0	δ_4	0.29	0.28	0.3
γ_{32}	0.15	0.15	0.16	δ_5	0.04	0.03	0.04
γ_{33}	0.85	0.84	0.85				

m	AIC	BIC
2	518598	518636
3	494824	494908
4	485206	485356
5	481627	481862

Appendix E. Performance and accuracy of different optimizers: additional figures and results

Figs. E1 to E3 show the median of the parameter estimates and the nll, along with 95% CIs from the three different simulation designs described in Section 4 across all optimizers studied, over 1000 realizations from the different models. All figures show that the different optimizers behave almost identically in terms of accuracy. However, we restate that the initial values used here are equal to the true parameter values.

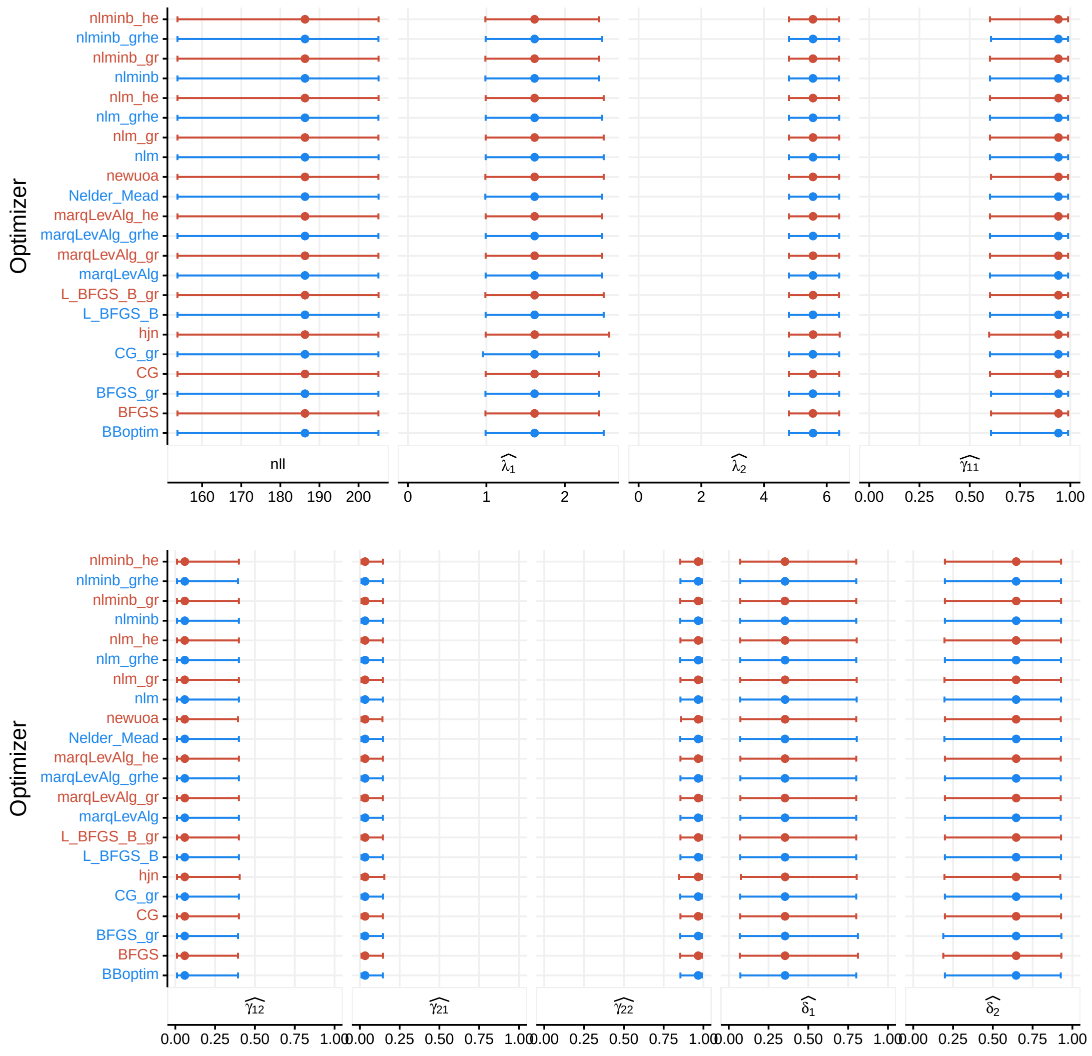

Figure E1. Plots of estimates and NLL when estimating a two-state Poisson HMM on the TYT data set (87 data), over 1000 realizations. The columns display in order the NLL, Poisson rates, TPM elements, and the stationary distribution. The dots represent the medians, and the lines display the 95% percentile CIs.

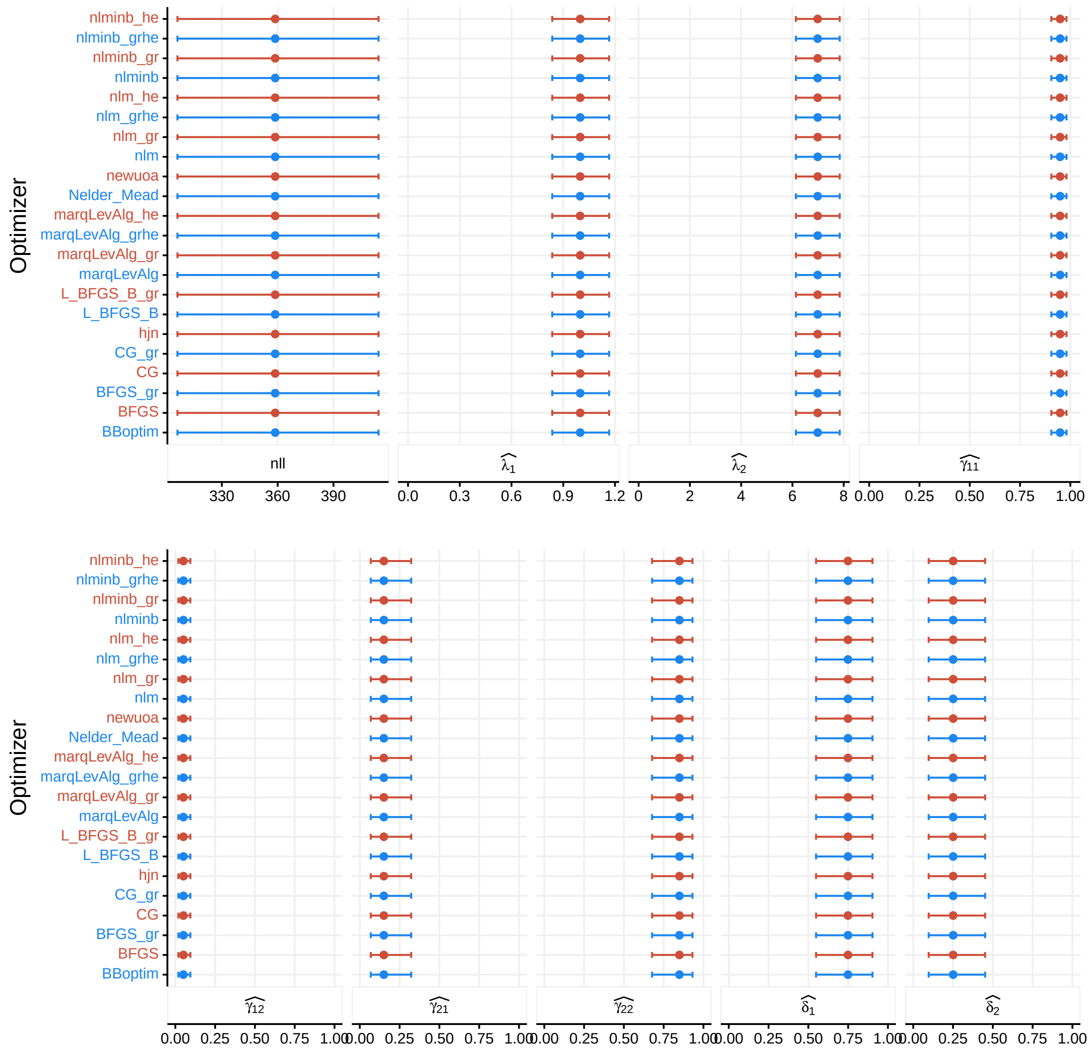

Figure E2. Plots of estimates and NLL when estimating a two-state Poisson HMM in the second study design, with (200 observations), over 1000 realizations. The columns display in order the NLL, Poisson rates, TPM elements, and the stationary distribution. The dots represent the medians, and the lines display the 95% percentile CIs.

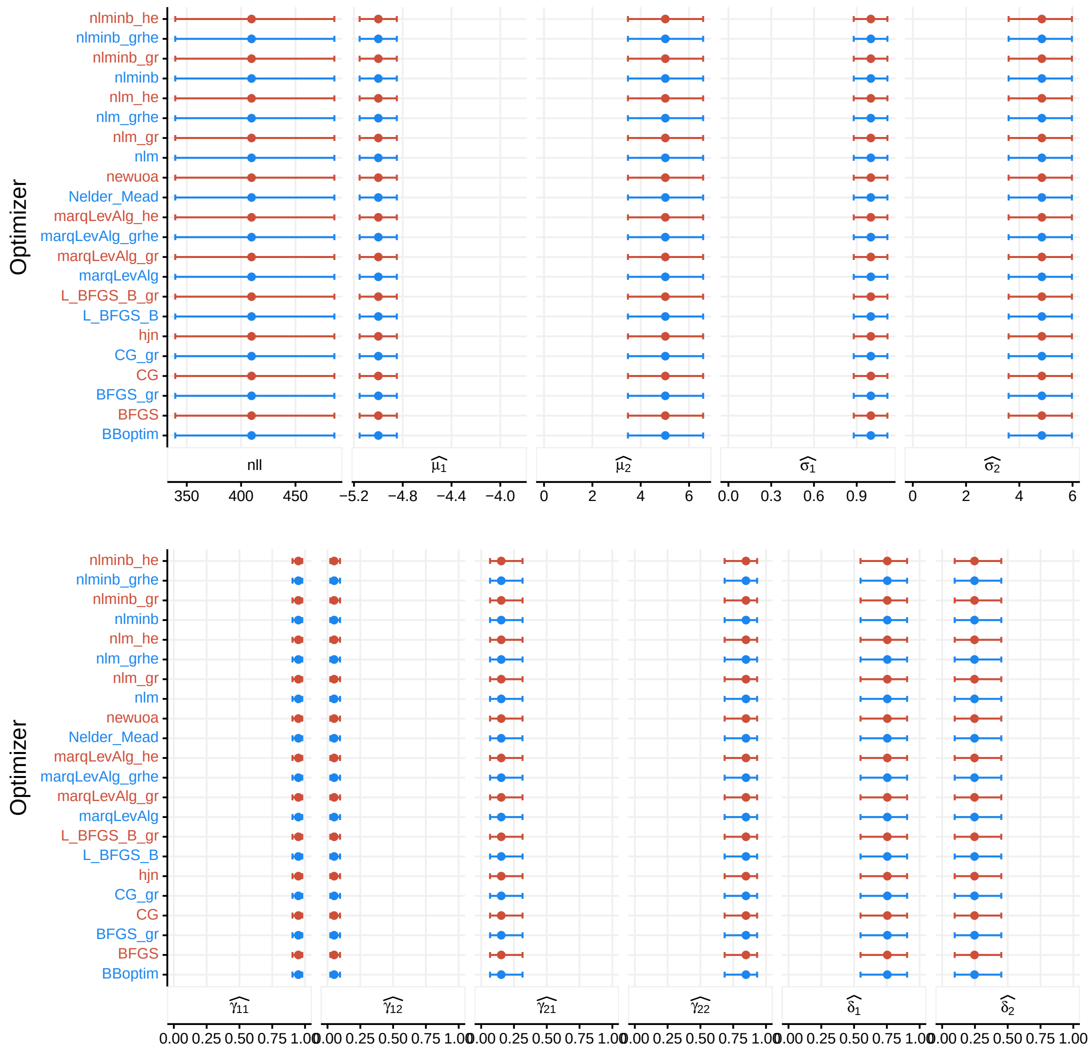

Figure E3. Plots of estimates and NLL when estimating a two-state Poisson HMM in the third study design, with (200 observations), over 1000 realizations. The columns display in order the NLL, Poisson rates, TPM elements, and the stationary distribution. The dots represent the medians, and the lines display the 95% percentile CIs.