Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

journal homepage: www.elsevier.com/locate/csda

hsmm — An **R** package for analyzing hidden semi-Markov models

Jan Bulla[b,*], Ingo Bulla[a], Oleg Nenadić[c]

[a] *Georg-August-Universität Göttingen, Institut für Mikrobiologie und Genetik, Abteilung für Bioinformatik, Goldschmidtstr. 1, 37077 Göttingen, Germany*
[b] *Victoria University of Wellington, School of Mathematics, Statistics and Computer Science, P.O. Box 600, Wellington 6140, New Zealand*
[c] *Georg-August-Universität Göttingen, Institut für Statistik und Ökonometrie, Platz der Göttinger Sieben 5, 37073 Göttingen, Germany*

A R T I C L E   I N F O

Article history:
Available online 27 August 2008

A B S T R A C T

Hidden semi-Markov models are a generalization of the well-known hidden Markov model. They allow for a greater flexibility of sojourn time distributions, which implicitly follow a geometric distribution in the case of a hidden Markov chain. The aim of this paper is to describe hsmm, a new software package for the statistical computing environment **R**. This package allows for the simulation and maximum likelihood estimation of hidden semi-Markov models. The implemented Expectation Maximization algorithm assumes that the time spent in the last visited state is subject to right-censoring. It is therefore not subject to the common limitation that the last visited state terminates at the last observation. Additionally, hsmm permits the user to make inferences about the underlying state sequence via the Viterbi algorithm and smoothing probabilities.

1. Introduction

The Hidden Markov Model (HMM) has become a valuable, well-established tool for the analysis of many different types of sequences of observations. For an overview we refer to the paper of Ephraim and Merhav (2002) and the references therein, as well as to the comprehensive works of MacDonald and Zucchini (1997) and Cappé et al. (2007). However, one weakness of the conventional HMM is the lack of adaptability to different sojourn time (or state duration) distributions, since it is based on a hidden Markov chain whose sojourn times follow a geometric distribution. This is not always desirable and, moreover, limits the range of possible applications (Rabiner, 1989).

The hidden semi-Markov model (HSMM), also referred to as explicit duration HMM or state duration HMM, is a generalization of the HMM that allows one to utilize more general sojourn time distributions. In his pioneering work, Ferguson (1980) introduced a HSMM with non-parametric sojourn time distributions in the field of speech recognition. Since then, the model was further investigated by various authors. Applications include, e.g., speech and pattern recognition (Levinson, 1986; Sin and Kim, 1995), the analysis of branching and flowering patterns, rainfall data, and user request patterns to a Web server (Guédon et al., 2001; Sansom and Thomson, 2001; Yu and Kobayashi, 2003), gene finding (Burge and Karlin, 1997; Lukashin and Borodovsky, 1998), and protein secondary structure prediction (Schmidler et al., 2000). However, despite the different fields of application, there is currently no generally available flexible software that allows the estimation of a HSMM. To our knowledge, the only existing implementation is included in the publicly available program AMAPmod (Godin and Guédon, 2007), which is tailored to specific problems and thus cannot be easily modified. AMAPmod is a sophisticated and specialized statistical software for exploration of a plant architecture database and provides, inter alia, a HSMM-based analysis. However, it is designed to deal with a particular data type and a HSMM suited for this specific application. Therefore, it is not easily extensible towards other applications, in particular not without at least good knowledge of the C++ programming language.

* Corresponding author. Tel.: +64 4 463 5341; fax: +64 4 463 5045.
E-mail addresses: Jan.Bulla@mcs.vuw.ac.nz (J. Bulla), ibulla@uni-goettingen.de (I. Bulla), onenadi@uni-goettingen.de (O. Nenadić).

0167-9473/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2008.08.025

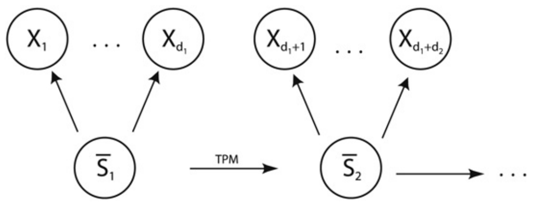

Fig. 1. The hidden semi-Markov model. Basic structure of a hidden semi-Markov model. The arrows from $\bar{S}_1$ to $X_1, \ldots, X_{d_1}$ and from $\bar{S}_2$ to $X_{d_1+1}, \ldots, X_{d_2}$ indicate the conditional dependence of the observations on the hidden states.

The aim of this paper is to present a new software package for the statistical computing environment **R** (R Development Core Team, 2007). The functions contained in the package address three important aspects of the HSMM. Firstly, the simulation of sequences of states and observations given the model specifications (sojourn time and conditional distributions) and parameters. Secondly, maximum likelihood estimation of the model parameters, given a sequence of observation and the model specifications. Thirdly, acquisition of information about the underlying state sequence via the Viterbi algorithm and the smoothing probabilities. With respect to the second point, it is notable that the implemented Expectation Maximization (EM) algorithm is not subject to the common assumption that the last visited state terminates at the last observation. This is an appropriate assumption for many time series and allows for application in various settings.

The paper displays the key features of the package to ensure its accessibility by all researchers interested in the application of HSMMs. We omit the minutiae of the model and refer to Bulla (2006) and the references therein for a theoretical background of the HSMM. The remainder of this paper is organized as follows. Section 2 provides a short introduction to the HSMM. Section 3 presents the `hsmm` package and explains its use by giving examples together with the corresponding **R** code. Section 4 closes with concluding remarks.

2. Hidden semi-Markov models

A HSMM consists of a pair of discrete-time stochastic processes $\{S_t\}$ and $\{X_t\}$. Similar to HMMs, the observed process $\{X_t\}$ is related to the unobserved semi-Markovian state process $\{S_t\}$ by the so-called conditional distributions.

Let $X_1^T := (X_1, \ldots, X_T)$ denote the observed sequence of length T. The same convention is used for the state sequence S_t, and θ denotes the set of model parameters. The state process is a finite-state semi-Markov chain, which is constructed as follows. A homogeneous Markov chain with J states, labeled $1, \ldots, J$, models the transitions between different states. The stochastic process $\{S_t\}$ is specified by the *initial probabilities* $\pi_j := P(S_1 = j)$ with $\sum_j \pi_j = 1$, and the *transition probabilities* p_{ij}. For states $i, j \in \{1, \ldots, J\}$ with $j \neq i$, these are given by

$$p_{ij} := P(S_{t+1} = j | S_{t+1} \neq i, S_t = i)$$

satisfying $\sum_j p_{ij} = 1$, and $p_{ii} = 0$. The diagonal elements of the transition probability matrix (TPM) of a HSMM are required to be zero, since we separately model the *runlength distribution* and do not consider the case of absorbing states. This distribution, also referred to as *sojourn time distribution*, is associated with each state. It models the duration the process $\{S_t\}$ remains in the state j and is defined by

$$d_j(u) := P(S_{t+u+1} \neq j, S_{t+u} = j, \ldots, S_{t+2} = j | S_{t+1} = j, S_t \neq j).$$

The combination of a Markov chain, modeling state changes, and runlength distributions, determining the sojourn times in the states, define $\{S_t\}$ and illustrate the main difference between the HMM and the HSMM. The semi-Markovian state process $\{S_t\}$ of a HSMM does not have the Markov property at each time t, but is Markovian at the times of state changes only.

The observed process $\{X_t\}$ at time t is related to the state process $\{S_t\}$ by the *conditional distributions* $b_j(x_t)$, which are either probability functions in the case of discrete conditional distributions or probability densities in the case of continuous conditional distributions:

$$b_j(x_t) = \begin{cases} P(X_t = x_t | S_t = j) & \text{for discrete } X_t \\ f(X_t = x_t | S_t = j) & \text{for continuous } X_t. \end{cases}$$

For the observation component, the so-called conditional independence property is fulfilled:

$$P(X_t = x_t | X_1^T = x_1^T, S_1^{t-1} = s_1^{t-1}, S_t = j, S_{t+1}^T = s_{t+1}^T) = P(X_t = x_t | S_t = j),$$

that is, the output process at time t depends only on the value of S_t.

Fig. 1 visualizes the general setup of a HSMM without further assumption on the runlength or conditional distributions. The state sequence $\overline{S}_t$ represents the embedded first-order Markov chain. The initial state $\overline{S}_1$ is chosen according to the initial distribution $\vec{\pi} = (\pi_j)_{j \in \{1,\ldots,J\}}$. The sojourn time in the first visited state, d_1, is generated by the respective sojourn time distribution. The first d_1 observations $X_1, \ldots, X_{d_1}$ are sampled from the respective conditional distribution linked to value of $\overline{S}_1$. Then, the state change is modeled according to the TPM.

Table 1
Specification of the observation distribution and the runlength distribution

Observation distribution			Runlength distribution		
Distribution	R-name	`od.par`	Distribution	R-name	`rd.par`
Bernoulli	`"bern"`	`b`	'Non-parametric'	`"nonp"`	`np`
Normal	`"norm"`	`mean, var`	Geometric	`"geom"`	`p`
Poisson	`"pois"`	`lambda`	Negative Binomial	`"nbinom"`	`r, pi`
Student-t	`"t"`	`mean, var, df`	Logarithmic	`"log"`	`p`
			Poisson	`"pois"`	`lambda`

The underlying semi-Markovian state sequence $\{S_t\}$ is constructed from the embedded Markov chain and the sojourn time distributions by setting $S_1 = \bar{s}_1, \ldots, S_{d_1} = \bar{s}_1, S_{d_1+1} = \bar{s}_2, \ldots, S_{d_1+d_2} = \bar{s}_2, \ldots$ (and vice versa).

3. The `hsmm` package

The package `hsmm` provides tools for performing HSMM analyses, which are commonly required when working with this model. The main requirements are

- the simulation of sequences of states and observations (Section 3.1),
- the estimation of model parameters (Section 3.2), and
- the analysis of the underlying state sequence (Section 3.3).

Due to the flexibility of HSMMs, the corresponding functions contain a number of arguments. As is the case with simple HMMs, the model specification includes a transition probability matrix, the initial probabilities and the conditional distributions of the observations, which are hereafter referred to as the observation distributions. In addition, the sojourn time distributions, which are referred to as the runlength distributions within the package, need to be specified. Table 1 displays the currently implemented distributions for the observations and the runlengths.

3.1. Simulation of observation and state sequences

To obtain a first understanding of the nature and properties of HSMMs, the simulation of sequences of states and observations is a helpful tool. For given model specifications, this is carried out by the function `hsmm.sim()`. The **R** commands given below generate a sequence of length $n = 2000$ from a HSMM with Gaussian observation distributions, logarithmic runlength distributions, and three hidden states.

```
> pipar  <- rep(1/3, 3)
> tpmpar <- matrix(c(0, 0.5, 0.5,
+                    0.7, 0, 0.3,
+                    0.8, 0.2, 0), 3, byrow = TRUE)
> rdpar  <- list(p = c(0.98, 0.98, 0.99))
> odpar  <- list(mean = c(-1.5, 0, 1.5), var = c(0.5, 0.6, 0.8))
> sim    <- hsmm.sim(n = 2000, od = "norm", rd = "log",
+                    pi.par = pipar, tpm.par = tpmpar,
+                    rd.par = rdpar, od.par = odpar, seed = 3539)
```

The vector *pi.par* contains the initial values for the initial probabilities of the semi-Markov chain. The matrix *tpm.par* provides the initial values for the TPM (probabilities of transitions with the same source state are given row-wise). The two arguments *od* and *rd* are used for specifying the observation distribution and the runlength distribution, respectively. In this example, the observation distribution is given by a Gaussian distribution (*od="norm"*) and the runlength is given by a logarithmic distribution (*rd="log"*). The parameters of the observation and runlength distributions are specified as a list by the options *rd.par* and *od.par*, respectively. For example, if the observation distribution is given by a Gaussian distribution, the corresponding list for *od.par* requires the entries *mean* and *var*. Table 1 gives an overview of the implemented distributions.

The function `hsmm.sim()` returns a list containing the simulated sequence of observations and the simulated state sequence. The simulated observations and states are accessed with

```
> round(sim$obs[1:15], 3)
 [1] -0.335  1.263 -2.104 -1.518 -0.375 -0.465  1.454 -2.229
 [9] -1.187 -1.264 -1.765 -1.529 -2.177 -2.868 -1.701
> sim$path[1:15]
 [1] 2 2 1 2 2 2 2 1 1 1 1 1 1 1 1
```

Fig. 2 displays the simulated series. The upper panel shows the sequence of observations $\{X_t\}$ together with their conditional expected value $E(X_t|S_t = j)$, which corresponds to the mean of the respective Gaussian distribution. The lower panel shows the state process $\{S_t\}$.

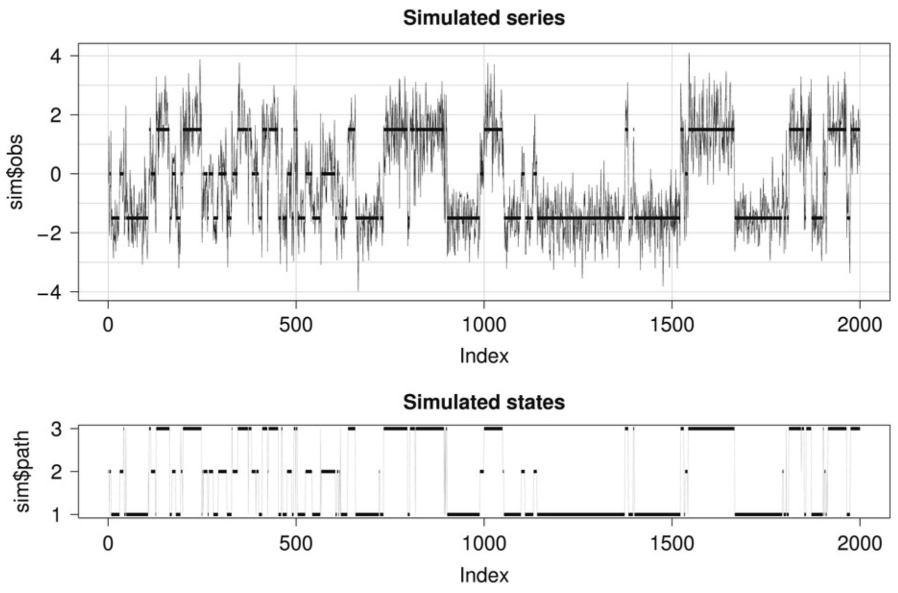

Fig. 2. Simulated observations and states and predicted states. The upper panel displays the simulated sequence of observations. The horizontal lines represent the conditional means (−1.5, 0, 1.5) of the respective underlying state. The lower panel shows the simulated state process.

For further details on the use of `hsmm.sim`, just as the functions introduced below, we refer to the reference manual. The corresponding part of the manual is called by typing `?function_name` in R.

3.2. *Maximum likelihood estimation of the model parameters*

The algorithms for parameter estimation in the package `hsmm` are based on the right-censored HSMM introduced by Guédon (2003). This model does not assume that the last observation coincides with an exit from the last visited state, which is desirable for many applications. However, the state sequence starts simultaneously with the first observation. If the observations and the underlying state sequence were both known, the likelihood of a HSMM could be easily evaluated. However, this is not the case in general. Hence, we are confronted with a missing data problem. A common way of dealing with this type of problem is to use the EM algorithm (Baum et al., 1970; Dempster et al., 1977). After assigning initial values to the parameters, the EM algorithm is implemented by successively iterating the so-called E- and M-step. This iterative procedure increases the likelihood monotonically until a stationary point is reached. Details on the algorithm utilized for the package `hsmm` can be found in Guédon (2003) and Bulla (2006).

The estimation is carried out by the function `hsmm()`. The arguments for this function are similar to `hsmm.sim()` with the only difference being in the parameter specification (i.e. the arguments ending on *`.par`*). In the case of `hsmm()`, the arguments *`pi.par`*, *`tpm.par`*, *`rd.par`*, *`od.par`* specify the starting values for the parameter estimation. Thus, the overview given for the observation and the runlength distributions in Table 1 also applies to `hsmm()`.

The following example illustrates the usage of `hsmm()` for parameter estimation.

```
> pipar  <- rep(1/3, 3)
> tpmpar <- matrix(c(0, 0.5, 0.5,
+                    0.7, 0, 0.3,
+                    0.8, 0.2, 0), 3, byrow = T)
> rdpar  <- list(p = c(0.98, 0.98, 0.99))
> odpar  <- list(mean = c(-1.5, 0, 1.5), var = c(0.5, 0.6, 0.8))
> fit    <- hsmm(sim$obs, od = "norm", rd = "log",
+                pi.par = pipar, tpm.par = tpmpar,
+                od.par = odpar, rd.par = rdpar)
```

The function `hsmm()` returns a list containing the output. For example, the observed data log likelihood is returned in the `logl` entry:

```
> fit$logl
[1] -2671.79
```

The estimated parameters are given in the `para` entry, where each component (TPM, runlength distribution and observation distribution) is given in a sub-list. For example, the estimated parameters for the observation distribution are given by

```
> fit$para$od
$mean
[1] -1.54168985  0.08857843  1.52942672

$var
[1] 0.4780348 0.6139798 0.7984119
```

In the case of the Gaussian distribution, there are two entries. For demonstration purpose, we chose the true parameters as initial values.

However, in almost all cases the true parameter values are unknown and the user may only have, if at all, little intuition as to the selection of the exact number of states, the runlength and observation distribution, and the initial values. In the following, we provide brief guidance to these aspects.

The choice of "good" initial values is not an exceedingly crucial point because stability of the EM algorithm in terms of convergence to the global maximum is one of the key features of this estimation technique (for the HMM case, see Bulla and Berzel (2008) and Hathaway (1986)). For HSMMs with a low number states, we also noted similar properties. For example, Bulla and Bulla (2006) present an application to return series and estimate two-state models with negative binomial runlength and Gaussian/Student t observation distribution. Exploring the effect of different initial values by grid searches in this context, we discovered a stable convergence to the global maximum, except when very poor initial guesses were used. Nevertheless, our personal experience shows that starting the algorithm with different initial values is often worthwhile when fitting models inheriting a high number of states and flexible distributions. Unfortunately, there is no generally accepted approach to guide the user in such a setup. If a content- or data-driven choice of the initial values is not possible, doubtless, a grid search still is the safest approach. However, this technique may become rather time-consuming for complex models with a high number of parameters. Alternatively, randomly generated initial values (within reasonable bounds) are worthwhile to be explored.

The choice of the number of states, the runlength and the observation distribution is a different problem and falls into the model selection category. For the number of states, a vast literature exists in the HMM case, the presentation of which would go beyond the scope of this article. As many arguments are also valid for HSMMs, we would like to refer the interested reader to Cappé et al. (2007) and MacDonald and Zucchini (1997). In our personal opinion, the model selection criteria AIC and BIC provide good first guidance, apart from data-driven arguments.

Likewise, the choice of observation and runlength distribution may be checked with the same criteria and, additionally, pseudo-residuals help to assess the model fit to the observations (Zucchini and MacDonald, 1999). However, the selection of the most appropriate runlength distribution is a complicated problem. Comparison of the empirical sojourn times – estimated, e.g., by the function `hsmm.smooth` – with the theoretical distribution can help to detect an unsatisfactory fit (personal communication with Y. Guédon). If the user has no idea at all with respect to the runlength distribution, Sansom and Thomson (2001) present a procedure to derive the general shape of the distribution. In a first step, they assume a non-parametric runlength distribution. In a second step, they examine the fitted distribution and consider parametric alternatives.

The non-parametric runlength distribution is implemented as default distribution in the package and initialized by a uniform distribution on the first ten runlengths. Similarly, the initial probabilities for `pi` follow a uniform distribution. The default model has two states, thus the TPM does not require any initial values. For models with $J > 2$ states, the TPM may be initialized by the value $1/(J-1)$ for the off-diagonal elements. However, we would like to note that the non-parametric runlength distribution often requires a high number of observations. For example, Sansom and Thomson (2001) worked with series of length 20000 and greater.

3.3. Inference on the underlying state sequence

A natural question arising in the context of many applications is the following. Given the estimated HSMM and the observations, which are the most likely states of the underlying Markov chain? This investigation of the state sequence is called decoding. The most popular methods are the Viterbi algorithm, a dynamic programming algorithm, and the smoothing probabilities. The Viterbi algorithm performs a global decoding. It determines the most probable state sequence given the observations, i.e.,

$$\underset{j_1,\ldots,j_T}{\operatorname{argmax}}\, P(S_1 = j_1, \ldots, S_T = j_T | X_1^T = x_1^T).$$

The smoothing probabilities, however, carry out local decoding by calculating the probability of visiting state j at time t for given observations,

$$P(S_t = j | X_1^T = x_1^T), \quad t \in \{1, \ldots, T\},\ j \in \{1, \ldots, J\}.$$

Both methods are implemented as the functions `hsmm.viterbi()` and `hsmm.smooth()`. It should be noted that the syntax of these two functions follows the same scheme as in `hsmm()`. The following example illustrates the corresponding functions.

Table 2
Functions in the hsmm package

Core functions	Auxiliary functions	C++ functions
hsmm.sim	get.d	FB
hsmm	get.pdf	Viterbi
hsmm.smooth	⋮	
hsmm.viterbi		
S3-methods for hsmm-class		

```
> fit.vi <- hsmm.viterbi(sim$obs, od = "norm", rd = "log",
+                          pi.par = pipar, tpm.par = tpmpar,
+                          od.par = odpar, rd.par = rdpar)
> fit.sm <- hsmm.smooth(sim$obs, od = "norm", rd = "log",
+                         pi.par = pipar, tpm.par = tpmpar,
+                         od.par = odpar, rd.par = rdpar)
> fit.vi$path[1:15]
 [1] 2 2 1 1 1 1 3 1 1 1 1 1 1 1 1
> fit.sm$path[1:15]
 [1] 2 2 1 1 2 2 3 1 1 1 1 1 1 1 1
> round(fit.sm$sm[, 1:15], 2)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 0.19  0.0 0.93 0.87 0.47 0.37 0.01 0.99 0.99  0.99
[2,] 0.64  0.5 0.07 0.13 0.51 0.57 0.42 0.01 0.01  0.01
[3,] 0.17  0.5 0.00 0.00 0.02 0.06 0.58 0.00 0.00  0.00
     [,11] [,12] [,13] [,14] [,15]
[1,]     1     1     1     1     1
[2,]     0     0     0     0     0
[3,]     0     0     0     0     0
```

Fig. 3 gives an overview of the observations, the decoded states and the smoothing probabilities. The top panel presents the observations $\{X_t\}$. The second and third panel from the top present the state sequence estimated by the Viterbi algorithm and the smoothed probabilities, respectively. The panel at the bottom displays the smoothing probabilities for each of the three states. A probability is plotted in black color if it is the maximum of the three smoothing probabilities and in gray color otherwise.

3.4. Structure of the hsmm Package

The previous sections show that the package comprises four core functions for simulating, fitting and decoding of hidden semi-Markov models. In this section we further illustrate the underlying architecture of the package. Table 2 gives a brief overview of the functions provided with the package.

The core functions discussed in the previous sections are hsmm.sim, hsmm, hsmm.smooth, and hsmm.viterbi. In order to provide an appropriate formatting of the results, S3 methods are implemented into the package. For efficiency reasons, computer intensive procedures are implemented in C++. The relevant part for the estimation and smoothing procedure is the forward–backward algorithm in 'FB' while the Viterbi algorithm is implemented in 'Viterbi'.

Function calls to the runlength and the conditional distribution have been outsourced to the two auxiliary functions, get.d for $d_j(u)$ and get.pdf for $b_j(x_t)$. In this connection it has to be noted that the extension of the package to cover further distributions requires some additional work with the core function hsmm. While the E-step of the EM algorithm does only depend on the auxiliary functions get.d and get.pdf, more difficulties arise with the M-step. In the maximization step, the next set of parameters has to be determined by maximizing the Q-function, whose form changes severely with the distributional assumptions. The function hsmm takes account of this step internally and thus extensions have to cover this part as well.

3.5. Comparison to other software packages

Other than hsmm, the only existing software package that includes algorithms on HSMMs is AMAPmod (Godin and Guédon, 2007). In order to compare the results of hsmm and AMAPmod, we generated a test sequence of length 5000 from a two-state HSMM with Bernoulli distributed observations and negative binomial runlength distributions, because this model is analyzable by both packages. The commands are similar to the ones described in Section 3.1, except for the argument *seed = 14532*. Table 3 summarizes the true parameter values, the estimation results from the two packages, and empirical quantiles from a simulation study.

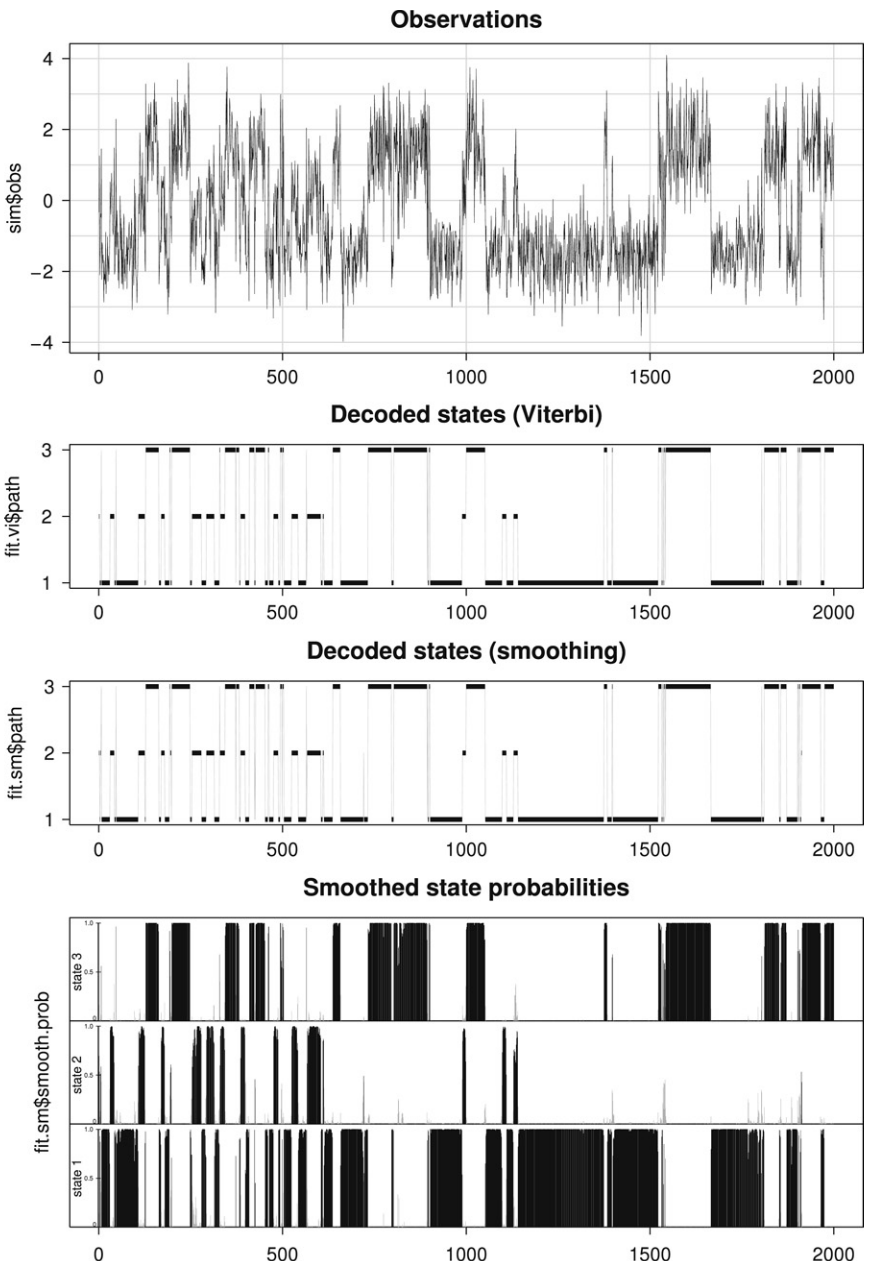

Fig. 3. Simulated observations and states and predicted states. The upper panel displays the sequence of observations. The second and third panel show the state sequence estimated by the Viterbi algorithm and the smoothed probabilities, respectively. The lower panel shows the smoothed probabilities (black if it is the maximum probability and gray otherwise).

Table 3
Parameter estimates of `hsmm` and `AMAPmod`

Parameter	Runlength dist.				Obs. dist.	
	r_1	r_2	pi_1	pi_2	b_1	b_2
True value	1	4	0.1	0.2	0.95	0.05
Estimate AMAPmod	0.888	3.95	0.0787	0.187	0.949	0.0562
Estimate hsmm	0.953	4.01	0.0841	0.190	0.949	0.0559
2.5% - Quantile	0.723	2.84	0.0747	0.146	0.937	0.0390
97.5% - Quantile	1.443	5.88	0.1385	0.275	0.962	0.0610

Both packages estimate the parameter values close to the true values. The minor differences between `hsmm` and `AMAPmod` are caused by small numerical inaccuracies, which result from the application of a form of rounding procedure in `AMAPmod`.

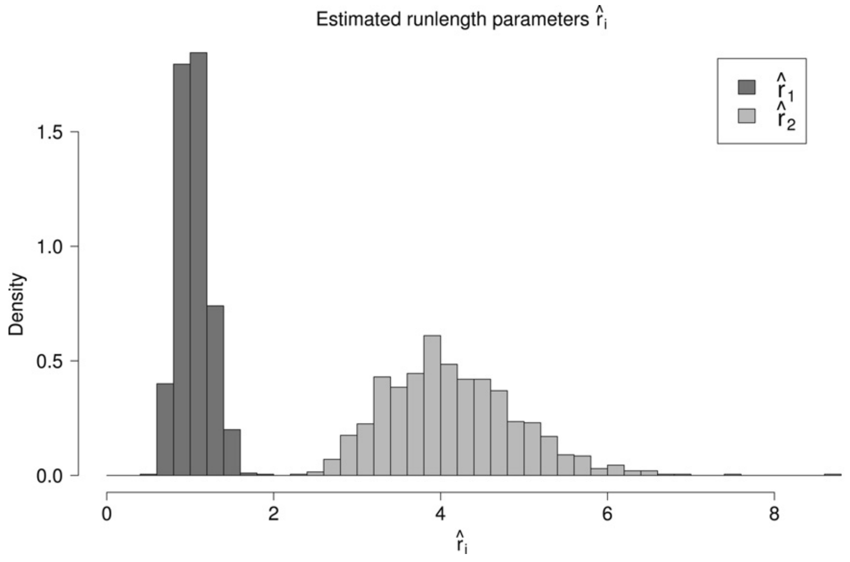

Fig. 4. The estimated runlength parameters $\hat{r}_i$, $i \in \{1, 2\}$ from 1000 simulated series. The figure displays histograms of the parameter estimates for r_i, $i \in \{1, 2\}$ of the negative binomial runlength distribution.

Table 4
Parameter estimates of `hsmm` and `HiddenMarkov`

Parameter	Runlength dist.		Obs. dist.	
	p_1	p_2	b_1	b_2
True value	-	-	0.95	0.05
Estimate `hsmm`	0.09478483	0.05861726	0.95657677	0.05647321
Estimate `HiddenMarkov`	0.09478474	0.05861718	0.95657679	0.05647333

This approach limits the length of a tail, typically with very small probability masses, and thus considerably speeds up the convergence of the EM algorithm. The log likelihoods of −2005.42 and −2005.56 for `hsmm` and `AMAPmod`, respectively, show that the algorithm of `hsmm` attained somewhat closer to the maximum, although this might be negligible for most applications.

In order to investigate the variability of the parameter estimates we generated 1000 sequences of length 5000 with the true parameters given in Table 3. The resulting empirical quantiles (2.5% and 97.5%) for the estimates are given in the last two rows of Table 3. Considering the variability, the deviations of the estimated parameters from the true values seem well within the range. Fig. 4 depicts the distribution of the parameter estimates $\hat{r}_1$ and $\hat{r}_2$.

Furthermore, we carried out a comparison to the `R` package `HiddenMarkov` (Harte, 2008), which allows for HMM estimation. Table 4 presents the estimation results of a HSMM with geometric runlength distribution and `HiddenMarkov` using the same data as for the comparison of `hsmm` and `AMAPmod`. The log likelihood is −2029.918 in both cases. The packages obtain almost identical parameter estimates, which indicates a good numerical accuracy of `hsmm`.

4. Concluding remarks

In this paper we have presented `hsmm`, an **R** package for simulation, inference and state estimation of HSMMs. The package is the first open source software of this kind that allows one to analyze HSMMs within a standard software package. It provides interested researchers and practitioners facilities for working with the computationally complex HSMM within a commonly available, comprehensive and flexible statistical computing system.

In the current implementation, `hsmm` offers a variety of sojourn time and conditional distributions. Although limited, the range of available distributions is very wide and should meet most users' requirements in practice. However, the algorithm can be easily extended. The computationally intensive part, written in `C++`, is independent of the selected distributions. Extension to include other distributions requires minor modifications to the **R** code. The biggest challenge may lie in the parameter re-estimation procedure, the so-called M-step of the EM algorithm. This part of the EM algorithm often entails the optimization of complex functions and therefore requires care and attention.

The Comprehensive R Archive Network http://cran.r-project.org/ provides the package, including sources, binaries and documentation, for download under the GNU Public License. The demonstration called by `demo(hsmm)` contains, inter alia, the examples described in this paper.

Acknowledgments

We would like to sincerely thank Daniel Adler, David Harte, John Sansom, Peter Thomson, and David Vere-Jones for their inspiring comments, helpful discussions and support. Special thanks go to the main author of AMAPmod, Yann Guédon, who kindly provided us with estimation results from his package. Not to forget, we thank Gisela and Klaus Bulla for editorial assistance. Jan Bulla was supported by a fellowship within the Postdoc-Program of the German Research Foundation (DFG).

References

Baum, L.E., Petrie, T., Soules, G., Weiss, N., 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann. Math. Statist. 41, 164–171.

Bulla, J., 2006. Application of hidden markov and hidden semi-markov models to financial time series. Ph.D. Thesis, Institute for Statistics and Econometrics, Georg-August-Universität Göttingen, http://webdoc.sub.gwdg.de/diss/2006/bulla/.

Bulla, J., Berzel, A., 2008. Computational issues in parameter estimation for stationary hidden Markov models. Comput. Statist. 23 (1), 1–18.

Bulla, J., Bulla, I., 2006. Stylized facts of financial time series and hidden semi-Markov models. Comput. Statist. Data Anal. 51 (4), 2192–2209.

Burge, C., Karlin, S., 1997. Prediction of complete gene structures in human genomic DNA. J. Mol. Biol. 268 (1), 78–94.

Cappé, O., Moulines, E., Ryden, T., 2007. Inference in Hidden Markov Models. In: Springer Series in Statistics, Springer–Verlag, New York - Heidelberg - Berlin.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Statist. Soc. Ser. B 39 (1), 1–38. with discussion.

Ephraim, Y., Merhav, N., 2002. Hidden Markov processes, Shannon Theory: Perspective, Trends, and Applications. In: IEEE Trans. Inform. Theory 48 (6) 1518–1569 (special issue).

Ferguson, J.D., 1980. Variable duration models for speech. In: Proceedings of the Symposium on the Applications of Hidden Markov Models to Text and Speech. Princeton, New Jersey, pp. 143–179.

Godin, C., Guédon, Y., 2007. AMAPmod and reference manual. UMR CIRAD/INRA AMAP, Montpellier, France. http://amap.cirad.fr/amapmod/refermanual15/accueil.html.

Guédon, Y., 2003. Estimating hidden semi-Markov chains from discrete sequences. J. Comput. Graph. Statist. 12 (3), 604–639.

Guédon, Y., Barthélémy, D., Caraglio, Y., Costes, E., 2001. Pattern analysis in branching and axillary flowering sequences. J. Theor. Biol. 212 (4), 481–520.

Harte, D., 2008. Reference Manual Package ‘HiddenMarkov’. Statistics Research Associates Limited, Wellington, New Zealand, http://homepages.paradise.net.nz/david.harte/SSLib/#HiddenMarkov.

Hathaway, R.J., 1986. A constrained em algorithm for univariate normal mixtures. J. Stat. Comput. Simul. 23, 211–230.

Levinson, S.E., 1986. Continuously variable duration hidden Markov models for automatic speech recognition. Comput. Speech Lang. 1, 29–45.

Lukashin, A.V., Borodovsky, M., 1998. Genemark.hmm: New solutions for gene finding. Nucleic Acids Res. 26 (4), 1107–1115.

MacDonald, I.L., Zucchini, W., 1997. Hidden Markov and other models for discrete-valued time series. In: Monographs on Statistics and Applied Probability, vol. 70. Chapman & Hall, London.

R Development Core Team, 2007. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org.

Rabiner, L., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. IEEE Trans. Inform. Theory 77 (2), 257–284.

Sansom, J., Thomson, P.J., 2001. Fitting hidden semi-Markov models to breakpoint rainfall data. J. Appl. Probab. 38A, 142–157.

Schmidler, S.C., Liu, J.S., Brutlag, D.L., 2000. Bayesian segmentation of protein secondary structure. J. Comput. Biol. 7 (1–20), 233–248.

Sin, B., Kim, J.H., 1995. Nonstationary hidden Markov model. Signal Process. 46 (1), 31–46.

Yu, S.-Z., Kobayashi, H., 2003. An efficient forward–backward algorithm for an explicit-duration hidden Markov model. IEEE Signal Process. Lett. 10 (1), 11–14.

Zucchini, W., MacDonald, I.L., 1999. Illustrations of the use of pseudo-residuals in assessing the fit of a model. In: Friedl, H., Berghold, A., Kauermann, G. (Eds.), Proceedings of the 14th International Workshop on Statistical Modelling, vol. 3. Graz, Austria, pp. 409–416.