Multi-state modelling with R: the `msm` package

Version 0.6.2

Christopher Jackson
Department of Epidemiology and Public Health
Imperial College, London

chris.jackson@imperial.ac.uk

Abstract

The multi-state Markov model is a useful way of describing a process in which an individual moves through a series of states in continuous time. The *msm* package for R allows a general multi-state model to be fitted to longitudinal data. Data often consist of observations of the process at arbitrary times, so that the exact times when the state changes are unobserved. For example, the progression of chronic diseases is often described by stages of severity, and the state of the patient may only be known at doctor or hospital visits. Features of *msm* include the ability to model transition rates and hidden Markov output models in terms of covariates, and the ability to model data with a variety of observation schemes, including censoring.

Hidden Markov models, in which the true path through states is only observed through some error-prone marker, can also be fitted. The observation is generated, conditionally on the underlying states, via some distribution. An example is a screening misclassification model in which states are observed with error. More generally, hidden Markov models can have a continuous response, with some arbitrary distribution, conditionally on the underlying state.

This manual introduces the theory behind multi-state Markov and hidden Markov models, and gives a tutorial in the typical use of the *msm* package, illustrated by some typical applications to modelling chronic diseases.

Contents

1 Multi-state models

1.1 Introduction

Figure 1 illustrates a multi-state model in continuous time. Its four states are labelled **1, 2, 3, 4**. At a time t the individual is in state $S(t)$. The arrows show which transitions are possible between states. The next state to which the individual moves, and the time of the change, are governed by a set of *transition intensities* $q_{rs}(t, z(t))$ for each pair of states r and s. The intensities may also depend on the time of the process t, or more generally a set of individual-specific or time-varying explanatory variables $z(t)$. The intensity represents the instantaneous risk of moving from state r to state s:

$$q_{rs}(t, z(t)) = \lim_{\delta t \to 0} P(S(t + \delta t) = s | S(t) = r)/\delta t \tag{1}$$

The intensities form a matrix Q whose rows sum to zero, so that the diagonal entries are defined by $q_{rr} = -\sum_{s \neq r} q_{rs}$. To fit a multi-state model to data, we estimate this transition intensity matrix. We concentrate on *Markov* models here. The Markov assumption is that future evolution only depends on the current state. That is, $q_{rs}(t, z(t), \mathcal{F}_t)$ is independent of $\mathcal{F}_t$, the observation history $\mathcal{F}_t$ of the process up to the time preceding t. See, for example, Cox and Miller[**?**] for a thorough introduction to the theory of continuous-time Markov chains.

1.2 Disease progression models

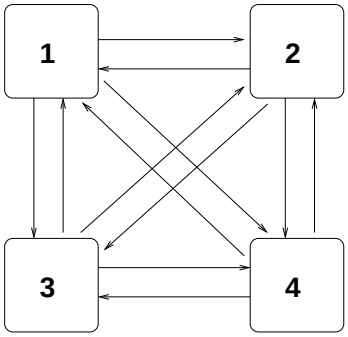
The development of the *msm* package was motivated by applications to disease modelling. Many chronic diseases have a natural interpretation in terms of staged progression. Multi-state Markov models in continuous time are often used to model the course of diseases. A commonly-used model is illustrated in Figure 2. This represents a series of successively more severe disease stages, and an 'absorbing' state, often death. The patient may advance into or recover from adjacent disease stages, or die at any disease stage. Observations of the state $S_i(t)$ are made on a number of individuals i at arbitrary times t, which may vary between individuals. The stages of disease may be modelled as a homogeneous continuous-time Markov process, with a transition matrix Q, pictured below Figure 2.

A commonly-used model is the *illness-death* model, with three states representing health, illness and death (Figure 3). Transitions are permitted from health to illness, illness to death and health to death. Recovery from illness to health is sometimes also considered.

A wide range of medical situations have been modelled using multi-state methods, for example, screening for abdominal aortic aneurysms (Jackson *et al.*[**?**]), problems following lung transplantation (Jackson and Sharples[**?**]), problems following heart transplantation (Sharples[**?**], Klotz and Sharples[**?**]), hepatic cancer (Kay[**?**]), HIV infection and AIDS (Longini *et al.*[**?**], Satten and Longini[**?**], Guihenneuc-Jouyaux *et al.*[**?**], Gentleman *et al.*[**?**]), diabetic complications (Marshall and Jones[**?**], Andersen[**?**]), breast cancer screening (Duffy and Chen[**?**], Chen *et al.*[**?**]), cervical cancer screening (Kirby and Spiegelhalter[**?**]) and liver cirrhosis (Andersen *et al.*[**?**]). Many of these references also describe the mathematical theory, which will be reviewed in the following sections.

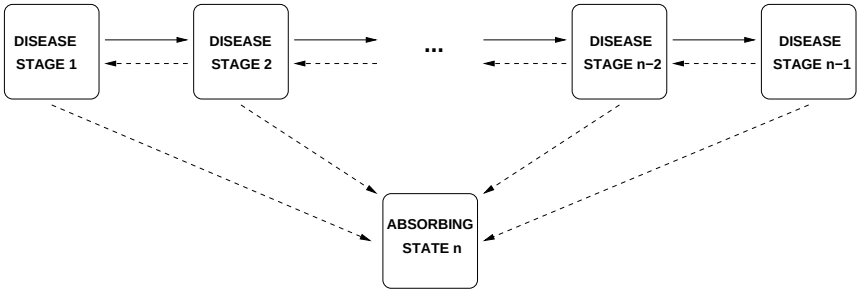
1.3 Arbitrary observation times

Longitudinal data from monitoring disease progression are often incomplete in some way. Usually patients are seen at intermittent follow-up visits, at which monitoring information is collected, but



$$Q = \begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{pmatrix}$$

Figure 1: General multi-state model.



$$Q = \begin{pmatrix} q_{11} & q_{12} & 0 & 0 & \dots & q_{1n} \\ q_{21} & q_{22} & q_{23} & 0 & \dots & q_{2n} \\ 0 & q_{32} & q_{33} & q_{34} & \ddots & q_{3n} \\ 0 & 0 & q_{43} & q_{44} & \ddots & q_{4n} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

Figure 2: General model for disease progression.

information from the periods between visits is not available. Often the exact time of disease onset is unknown. Thus, the changes of state in a multi-state model usually occur at unknown times. Also a subject may only be followed up for a portion of their disease history. A fixed observation schedule may be specified in advance, but in practice times of visits may vary due to patient and hospital pressures. The states of disease progression models often include death. Death times are commonly recorded to within a day. Also observations may be censored. For example, at the end of a study, an individual may be known only to be alive, and in an unknown state.

A typical sampling situation is illustrated in Figure 4. The individual is observed at four occasions through 10 months. The final occasion is the death date which is recorded to within a day. The only other information available is the occupation of states 2, 2, and 1 at respective times 1.5, 3.5 and 5. The times of movement between states and the state occupancy in between the observation times are unknown. Although the patient was in state 3 between 7 and 9 months this was not observed at all.

Informative sampling times To fit a model to longitudinal data with arbitrary sampling times we must consider the reasons why observations were made at the given times. This is analogous to the problem of missing data, where the fact that a particular observation is missing may implicitly give information about the value of that observation. Possible observation schemes include:

- *fixed*. Each patient is observed at fixed intervals specified in advance.
- *random*. The sampling times vary randomly, independently of the current state of the disease.
- *doctor's care*. More severely ill patients are monitored more closely. The next sampling time is chosen on the basis of the current disease state.
- patient *self-selection*. A patient may decide to visit the doctor on occasions when they are in a poor condition.

Grüger *et al.* [**?**] discussed conditions under which sampling times are *informative*. If a multi-state model is fitted, ignoring the information available in the sampling times, then inference may be biased. Mathematically, because the sampling times are often themselves random, they should be modelled along with the observation process X_t. However the ideal situation is where the joint likelihood for the times and the process is proportional to the likelihood obtained if the sampling times were fixed in advance. Then the parameters of the process can be estimated independently of the parameters of the sampling scheme.

In particular, they showed that fixed, random and doctor's care observation policies are not informative, whereas patient self-selection is informative.

1.4 Likelihood for the multi-state model

Kalbfleisch and Lawless[**?**] and later Kay [**?**] described a general method for evaluating the likelihood for a general multi-state model in continuous time, applicable to any form of transition matrix. The only available information is the observed state at a set of times, as in Figure 4. The sampling times are assumed to be non-informative.

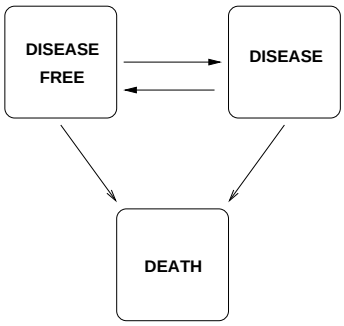


Figure 3: Illness-death model.

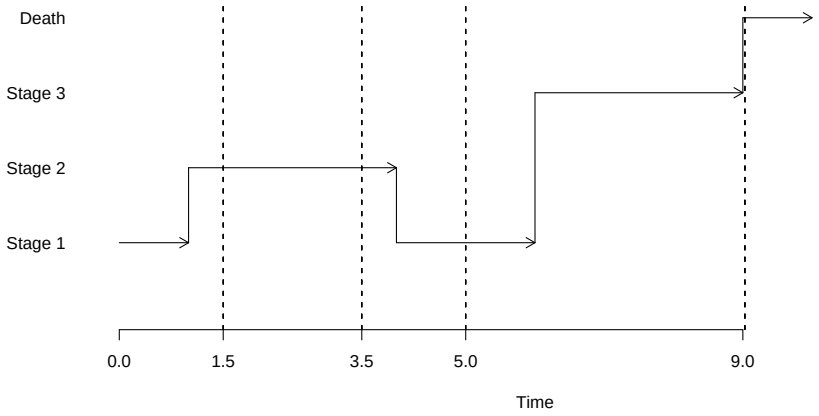


Figure 4: Evolution of a multi-state model. The process is observed on four occasions.

Transition probability matrix The likelihood is calculated from the *transition probability matrix* $P(t)$. For a time-homogeneous process, the (r, s) entry of $P(t)$, $p_{rs}(t)$, is the probability of being in state s at a time $t + u$ in the future, given the state at time u is r. It does not say anything about the time of transition from r to s, indeed the process may have entered other states between times u and $t + u$. $P(t)$ can be calculated by taking the matrix exponential of the scaled transition intensity matrix (see, for example, Cox and Miller [**?**]).

$$P(t) = \exp(tQ) \tag{2}$$

For simpler models, it is feasible to calculate an analytic expression for each element of $P(t)$ in terms of Q. This is generally faster and avoids the potential numerical instability of calculating the matrix exponential. Symbolic algebra sofware, such as Mathematica, can be helpful for obtaining these expressions. For example, the three-state illness-death model with no recovery has a transition intensity matrix of

$$Q = \begin{pmatrix} -(q_{12} + q_{13}) & q_{12} & q_{13} \\ 0 & -q_{23} & q_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

The corresponding time t transition probabilities are

$$\begin{aligned}
p_{00}(t) &= e^{-(q_{12}+q_{13})t} \\
p_{01}(t) &= \begin{cases} \frac{q_{12}(-1+e^{(q_{12}+q_{13}-q_{23})t})e^{-(q_{12}+q_{13})t}}{(q_{12}+q_{13}-q_{23})} & (q_{12}+q_{13} \neq q_{23}) \\ q_{12}te^{(-(q_{12}+q_{13})t} & (q_{12}+q_{13} = q_{23}) \end{cases} \\
p_{02}(t) &= \begin{cases} \frac{1+(-q_{13}+q_{23})e^{-(q_{12}+q_{13})t}}{q_{12}+q_{13}-q_{23}} - \frac{q_{12}e^{-q_{23}t}}{q_{12}+q_{13}-q_{23}} & (q_{12}+q_{13} \neq q_{23}) \\ (-1+e^{(q_{12}+q_{13})t} - q_{12}t)e^{-(q_{12}+q_{13})t} & (q_{12}+q_{13} = q_{23}) \end{cases} \\
p_{10}(t) &= 0 \\
p_{11}(t) &= e^{-q_{23}t} \\
p_{12}(t) &= 1 - e^{-q_{23}t} \\
p_{20}(t) &= 0 \\
p_{21}(t) &= 0 \\
p_{22}(t) &= 1
\end{aligned}$$

The *msm* package calculates $P(t)$ analytically for selected 2, 3, 4 and 5-state models. These models are illustrated in Figures 5–8. Notice that the states are not labelled in these figures. Each graph can correspond to several different Q matrices, depending on how the states are labelled. For example, Figure 1 a) illustrates the model defined by either $Q = \begin{pmatrix} -q_{12} & q_{12} \\ 0 & 0 \end{pmatrix}$ or $Q = \begin{pmatrix} 0 & 0 \\ q_{21} & -q_{21} \end{pmatrix}$.

Likelihood for intermittently-observed processes Suppose i indexes M individuals. The data for individual i consist of a series of times $(t_{i1}, \ldots, t_{in_i})$ and corresponding states $(S(t_{i1}), \ldots, S(t_{in_i}))$. Consider a general multi state model, with a pair of successive observed disease states $S(t_j), S(t_{j+1})$ at times t_j, t_{j+1}. The contribution to the likelihood from this pair of states is

Figure 5: Two-state models fitted using analytic $P(t)$ matrices in *msm*. Implemented for all permutations of state labels 1, 2.

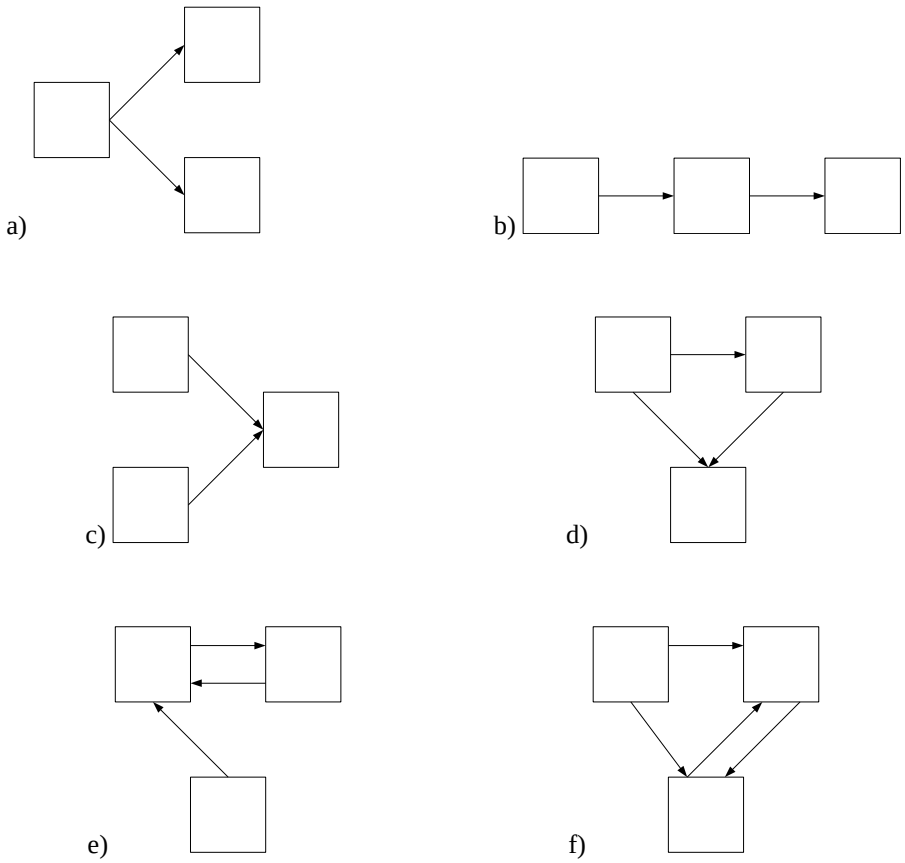


Figure 6: Three-state models fitted using analytic $P(t)$ matrices in *msm*. Implemented for all permutations of state labels 1, 2, 3.

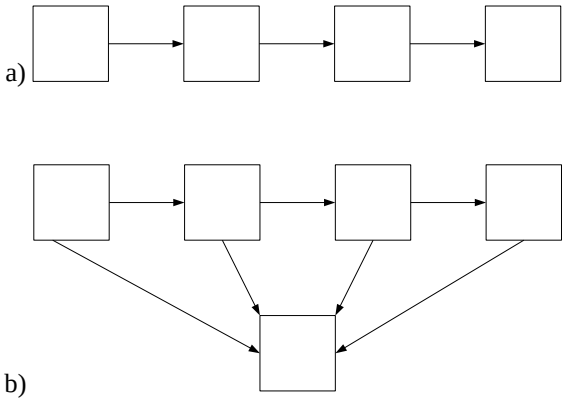


Figure 7: Four-state models fitted using analytic $P(t)$ matrices in *msm*. Implemented for all permutations of state labels 1, 2, 3, 4.

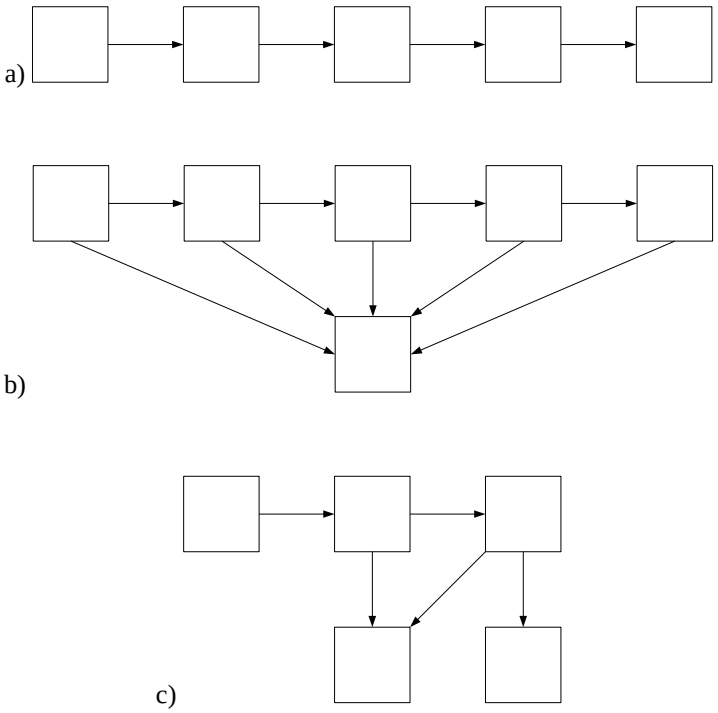


Figure 8: Five-state models fitted using analytic $P(t)$ matrices in *msm*. Implemented for all permutations of state labels 1, 2, 3, 4, 5.

$$L_{i,j} = p_{S(t_j)S(t_{j+1})}(t_{j+1} - t_j) \tag{3}$$

This is the entry of the transition matrix $P(t)$ at the $S(t_j)$th row and $S(t_{j+1})$th column, evaluated at $t = t_{j+1} - t_j$.

The full likelihood $L(Q)$ is the product of all such terms $L_{i,j}$ over all individuals and all transitions. It depends on the unknown transition matrix Q, which was used to determine $P(t)$.

Death states In observational studies of chronic diseases, it is common that the time of death is known, but the state on the previous instant before death is unknown. If $S(t_{j+1}) = D$ is such a death state, then the contribution to the likelihood is summed over the unknown state m on the day before death:

$$L_{i,j} = \sum_{m \neq D} p_{S(t_j),m}(t_{j+1} - t_j) q_{m,D} \tag{4}$$

assuming a time unit of days. The sum is taken over all possible states m which can be visited between $S(t_j)$ and D.

Exactly observed transition times If the times $(t_{i1}, \ldots, t_{in_i})$ had been the *exact* transition times between the states, with no transitions between the observation times, then the contributions would be of the form

$$L_{i,j} = p_{S(t_j)S(t_j)}(t_{j+1} - t_j) q_{S(t_j)S(t_{j+1})} \tag{5}$$

since the state is assumed to be $S(t_j)$ throughout the interval between t_j and t_{j+1} with a known transition to state $S(t_{j+1})$ at t_{j+1}.

Censoring At the end of some chronic disease studies, patients are known to be alive but in an unknown state. For such a censored observation $S(t_{j+1})$, known only to be a state in the set C, the equivalent contribution to the likelihood is

$$L_{i,j} = \sum_{m \in C} p_{S(t_j),m}(t_{j+1} - t_j) \tag{6}$$

More generally, suppose every observation from a particular individual is censored. Observations $1, 2, \ldots n_i$ are known only to be in the sets $C_1, C_2, \ldots, C_{n_i}$ respectively. The likelihood for this individual is a sum of the likelihoods of all possible paths through the unobserved states.

$$L_i = \sum_{s_{n_i} \in C_{n_i}} \cdots \sum_{s_2 \in C_2} \sum_{s_1 \in C_1} p_{s_1 s_2}(t_2 - t_1) p_{s_2 s_3}(t_3 - t_2) \ldots p_{s_{n_i - 1} s_{n_i}}(t_{n_i} - t_{n_i - 1}) \tag{7}$$

Suppose the states comprising the set C_j are $c_1^{(j)}, \ldots, c_{m_j}^{(j)}$. This likelihood can also be written as a matrix product, say,

$$L_i = \mathbf{1}^T P^{1,2} P^{2,3} \ldots P^{n_i - 1, n_i} \mathbf{1} \tag{8}$$

where $P^{j-1,j}$ is a $m_{j-1} \times m_j$ matrix with (r, s) entry $p_{c_r^{(j-1)} c_s^{(j)}}(t_j - t_{j-1})$, and $\mathbf{1}$ is the vector of ones.

The *msm* package allows multi-state models to be fitted to data from processes with arbitrary observation times, exactly-observed transition times, "death" states and censored states, or a mixture of these schemes.

1.5 Covariates

The relation of constant or time-varying characteristics of individuals to their transition rates is often of interest in a multi-state model. Explanatory variables for a particular transition intensity can be investigated by modelling the intensity as a function of these variables. Marshall and Jones [**?**] described a form of a *proportional hazards* model, where the transition intensity matrix elements q_{rs} which are of interest can be replaced by

$$q_{rs}(z(t)) = q_{rs}^{(0)} \exp(\beta_{rs}^T z(t))$$

The new Q is then used to determine the likelihood. If the covariates $z(t)$ are time dependent, the contributions to the likelihood of the form $p_{rs}(t-u)$ are replaced by

$$p_{rs}(t-u, z(u))$$

although this requires that the value of the covariate is known at every observation time u. Sometimes covariates are observed at different times to the main response, for example recurrent disease events or other biological markers. In some of these cases it could be assumed that the covariate is a step function which remains constant between its observation times. The *msm* package allows individual-specific or time dependent covariates to be fitted to transition intensities. Time-dependent covariates are assumed to be known at the same time as the main response, and constant between observation times of the main response.

Marshall and Jones [**?**] described likelihood ratio and Wald tests for covariate selection and testing hypotheses, for example whether the effect of a covariate is the same for all forward transitions in a disease progression model.

1.6 Hidden Markov models

In a *hidden Markov model* (HMM) the states of the Markov chain are not observed. The observed data are governed by some probability distribution (the *emission* distribution) conditionally on the unobserved state. The evolution of the underlying Markov chain is governed by a transition intensity matrix Q as before. (Figure 9). Hidden Markov models are mixture models, where observations are generated from a certain number of unknown distributions. However the distribution changes through time according to states of a hidden Markov chain. This class of model is commonly used in areas such as speech and signal processing [**?**] and the analysis of biological sequence data [**?**]. In engineering and biological sequencing applications, the Markov process usually evolves over an equally-spaced, discrete 'time' space. Therefore most of the theory of HMM estimation was developed for discrete-time models.

HMMs have less frequently been used in medicine, where continuous time processes are often more suitable. A disease process evolves in continuous time, and patients are often monitored at irregular and differing intervals. These models are suitable for estimating population quantities for chronic diseases which have a natural staged interpretation, but which can only be diagnosed by an error-prone marker. The *msm* package can fit continuous-time hidden Markov models with a variety of emission distributions.

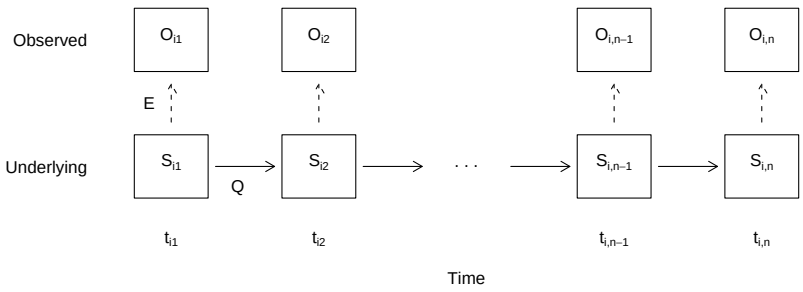


Figure 9: A hidden Markov model in continuous time. Observed states are generated conditionally on an underlying Markov process.

1.6.1 Misclassification models

An example of a hidden Markov model is a multi-state model with misclassification. Here the observed data are states, assumed to be misclassifications of the true, underlying states.

For example, consider a disease progression model with at least a disease-free and a disease state. When screening for the presence of the disease, the screening process can sometimes be subject to error. Then the Markov disease process $S_i(t)$ for individual i is not observed directly, but through realisations $O_i(t)$. The quality of a diagnostic test is often measured by the probabilities that the true and observed states are equal, $Pr(O_i(t) = r|S_i(t) = r)$. Where r represents a 'positive' disease state, this is the *sensitivity*, or the probability that a true positive is detected by the test. Where r represents a 'negative' or disease-free state, this represents the *specificity*, or the probability that, given the condition of interest is absent, the test produces a negative result.

As an extension to the simple multi-state model described in section 1, the *msm* package can fit a general multi-state model with misclassification. For patient i, observation time t_{ij}, observed states O_{ij} are generated conditionally on true states S_{ij} according to a *misclassification matrix* E. This is a $n \times n$ matrix, whose (r, s) entry is

$$e_{rs} = Pr(O(t_{ij}) = s|S(t_{ij}) = r), \tag{9}$$

which we first assume to be independent of time t. Analogously to the entries of Q, some of the e_{rs} may be fixed to reflect knowledge of the diagnosis process. For example, the probability of misclassification may be negligibly small for non-adjacent states of disease. Thus the progression through underlying states is governed by the transition intensity matrix Q, while the observation process of the underlying states is governed by the misclassification matrix E.

To investigate explanatory variables $w(t)$ for the probability of misclassification e_{rs} for each pair

of states r and s, a logistic model can be used,

$$\log \frac{e_{rs}(t)}{1 - e_{rs}(t)} = \gamma_{rs}^T w(t). \tag{10}$$

1.6.2 General hidden Markov model

Consider now a general hidden Markov model in continuous time. The true state of the model S_{ij} evolves as an unobserved Markov process. Observed data y_{ij} are generated conditionally true states $S_{ij} = 1, 2, \ldots, n$ according to a set of distributions $f_1(y|\theta_1, \gamma_1)$, $f_2(y|\theta_2, \gamma_2)$, ..., $f_n(y|\theta_n, \gamma_n)$, respectively. θ_r is a vector of parameters for the state r distribution. One or more of these parameters may depend on explanatory variables through a link-transformed linear model with coefficients γ_r.

1.6.3 Likelihood for general hidden Markov models

A type of EM algorithm known as the *Baum-Welch* or *forward-backward* algorithm [**?**, **?**], is commonly used for hidden Markov model estimation in discrete-time applications. See, for example, Durbin *et al.*[**?**], Albert [**?**]. A generalisation of this algorithm to continuous time was described by Bureau *et al.*[**?**].

The *msm* package uses a direct method of calculating likelihoods in discrete or continuous time based on matrix products. This type of method has been described by Macdonald and Zucchini [**?**, pp. 77–79], Lindsey [**?**, p.73] and Guttorp [**?**]. Satten and Longini [**?**] used this method to calculate likelihoods for a hidden Markov model in continuous time with observations of a continuous marker generated conditionally on underlying discrete states.

Patient i's contribution to the likelihood is

$$\begin{aligned} L_i &= Pr(y_{i1}, \ldots, y_{im_i}) \\ &= \sum Pr(y_{i1}, \ldots, y_{im_i} | S_{i1}, \ldots, S_{im_i}) Pr(S_{i1}, \ldots, S_{im_i}) \end{aligned} \tag{11}$$

where the sum is taken over all possible paths of underlying states $S_{i1}, \ldots, S_{im_i}$. Assume that the observed states are conditionally independent given the values of the underlying states. Also assume the Markov property, $Pr(S_{ij}|S_{i,j-1}, \ldots, S_{i1}) = Pr(S_{ij}|S_{i,j-1})$. Then the contribution L_i can be written as a product of matrices, as follows. To derive this matrix product, decompose the overall sum in equation 11 into sums over each underlying state. The sum is accumulated over the unknown first state, the unknown second state, and so on until the unknown final state:

$$\begin{aligned} L_i &= \sum_{S_{i1}} Pr(y_{i1}|S_{i1}) Pr(S_{i1}) \sum_{S_{i2}} Pr(y_{i2}|S_{i2}) Pr(S_{i2}|S_{i1}) \sum_{S_{i3}} Pr(y_{i3}|S_{i3}) Pr(S_{i3}|S_{i2}) \\ &\quad \ldots \sum_{S_{im_i}} Pr(y_{im_i}|S_{im_i}) Pr(S_{im_i}|S_{in_{i-1}}) \end{aligned} \tag{12}$$

where $Pr(y_{ij}|S_{ij})$ is the emission probability density. For misclassification models, this is the misclassification probability $e_{S_{ij}O_{ij}}$. For general hidden Markov models, this is the probability density $f_{S_{ij}}(y_{ij}|\theta_{S_{ij}}, \gamma_{S_{ij}})$. $Pr(S_{i,j+1}|S_{ij})$ is the $(S_{ij}, S_{i,j+1})$ entry of the Markov chain transition matrix $P(t)$ evaluated at $t = t_{i,j+1} - t_{ij}$. Let f be the vector with r element the product of the initial state

occupation probability $Pr(S_{i1} = r)$ and $Pr(y_{i1}|r)$, and let $\mathbf{1}$ be a column vector consisting of ones. For $j = 2, \ldots, m_i$ let T_{ij} be the $n \times n$ matrix with (r, s) entry

$$Pr(y_{ij}|s)p_{rs}(t_{ij} - t_{i,j-1})$$

Then subject i's likelihood contribution is

$$L_i = fT_{i2}T_{i3}, \ldots T_{im_i}\mathbf{1} \tag{13}$$

If $S(t_j) = D$ is an absorbing state such as death, measured without error, whose entry time is known exactly, then the contribution to the likelihood is summed over the unknown state at the previous instant before death. the day before entry. The (r, s) entry of T_{ij} is then

$$p_{rs}(t_j - t_{j-1})q_{s,D}$$

Section 2.13 describes how to fit multi-state models with misclassification using the *msm* package, and Section 2.17 describes how to apply general hidden Markov models.

1.6.4 Example of a general hidden Markov model

Jackson and Sharples [**?**] described a model for FEV_1 (forced expiratory volume in 1 second) in recipients of lung transplants. These patients were at risk of BOS (bronchiolitis obliterans syndrome), a progressive, chronic deterioration in lung function. In this example, BOS was modelled as a discrete, staged process, a model of the form illustrated in Figure 2, with 4 states. State 1 represents absence of BOS. State 1 BOS is roughly defined as a sustained drop below 80% below baseline FEV_1, while state 2 BOS is a sustained drop below 65% baseline. FEV_1 is measured as a percentage of a baseline value for each individual, determined in the first six months after transplant, and assumed to be 100% baseline at six months.

As FEV_1 is subject to high short-term variability due to acute events and natural fluctuations, the exact BOS state at each observation time is difficult to determine. Therefore, a hidden Markov model for FEV_1, conditionally on underlying BOS states, was used to model the natural history of the disease. Discrete states are appropriate as onset is often sudden.

Model 1 Jackson [**?**] considered models for these data where FEV_1 were Normally distributed, with an unknown mean and variance conditionally each state (14). This model seeks the most likely location for the within-state FEV_1 means.

$$y_{ij}|\{S_{ij} = k\} \sim N(\mu_k + \beta x_{ij}, \sigma_k^2) \tag{14}$$

Model 2 Jackson and Sharples [**?**] used a more complex two-level model for FEV_1 measurements. Level 1 (15) represents the short-term fluctuation error of the marker around its underlying continuous value y_{ij}^{hid}. Level 2 (16) represents the distribution of y_{ij}^{hid} conditionally on each underlying state, as follows.

$$y_{ij}|y_{ij}^{hid} \qquad \sim N(y_{ij}^{hid} + \beta x_{ij}, \sigma_\epsilon^2) \tag{15}$$

$$y_{ij}^{hid}|S_{ij} \quad \sim \quad \left\{ \begin{array}{lll} \text{State} & \text{Three state model} & \text{Four state model} \\ S_{ij}=0 & N(\mu_0,\sigma_0^2)I_{[80,\infty)} & N(\mu_0,\sigma_0^2)I_{[80,\infty)} \\ S_{ij}=1 & N(\mu_1,\sigma_1^2)I_{(0,80)} & Uniform(65,80) \\ S_{ij}=2 & \text{(death)} & N(\mu_2,\sigma_2^2)I_{(0,65)} \\ S_{ij}=3 & & \text{(death)} \end{array} \right. \tag{16}$$

Integrating over y_{ij}^{hid} gives an explicit distribution for y_{ij} conditionally on each underlying state (given in Section 2.17, Table 1). Similar distributions were originally applied by Satten and Longini [**?**] to modelling the progression through discrete, unobserved HIV disease states using continuous CD4 cell counts. The *msm* package includes density, quantile, cumulative density and random number generation functions for these distributions.

In both models 1 and 2, the term βx_{ij} models the short-term fluctuation of the marker in terms of acute events. x_{ij} is an indicator for the occurrence of an acute rejection or infection episode within 14 days of the observation of FEV_1.

Section 2.17 describes how these and more general hidden Markov models can be fitted with the *msm* package.

2 Fitting multi-state models with `msm`

msm is a package of functions for multi-state modelling using the R statistical software. The `msm` function itself implements maximum-likelihood estimation for general multi-state Markov or hidden Markov models in continuous time. We illustrate its use with a set of data from monitoring heart transplant patients. Throughout this section "`>`" indicates the R command prompt, *`slanted typewriter`* text indicates R commands, and `typewriter` text R output.

2.1 Installing `msm`

The easiest way to install the *msm* package on a computer connected to the Internet is to run the R command:

```
install.packages("msm")
```

This downloads *msm* from the CRAN archive of contributed R packages (`cran.r-project.org` or one of its mirrors) and installs it to the default R system library. To install to a different location, for example if you are a normal user with no administrative privileges, create a directory in which R packages are to be stored, say, `/your/library/dir`, and run

```
install.packages("msm", lib='/your/library/dir')
```

After *msm* has been installed, its functions can be made visible in an R session by

```
> library(msm)
```

or, if it has been installed into a non-default library,

```
library(msm, lib.loc='/your/library/dir')
```

2.2 Getting the data in

The data are specified as a series of observations, grouped by patient. At minimum there should be a data frame with variables indicating

- the time of the observation,
- the observed state of the process.

If the data do not also contain

- the subject identification number,

then all the observations are assumed to be from the same subject. The subject ID does not need to be numeric, but data must be grouped by subject, and observations must be ordered by time within subjects. An example data set, taken from monitoring a set of heart transplant recipients, is provided with *msm*. Sharples *et al.* [**?**] studied the progression of coronary allograft vasculopathy (CAV), a post-transplant deterioration of the arterial walls, using these data. Risk factors and the accuracy of the screening test were investigated using multi-state Markov and hidden Markov models.

This data set can be made available to the current R session by issuing the command

```
> data(heart)
```

The first three patient histories are shown below. There are 622 patients in all. `PTNUM` is the subject identifier. Approximately each year after transplant, each patient has an angiogram, at which CAV can be diagnosed. The result of the test is in the variable `state`, with possible values 1, 2, 3 representing CAV-free, mild CAV and moderate or severe CAV respectively. A value of 4 is recorded at the date of death. `years` gives the time of the test in years since the heart transplant. Other variables include `age` (age at screen), `dage` (donor age), `sex` (0=male, 1=female), `pdiag` (primary diagnosis, or reason for transplant - IHD represents ischaemic heart disease, IDC represents idiopathic dilated cardiomyopathy), and `cumrej` (cumulative number of rejection episodes).

```
> heart[1:21, ]
```

```
    PTNUM      age    years dage sex pdiag cumrej state
1  100002 52.49589 0.000000   21   0   IHD      0     1
2  100002 53.49863 1.002740   21   0   IHD      2     1
3  100002 54.49863 2.002740   21   0   IHD      2     2
4  100002 55.58904 3.093151   21   0   IHD      2     2
5  100002 56.49589 4.000000   21   0   IHD      3     2
6  100002 57.49315 4.997260   21   0   IHD      3     3
7  100002 58.35068 5.854795   21   0   IHD      3     4
8  100003 29.50685 0.000000   17   0   IHD      0     1
9  100003 30.69589 1.189041   17   0   IHD      1     1
10 100003 31.51507 2.008219   17   0   IHD      1     3
11 100003 32.49863 2.991781   17   0   IHD      2     4
12 100004 35.89589 0.000000   16   0   IDC      0     1
13 100004 36.89863 1.002740   16   0   IDC      2     1
14 100004 37.90685 2.010959   16   0   IDC      2     1
15 100004 38.90685 3.010959   16   0   IDC      2     1
16 100004 39.90411 4.008219   16   0   IDC      2     1
17 100004 40.88767 4.991781   16   0   IDC      2     1
18 100004 41.91781 6.021918   16   0   IDC      2     2
19 100004 42.91507 7.019178   16   0   IDC      2     3
20 100004 43.91233 8.016438   16   0   IDC      2     3
21 100004 44.79726 8.901370   16   0   IDC      2     4
```

A useful way to summarise multi-state data is as a frequency table of pairs of consecutive states. This counts over all individuals, for each state r and s, the number of times an individual had an observation of state r followed by an observation of state s. The function `statetable.msm` can be used to produce such a table, as follows,

```
> statetable.msm(state, PTNUM, data = heart)
```

```
    to
from    1    2    3    4
   1 1367  204   44  148
   2   46  134   54   48
   3    4   13  107   55
```

Thus there were 148 CAV-free deaths, 48 deaths from state 2, and 55 deaths from state 3. On only four occasions was there an observation of severe CAV followed by an observation of no CAV.

2.3 Specifying a model

We now specify the multi-state model to be fitted to the data. A model is governed by a transition intensity matrix Q. For the heart transplant example, there are four possible states through which the patient can move, corresponding to CAV-free, mild/moderate CAV, severe CAV and death. We assume that the patient can advance or recover from consecutive states while alive, and die from any state. Thus the model is illustrated by Figure 2 with four states, and we have

$$Q = \begin{pmatrix} -(q_{12}+q_{14}) & q_{12} & 0 & q_{14} \\ q_{21} & -(q_{21}+q_{23}+q_{24}) & q_{23} & q_{24} \\ 0 & q_{32} & -(q_{32}+q_{34}) & q_{34} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

It is important to remember that this defines which *instantaneous* transitions can occur in the Markov process, and that the data are *snapshots* of the process (see section 1.3). Although there were 44 occasions on which a patient was observed in state 1 followed by state 3, the underlying model specifies that the patient must have passed through state 2 in between. If your data represent the exact and complete transition times of the process, then you must specify `exacttimes=TRUE` in the call to `msm`.

To tell `msm` what the allowed transitions of our model are, we define a matrix of the same size as Q, containing zeroes in the positions where the entries of Q are zero. All other positions contain an initial value for the corresponding transition intensity. The diagonal entries supplied in this matrix do not matter, as the diagonal entries of Q are defined as minus the sum of all the other entries in the row. This matrix will eventually be used as the `qmatrix` argument to the `msm` function. For example,

```
> twoway4.q <- rbind(c(0, 0.25, 0, 0.25), c(0.166,
+     0, 0.166, 0.166), c(0, 0.25, 0, 0.25), c(0,
+     0, 0, 0))
```

Fitting the model is a process of finding values of the seven unknown transition intensities: q_{12}, q_{14}, q_{21}, q_{23}, q_{24}, q_{32}, q_{34}, which maximise the likelihood.

2.4 Specifying initial values

The likelihood is maximised by numerical methods, which need a set of initial values to start the search for the maximum. For reassurance that the true maximum likelihood estimates have been found, models should be run repeatedly starting from different initial values. However a sensible choice of initial values can be important for unstable models with flat or multi-modal likelihoods. For example, the transition rates for a model with misclassification could be initialised at the corresponding estimates for an approximating model without misclassification. Initial values for a model without misclassification could be set by supposing that transitions between states take place only at the observation times. If we observe n_{rs} transitions from state r to state s, and a total of n_r transitions from state r, then q_{rs}/q_{rr} can be estimated by n_{rs}/n_r. Then, given a total of T_r years spent in state r, the mean sojourn time $1/q_{rr}$ can be estimated as T_r/n_r. Thus, n_{rs}/T_r is a crude estimate of q_{rs}. The *msm* package provides a function `crudeinits.msm` for calculating initial values in this way.

```
> crudeinits.msm(state ~ years, PTNUM, data = heart,
+     qmatrix = twoway4.q)
```

```
           1           2          3          4
1 -0.1173149  0.06798932  0.0000000 0.04932559
2  0.1168179 -0.37584883  0.1371340 0.12189692
3  0.0000000  0.04908401 -0.2567471 0.20766310
4  0.0000000  0.00000000  0.0000000 0.00000000
```

However, if there are are many changes of state in between the observation times, then this crude approach may fail to give sensible initial values. For the heart transplant example we could also guess that the mean period in each state before moving to the next is about 2 years, and there is an equal probability of progression, recovery or death. This gives $q_{rr} = -0.5$ for $r = 1, 2, 3$, and $q_{12} = q_{14} = 0.25$, $q_{21} = q_{23} = q_{24} = 0.166$, $q_{32} = q_{34} = 0.25$, and the `twoway4.q` shown above.

2.5 Running `msm`

To fit the model, call the `msm` function with the appropriate arguments. For our running example, we have defined a data set `heart`, a matrix `twoway4.q` indicating the allowed transitions, and initial values. We are ready to run `msm`.

Model 1: simple bidirectional model

```
> heart.msm <- msm(state ~ years, subject = PTNUM,
+     data = heart, qmatrix = twoway4.q, death = 4)
```

In this example the day of death is assumed to be recorded exactly, as is usual in studies of chronic diseases. At the previous instant before death the state of the patient is unknown. Thus we specify `death = 4`, to indicate to `msm` that state 4 is a "death" state. In terms of the multi-state model, a "death" state is assumed to have a known entry time, but the individual is in an unknown transient state at the previous instant. If the model had five states and states 4 and 5 were two competing causes of death with death times recorded exactly, then we would specify `death = c(4,5)`.

By default, the data are assumed to represent snapshots of the process at arbitrary times. However, observations can also represent exact times of transition, "death" times, or a mixture of these. See the `obstype` argument to `msm`.

While the `msm` function runs, it searches for the maximum of the likelihood of the unknown parameters. Internally, it uses the R function `optim` to minimise the minus log-likelihood. When the data set, the model, or both, are large, then this may take a long time. It can then be useful to see the progress of the optimisation algorithm. To do this, we can specify a `control` argument to `msm`, which is passed internally to the `optim` function. For example `control = list(trace=1, REPORT=1)`. See the help page for `optim`,

```
> help(optim)
```

for more options to control the optimisation. When completed, the `msm` function returns a value. This value is a list of the important results of the model fitting, including the parameter estimates and their covariances. To keep these results for post-processing, we store them in an R object, here called `heart.msm`. When running several similar `msm` models, it is recommended to store the respective results in informatively-named objects.

2.6 Showing results

To show the maximum likelihood estimates and 95% confidence intervals, type the name of the fitted model object at the R command prompt. [1] The confidence level can be changed using the `cl` argument to `msm`.

```
> heart.msm

Call:
msm(formula = state ~ years, subject = PTNUM, data = heart, qmatrix = twoway4.q,

Maximum likelihood estimates:
Transition intensity matrix

        State 1                    State 2
State 1 -0.1702 (-0.1901,-0.1524) 0.1277 (0.1112,0.1466)
State 2 0.2244 (0.167,0.3016)     -0.6062 (-0.7068,-0.5199)
State 3 0                         0.1312 (0.08003,0.2152)
State 4 0                         0
        State 3                    State 4
State 1 0                         0.04253 (0.03414,0.05298)
State 2 0.3406 (0.2714,0.4273)    0.0412 (0.01193,0.1423)
State 3 -0.4361 (-0.5517,-0.3447) 0.3049 (0.2368,0.3925)
State 4 0                         0

-2 * log-likelihood:  3968.803
```

From the estimated intensity matrix, we see patients are three times as likely to develop symptoms than die without symptoms (first row). After disease onset (state 2), progression to severe symptoms (state 3) is 50% more likely than recovery, and death from the severe disease state is rapid (mean of 1 / -0.44 = 2.3 years in state 3).

Section 2.9 describes various functions that can be used to obtain summary information from the fitted model.

2.7 Covariates on the transition rates

We now model the effect of explanatory variables on the rates of transition, using a proportional intensities model. Now we have an intensity matrix $Q(z)$ which depends on a covariate vector z. For each entry of $Q(z)$, the transition intensity for patient i at observation time j is $q_{rs}(z_{ij}) = q_{rs}^{(0)} \exp(\beta_{rs}^T z_{ij})$. The covariates z are specified through the `covariates` argument to `msm`. If z_{ij} is time-dependent, we assume it is constant in between the observation times of the Markov process. `msm` calculates the probability for a state transition from times $t_{i,j-1}$ to t_{ij} using the covariate value at time $t_{i,j-1}$.

We consider a model with just one covariate, female sex. Out of the 622 transplant recipients, 535 are male and 87 are female. By default, all linear covariate effects β_{rs} are initialised to zero.

[1] This is equivalent to typing `print.msm(heart.msm)`. The function `print.msm` formats the important information in the model object for printing on the screen.

To specify different initial values, use a `covinits` argument, as described in `help(msm)`. Initial values given in the `qmatrix` represent the intensities at covariate values of zero.

Model 2: sex as a covariate

```
> heartsex.msm <- msm(state ~ years, subject = PTNUM,
+     data = heart, qmatrix = twoway4.q, death = 4,
+     covariates = ~sex)
```

The `msm` object will now include the estimated covariate effects and their confidence intervals.

```
> heartsex.msm

Call:
msm(formula = state ~ years, subject = PTNUM, data = heart, qmatrix = twoway4.q,

Maximum likelihood estimates:
Transition intensity matrix with covariates set to their means

        State 1                   State 2
State 1 -0.1726 (-0.193,-0.1543) 0.1308 (0.1138,0.1505)
State 2 0.2429 (0.1817,0.3248)   -0.6811 (-0.7989,-0.5807)
State 3 0                        0.1748 (0.1028,0.2974)
State 4 0                        0
        State 3                   State 4
State 1 0                         0.04175 (0.03333,0.05229)
State 2 0.3794 (0.3016,0.4774)    0.05876 (0.0251,0.1375)
State 3 -0.4813 (-0.6156,-0.3763) 0.3065 (0.238,0.3947)
State 4 0                         0

Log-linear effects of sex

        State 1                 State 2
State 1 0                       -0.6276 (-1.136,-0.1188)
State 2 -0.01686 (-1.049,1.015) 0
State 3 0                       0.7751 (-1.914,3.465)
State 4 0                       0
        State 3               State 4
State 1 0                     0.2141 (-0.3622,0.7904)
State 2 0.4474 (-0.4951,1.39) 0.5854 (-1.266,2.437)
State 3 0                     0.6701 (-0.162,1.502)
State 4 0                     0

-2 * log-likelihood:  3961.345
```

Comparing the estimated log-linear effects of age and their standard errors, we see that the disease onset rate is smaller for females, whereas none of the other effects are significant. The first matrix

shown in the output of printing `heartsex.msm` is the estimated transition intensity matrix $q_{rs}(z) = q_{rs}^{(0)} \exp(\beta_{rs}^T z)$ with the covariate z set to its mean value in the data. This represents an average intensity matrix for the population of 535 male and 87 female patients. To extract separate intensity matrices for male and female patients ($z = 0$ and 1 respectively), use the function `qmatrix.msm`, as shown below. This and similar summary functions will be described in more detail in section 2.9.

```
> qmatrix.msm(heartsex.msm, covariates = list(sex = 0))

        State 1                    State 2
State 1 -0.1818 (-0.2044,-0.1616) 0.1411 (0.1221,0.163)
State 2 0.2434 (0.1801,0.3291)    -0.6578 (-0.7706,-0.5615)
State 3 0                         0.1593 (0.09821,0.2583)
State 4 0                         0
        State 3                   State 4
State 1 0                         0.04069 (0.03182,0.05202)
State 2 0.3596 (0.2859,0.4522)    0.05477 (0.02136,0.1404)
State 3 -0.442 (-0.5649,-0.3459) 0.2828 (0.2166,0.3692)
State 4 0                         0

> qmatrix.msm(heartsex.msm, covariates = list(sex = 1))

        State 1                     State 2
State 1 -0.1257 (-0.1747,-0.09044) 0.07531 (0.04623,0.1227)
State 2 0.2394 (0.08923,0.6421)    -0.9002 (-1.776,-0.4562)
State 3 0                          0.3458 (0.02453,4.873)
State 4 0                          0
        State 3                State 4
State 1 0                      0.0504 (0.02993,0.08488)
State 2 0.5625 (0.2255,1.403)  0.09835 (0.01996,0.4845)
State 3 -0.8984 (-2.5,-0.3228) 0.5527 (0.2513,1.216)
State 4 0                      0
```

We may also want to constrain the effect of age to be equal for certain transition rates, to reduce the number of parameters in the model, or to investigate hypotheses on the covariate effects. A `constraint` argument can be used to indicate which of the transition rates have common covariate effects.

Model 3: constrained covariate effects

```
> heart3.msm <- msm(state ~ years, subject = PTNUM,
+     data = heart, qmatrix = twoway4.q, death = 4,
+     covariates = ~sex, constraint = list(sex = c(1,
+         2, 3, 1, 2, 3, 2)))
```

This constrains the effect of age to be equal for the progression rates q_{12}, q_{23}, equal for the death rates q_{14}, q_{24}, q_{34}, and equal for the recovery rates q_{21}, q_{32}. The intensity parameters are assumed to be ordered by reading across the rows of the transition matrix, starting at the first row: ($q_{12}, q_{14}, q_{21}, q_{23}, q_{24}, q_{32}, q_{34}$),

giving constraint indicators `(1,2,3,1,2,3,2)`. Any vector of increasing numbers can be used for the indicators.

In a similar manner, we can constrain some of the baseline transition intensities to be equal to one another, using the `qconstraint` argument. For example, to constrain the rates q_{12} and q_{23} to be equal, and q_{24} and q_{34} to be equal, specify `qconstraint = c(1,2,3,1,4,5,4)`.

2.8 Fixing parameters at their initial values

For exploratory purposes we may want to fit a model assuming that some parameters are fixed, and estimate the remaining parameters. This may be necessary in cases where there is not enough information in the data to be able to estimate a proposed model, and we have strong prior information about a certain transition rate. To do this, use the `fixedpars` argument to `msm`. For model 1, the following statement fixes the parameters numbered 2, 5, 7, that is, q_{14}, q_{24}, q_{34}, to their initial values (0.25, 0.166 and 0.25, respectively).

Model 4: fixed parameters

```
> heart4.msm <- msm(state ~ years, subject = PTNUM,
+     data = heart, qmatrix = twoway4.q, death = 4,
+     control = list(trace = 2, REPORT = 1), fixedpars = c(2,
+         5, 7))
```

A `fixedpars` statement can also be useful for fixing covariate effect parameters to zero, that is to assume no effect of a covariate on a certain transition rate.

2.9 Extractor functions

We may want to extract some of the information from the `msm` model fit for post-processing, for example for plotting graphs or generating summary tables. A set of functions is provided for extracting interesting features of the fitted model.

Intensity matrices The function `qmatrix.msm` extracts a transition intensity matrix and its confidence intervals for a given set of covariate values, as shown in section 2.7. Confidence intervals are calculated from the covariance matrix of the estimates by assuming the distribution is symmetric on the log scale. Standard errors for the intensities are also available from the object returned by `qmatrix.msm`. These are calculated by the delta method. The *msm* package provides a general-purpose function `deltamethod` for estimating the variance of a function of a random variable X given the expectation and variance of X. See `help(deltamethod)` for further details.

Transition probability matrices The function `pmatrix.msm` extracts the estimated transition probability matrix $P(t)$ within a given time. For example, for model 1, the 10 year transition probabilities are given by:

```
> pmatrix.msm(heart.msm, t = 10)
```

```
          State 1    State 2    State 3   State 4
State 1 0.30959690 0.09780067 0.08775948 0.5048430
State 2 0.17187999 0.06588634 0.07810046 0.6841332
State 3 0.05943821 0.03009829 0.04705873 0.8634048
State 4 0.00000000 0.00000000 0.00000000 1.0000000
```

Thus, a typical person in state 1, disease-free, has a probability of 0.5 of being dead ten years from now, a probability of 0.3 being still disease-free, and probabilities of 0.1 of being alive with mild/moderate or severe disease, respectively.

This assumes Q is constant within the desired time interval. For non-homogeneous processes, where Q varies with time-dependent covariates but can be approximated as piecewise constant, there is an equivalent function `pmatrix.piecewise.msm`. Consult its help page for further details.

There are no estimates of error presented with `pmatrix.msm`, however bootstrap methods may be feasible for simpler models.

Mean sojourn times The function `sojourn.msm` extracts the estimated mean sojourn times in each transient state, for a given set of covariate values.

```
> sojourn.msm(heart.msm)

        estimates        SE        L        U
State 1  5.874810 0.3310261 5.260554 6.560791
State 2  1.649685 0.1292902 1.414784 1.923587
State 3  2.292950 0.2750939 1.812478 2.900791
```

Total length of stay Mean sojourn times describe the average period in a single stay in a state. For processes with successive periods of recovery and relapse, we may want to forecast the total time spent healthy or diseased, before death. The function `totlos.msm` estimates the forecasted total length of time spent in each transient state s between two future time points t_1 and t_2, for a given set of covariate values. This defaults to the expected amount of time spent in each state between the start of the process (time 0, the present time) and death or a specified future time. This is obtained as

$$L_s = \int_{t_1}^{t_2} P(t)_{r,s} dt$$

where r is the state at the start of the process, which defaults to 1. This is calculated using numerical integration. For model 1, each patient is forecasted to spend 8.8 years disease free, 2.2 years with mild or moderate disease and 1.8 years with severe disease. Notice that there are currently no estimates of error available from `totlos.msm`, however bootstrap methods may be feasible for simpler models.

```
> totlos.msm(heart.msm)

 State 1  State 2  State 3
8.823770 2.236885 1.746796
```

Ratio of transition intensities The function `qratio.msm` estimates a ratio of two entries of the transition intensity matrix at a given set of covariate values, together with a confidence interval estimated assuming normality on the log scale and using the delta method. For example, we may want to estimate the ratio of the progression rate q_{12} into the first state of disease to the corresponding recovery rate q_{21}. For example in model 1, recovery is 1.8 times as likely as progression.

```
> qratio.msm(heart.msm, ind1 = c(2, 1), ind2 = c(1,
+     2))

 estimate        se         L         U
1.7574521 0.2455929 1.3363906 2.3111790
```

Hazard ratios for transition The function `hazard.msm` gives the estimated hazard ratios corresponding to each covariate effect on the transition intensities. 95% confidence limits are computed by assuming normality of the log-effect. For example, for model 2 with female sex as a covariate, the following hazard ratios show more clearly that the only transition on which the effect of sex is significant at the 5% level is the 1-2 transition.

```
> hazard.msm(heartsex.msm)

$sex
                         HR         L          U
State 2 - State 1 0.9832775 0.3504128  2.7591299
State 1 - State 2 0.5338549 0.3209412  0.8880164
State 3 - State 2 2.1707943 0.1474241 31.9645757
State 2 - State 3 1.5641957 0.6094987  4.0142958
State 1 - State 4 1.2387824 0.6961528  2.2043748
State 2 - State 4 1.7957093 0.2818257 11.4417254
State 3 - State 4 1.9544936 0.8504352  4.4918710
```

Setting covariate values All of these extractor functions take an argument called `covariates`. If this argument is omitted, for example,

```
> qmatrix.msm(heart.msm)
```

then the intensity matrix is evaluated as $Q(\bar{x})$ with all covariates set to their mean values $\bar{x}$ in the data. Alternatively, set `covariates` to 0 to return the result $Q(0)$ with covariates set to zero. This will usually be preferable for categorical covariates, where we wish to see the result for the baseline category.

```
> qmatrix.msm(heartsex.msm, covariates = 0)
```

Alternatively, the desired covariate values can be specified explicitly as a list,

```
> qmatrix.msm(heartsex.msm, covariates = list(sex = 1))
```

If a covariate is categorical, that is, an R *factor* with k levels, then we use its internal representation as a set of $k-1$ 0/1 indicator functions. For example, consider a covariate `cov`, with three levels, `VAL1, VAL2, VAL3`, where the baseline level is `VAL1`. To set the value of `cov` to be `VAL1`, `VAL2` or `VAL3`, respectively, use statements such as

```
qmatrix.msm(example.msm, covariates = list(age = 60, covVAL2=0, covVAL3=0))
qmatrix.msm(example.msm, covariates = list(age = 60, covVAL2=1, covVAL3=0))
qmatrix.msm(example.msm, covariates = list(age = 60, covVAL2=0, covVAL3=1))
```

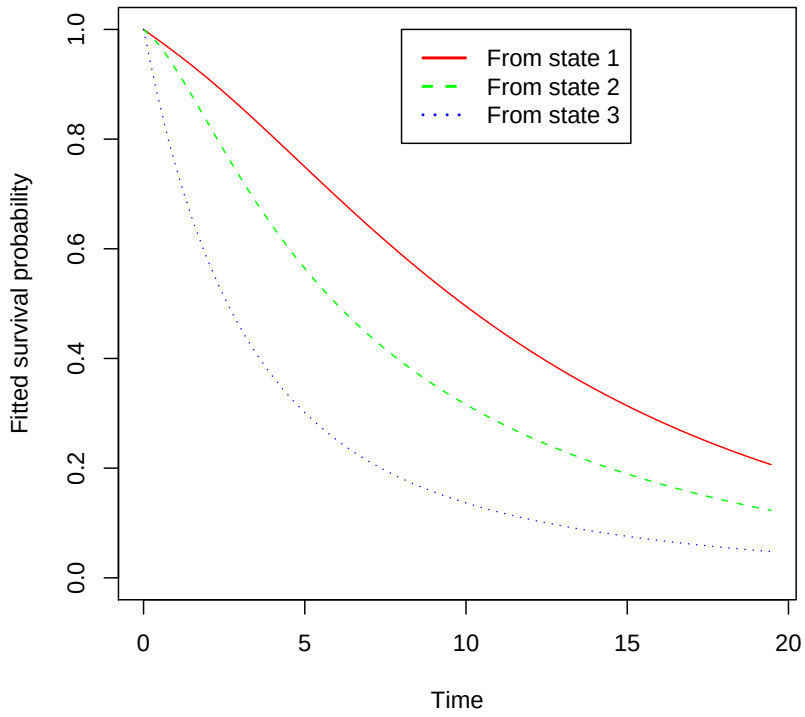
respectively. (This procedure is likely to be simplified in future versions of the package.)

2.10 Survival plots

In studies of chronic disease, an important use of multi-state models is in predicting the probability of survival for patients in increasingly severe states of disease, for some time t in the future. This can be obtained directly from the transition probability matrix $P(t)$.

The `plot` method for `msm` objects produces a plot of the expected probability of survival against time, from each transient state. Survival is defined as not entering the final absorbing state.

```
> plot(heart.msm, legend.pos = c(8, 1))
```



This shows that the 10-year survival probability with severe CAV is approximately 0.1, as opposed to 0.3 with mild CAV and 0.5 without CAV. With severe CAV the survival probability diminishes very quickly to around 0.3 in the first five years after transplant. The `legend.pos` argument adjusts the position of the legend in case of clashes with the plot lines. A `times` argument can be supplied to indicate the time interval to forecast survival for.

A more sophisticated analysis of these data might explore competing causes of death from causes related or unrelated to the disease under study.

2.11 Convergence failure

Inevitably if over-complex models are applied with insufficient data then the parameters of the model will not be identifiable. This will result in the optimisation algorithm failing to find the maximum of the log-likelihood, or even failing to evaluate the likelihood. For example, it will commonly be inadvisable to include several covariates in a model simultaneously.

In some circumstances, the optimisation may report convergence, but fail to calculate any standard errors. In these cases, the Hessian of the log-likelihood at the reported solution is not positive definite. Thus the reported solution is probably close to the maximum, but not the maximum.

Initial values Make sure that a sensible set of initial values have been chosen. The optimisation may only converge within a limited range of 'informative' initial values.

Scaling It is often necessary to apply a scaling factor to normalise the likelihood (`fnscale`), or certain individual parameters (`parscale`). This may prevent overflow or underflow problems within the optimisation. For example, if the value of the -2 $\times$ log-likelihood is around 5000, then the following option leads to an minimisation of the -2 $\times$ log-likelihood on an approximate unit scale: `options = list(fnscale = 5000)`

It is also advisable to analyse all variables, including covariates and the time unit, on a roughly normalised scale. For example, working in terms of a time unit of months or years instead of days, when the data range over thousands of days.

Convergence criteria "False convergence" can sometimes be solved by tightening the criteria (`reltol`, defaults to `1e-08`) for reporting convergence of the optimisation. For example, `options = list(reltol = 1e-16)`.

Alternatively consider using smaller step sizes for the numerical approximation to the gradient, used in calculating the Hessian. This is given by the control parameter `ndeps`. For example, for a model with 5 parameters, `options = list(ndeps = rep(1e-6, 5))`

Choice of algorithm By default, `msm` uses a Nelder-Mead algorithm which does not use analytic derivatives of the objective function. For potentially greater speed and accuracy, consider using a quasi-Newton algorithm such as the "BFGS" method of `optim`, which can make use of analytic derivatives, for example, `method = "BFGS", use.deriv = TRUE`.

Model simplification If none of these numerical adjustments lead to convergence, then the model is probably over-complicated. There may not be enough information in the data on a certain transition rate. It is always recommended to count all the pairs of transitions between states in successive observation times, making a frequency table of previous state against current state

(function `statetable.msm`). Although the data are a series of snapshots of a continuous-time process, and the actual transitions take place in between the observation times, this type of table may still be helpful. If there are not many observed 'transitions' from state 2 to state 4, for example, then the data may be insufficient to estimate q_{24}.

For a staged disease model (Figure 2), the number of disease states should be low enough that all transition rates can be estimated. Consecutive states of disease severity should be merged if necessary. If it is realistic, consider applying constraints on the intensities or the covariate effects so that the parameters are equal for certain transitions, or zero for certain transitions.

2.12 Model assessment

Observed and expected prevalence To compare the relative fit of two nested models, it is easy to compare their likelihoods. However it is not always easy to determine how well a fitted multi-state model describes an irregularly-observed process. Ideally we would like to compare observed data with fitted or expected data under the model. If there were times at which all individuals were observed then the fit of the expected numbers in each state or *prevalences* can be assessed directly at those times. Otherwise, some approximations are necessary. We could assume that an individual's state at an arbitrary time t was the same as the state at their previous observation time. This might be fairly accurate if observation times are close together. This approach is taken by the function `prevalence.msm`, which constructs a table of observed and expected numbers and percentages of individuals in each state at a set of times.

A set of expected counts can be produced if the process begins at a common time for all individuals. Suppose at this time, each individual is in state 0. Then given $n(t)$ individuals are under observation at time t, the expected number of individuals in state r at time t is $n(t)P(t)_{0,r}$.

For example, we calculate the observed expected numbers and percentages at two-yearly intervals up to 20 years after transplant, for the heart transplant model `heart.msm`. The number of individuals still alive and under observation decreases from 622 to 251 at year 20.

```
> options(digits = 3)
> prevalence.msm(heart.msm, times = seq(0, 20, 2))

Calculating approximate observed state prevalences...
Forecasting expected state prevalences...
$Observed
   State 1 State 2 State 3 State 4 Total
0      622       0       0       0   622
2      507      20       7      54   588
4      330      37      24      90   481
6      195      43      28     129   395
8      117      44      21     161   343
10      60      25      22     189   296
12      26      11      12     221   270
14      11       3       6     238   258
16       4       0       3     245   252
18       0       0       2     249   251
20       0       0       0     251   251
```

```
$Expected
   State 1 State 2 State 3 State 4 Total
0    622.0    0.00    0.00     0.0   622
2    437.0   74.51   23.70    52.8   588
4    279.8   68.66   38.66    93.9   481
6    184.2   52.07   37.95   120.8   395
8    129.9   39.41   32.84   140.9   343
10    91.6   28.95   25.98   149.4   296
12    68.7   22.21   20.80   158.3   270
14    54.0   17.73   17.02   169.2   258
16    43.5   14.41   14.05   180.0   252
18    35.8   11.92   11.72   191.6   251
20    29.6    9.88    9.77   201.8   251

$`Observed percentages`
   State 1 State 2 State 3 State 4
0   100.00    0.00   0.000    0.00
2    86.22    3.40   1.190    9.18
4    68.61    7.69   4.990   18.71
6    49.37   10.89   7.089   32.66
8    34.11   12.83   6.122   46.94
10   20.27    8.45   7.432   63.85
12    9.63    4.07   4.444   81.85
14    4.26    1.16   2.326   92.25
16    1.59    0.00   1.190   97.22
18    0.00    0.00   0.797   99.20
20    0.00    0.00   0.000  100.00

$`Expected percentages`
   State 1 State 2 State 3 State 4
0    100.0    0.00    0.00    0.00
2     74.3   12.67    4.03    8.97
4     58.2   14.27    8.04   19.51
6     46.6   13.18    9.61   30.57
8     37.9   11.49    9.57   41.07
10    31.0    9.78    8.78   50.48
12    25.4    8.23    7.70   58.64
14    20.9    6.87    6.60   65.59
16    17.3    5.72    5.57   71.43
18    14.3    4.75    4.67   76.32
20    11.8    3.94    3.89   80.38
```

Comparing the observed and expected percentages in states 1, 2 and 3, we see that the predicted number of individuals who die is under-estimated by the model from year 8 onwards. Similarly the number of individuals sill alive and free of CAV (State 1) is over-estimated by the model for year 10

onwards.

Such discrepancies could have many causes. One possibility is that the transition rates vary with the time since the beginning of the process, the age of the patient, or some other omitted covariate, so that the Markov model is *non-homogeneous*. This could be accounted for by modelling the intensity as a function of age, for example, such as a piecewise-constant function. In this example, it is likely that the hazard of death increases with age, so the model underestimates the number of deaths when forecasting far into the future.

Another cause of poor model fit may sometimes be the failure of the Markov assumption. That is, the transition intensities may depend on the time spent in the current state (a semi-Markov process) or other characteristics of the process history. Accounting for the process history is difficult as the process is only observed through a series of snapshots. For a multi-state model with one-way progression through states, and frequent observations, we may be able to estimate the time spent in each state by each individual.

2.13 Fitting misclassification models with *msm*

In fact, in the heart transplant example from section 2.2, it is not medically realistic for patients to recover from a diseased state to a healthy state. Progression of coronary artery vasculopathy is thought to be an irreversible process. The angiography scan for CAV is actually subject to error, which leads to some false measurements of CAV states and apparent recoveries. Thus we account for misclassification by fitting a *hidden Markov model* using *msm*. Firstly we replace the two-way multi-state model by a one-way model with transition intensity matrix

$$Q = \begin{pmatrix} -(q_{12}+q_{14}) & q_{12} & 0 & q_{14} \\ 0 & -(q_{23}+q_{24}) & q_{23} & q_{24} \\ 0 & 0 & -q_{34} & q_{34} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

We also assume that true state 1 (CAV-free) can be classified as state 1 or 2, state 2 (mild/moderate CAV) can be classified as state 1, 2 or 3, while state 3 (severe CAV) can be classified as state 2 or 3. Recall that state 4 represents death. Thus our matrix of misclassification probabilities is

$$E = \begin{pmatrix} 1-e_{12} & e_{12} & 0 & 0 \\ e_{21} & 1-e_{21}-e_{23} & e_{23} & 0 \\ 0 & e_{32} & 1-e_{32} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

with underlying states as rows, and observed states as columns.

To model observed states with misclassification, we define an matrix `ematrix` indicating the states that can be misclassified. Rows of this matrix correspond to true states, columns to observed states. It should contains zeroes in the positions where misclassification is not permitted. Non-zero entries are initial values for the corresponding misclassification probabilities. We then call `msm` as before, but with this matrix as the `ematrix` argument. Initial values of 0.1 are assumed for each of the four misclassification probabilities $e_{12}, e_{21}, e_{23}, e_{32}$. Zeroes are given where the elements of E are zero. The diagonal elements supplied in `ematrix` are ignored, as rows must sum to one. The matrix `qmatrix`, specifying permitted transition intensities and their initial values, also changes to correspond to the new Q representing the progression-only model for the underlying states.

We use an alternative quasi-Newton optimisation algorithm (`method="BFGS"`) which can often be faster or more robust than the default Nelder-Mead simplex-based algorithm. An optional argument `initprobs` could also have been given here, representing the vector f of the probabilities of occupying each true state at the initial observation (equation 13). If not given, all individuals are assumed to be in true state 1 at their initial observation.

Model 5: multi-state model with misclassification

```
> oneway4.q <- rbind(c(0, 0.148, 0, 0.0171), c(0,
+     0, 0.202, 0.081), c(0, 0, 0, 0.126), c(0,
+     0, 0, 0))
> ematrix <- rbind(c(0, 0.1, 0, 0), c(0.1, 0, 0.1,
+     0), c(0, 0.1, 0, 0), c(0, 0, 0, 0))
> heartmisc.msm <- msm(state ~ years, subject = PTNUM,
+     data = heart, qmatrix = oneway4.q, ematrix = ematrix,
+     death = 4, method = "BFGS")
> heartmisc.msm

Call:
msm(formula = state ~ years, subject = PTNUM, data = heart, qmatrix = oneway4.q,

Maximum likelihood estimates:
Transition intensity matrix

        State 1                  State 2
State 1 -0.142 (-0.1599,-0.1262) 0.1014 (0.08663,0.1186)
State 2 0                        -0.2607 (-0.3162,-0.2149)
State 3 0                        0
State 4 0                        0
        State 3                   State 4
State 1 0                         0.04068 (0.03253,0.05088)
State 2 0.2267 (0.1691,0.3041)    0.03394 (0.008598,0.134)
State 3 -0.3085 (-0.3906,-0.2436) 0.3085 (0.2436,0.3906)
State 4 0                         0

Misclassification matrix

        State 1                State 2
State 1 0.9923 (0.9822,0.9967) 0.007662 (0.003253,0.01794)
State 2 0.245 (0.1778,0.3276)  0.7039 (0.6517,0.7513)
State 3 0                      0.1244 (0.06253,0.2322)
State 4 0                      0
        State 3                   State 4
State 1 0                         0
State 2 0.05105 (0.02966,0.08648) 0
State 3 0.8756 (0.7841,0.9317)    0
```

```
State 4 0                          1 (1,1)

-2 * log-likelihood:  3952
```

Thus there is an estimated probability of about 0.01 that mild/moderate CAV will be diagnosed erroneously, but a rather higher probability of 0.24 that underlying mild/moderate CAV will be diagnosed as CAV-free. Between the two CAV states, the mild state will be misdiagnosed as severe with a probability of 0.05, and the severe state will be misdiagnosed as mild with a probability of 0.12.

The model also estimates the progression rates through underlying states. An average of 7 years is spent disease-free, an average of 3.8 years is spent with mild/moderate disease, and periods of severe disease last 3.2 years on average before death.

2.14 Effects of covariates on misclassification rates

We can investigate how the probabilities of misclassification depend on covariates in a similar way to the transition intensities, using a `misccovariates` argument to `msm`. For example, we now include female sex as a covariate for the misclassification probabilities. This requires an extra four initial values for the linear effect for each of the logit-probabilities, which we set to zero.

Model 6: misclassification model with misclassification probabilities modelled on sex

```
> miscinits <- c(0.148, 0.0171, 0.202, 0.081, 0.126,
+     0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0)

> heartmiscsex.msm <- msm(state ~ years, subject = PTNUM,
+     data = heart, qmatrix = oneway4.q, ematrix = ematrix,
+     death = 4, misccovariates = ~sex, method = "BFGS")

> heartmiscsex.msm

Call:
msm(formula = state ~ years, subject = PTNUM, data = heart, qmatrix = oneway4.q,

Maximum likelihood estimates:
Transition intensity matrix

        State 1                    State 2
State 1 -0.1419 (-0.1603,-0.1256) 0.1014 (0.08606,0.1194)
State 2 0                          -0.2679 (-0.3255,-0.2204)
State 3 0                          0
State 4 0                          0
        State 3                    State 4
State 1 0                          0.04053 (0.03236,0.05077)
State 2 0.2331 (0.1737,0.3127)     0.03478 (0.008897,0.136)
State 3 -0.3027 (-0.3835,-0.2388) 0.3027 (0.2388,0.3835)
State 4 0                          0
```

```
Misclassification matrix

        State 1                State 2
State 1 0.9947 (0.9126,0.9997) 0.005319 (0.0002941,0.08859)
State 2 0.2513 (0.1741,0.3483) 0.6971 (0.6367,0.7514)
State 3 0                      0.1462 (0.07781,0.2578)
State 4 0                      0
        State 3                   State 4
State 1 0                         0
State 2 0.05156 (0.02974,0.08795) 0
State 3 0.8538 (0.7615,0.9144)    0
State 4 0                         1 (1,1)

Logit-linear effects of sex

        State 1               State 2
State 1 0                     -5.615 (-29.06,17.83)
State 2 1.327 (-0.3563,3.011) 0
State 3 0                     1.936 (0.1946,3.678)
State 4 0                     0
        State 3                State 4
State 1 0                      0
State 2 -0.6299 (-2.744,1.484) 0
State 3 0                      0
State 4 0                      0

-2 * log-likelihood:  3945
```

Considering the large confidence intervals for the estimates, we do not see any significant effect of sex on the fitted misclassification probabilities, so that men are no more or less likely than women to have an inaccurate angiography scan.

2.15 Extractor functions

As well as the functions described in section 2.9 for extracting useful information from fitted models, there are a number of extractor functions specific to models with misclassification.

Misclassification matrix The function `ematrix.msm` gives the estimated misclassification probability matrix at the given covariate values. For illustration, the fitted misclassification probabilities for men and women in model 6 are given by

```
> ematrix.msm(heartmiscsex.msm, covariates = list(sex = 0))

        State 1                State 2
State 1 0.9896 (0.9776,0.9952) 0.01039 (0.004753,0.02257)
State 2 0.2225 (0.1546,0.3093) 0.7221 (0.6665,0.7716)
State 3 0                      0.1195 (0.05971,0.2247)
```

```
State 4 0                        0
        State 3                    State 4
State 1 0                          0
State 2 0.05539 (0.0316,0.09533) 0
State 3 0.8805 (0.7907,0.935)    0
State 4 0                          1 (1,1)
```

```
> ematrix.msm(heartmiscsex.msm, covariates = list(sex = 1))
```

```
        State 1                State 2
State 1 1 (1.749e-06,1)        3.823e-05 (2.555e-15,1)
State 2 0.519 (0.1673,0.8528) 0.4507 (0.2794,0.6345)
State 3 0                      0.4847 (0.1572,0.8259)
State 4 0                      0
        State 3                    State 4
State 1 0                          0
State 2 0.03029 (0.004057,0.1932) 0
State 3 0.5153 (0.316,0.7099)      0
State 4 0                          1 (1,1)
```

although these are not useful in this situation as there was no significant gender difference in angiography accuracy. The standard errors for the estimates for women are higher, since there are only 87 women in this set of 622 patients.

Odds ratios for misclassification The function `odds.msm` gives the estimated odds ratios corresponding to each covariate effect on the misclassification probabilities.

```
> odds.msm(heartmiscsex.msm)
```

```
$sex
                           OR        L        U
Obs State 1 | State 2 3.77019 7.00e-01 2.03e+01
Obs State 2 | State 1 0.00364 2.40e-13 5.52e+07
Obs State 2 | State 3 6.93300 1.21e+00 3.96e+01
Obs State 3 | State 2 0.53263 6.43e-02 4.41e+00
```

underlining the lack of evidence for any gender difference in misclassification.

Observed and expected prevalences The function `prevalence.msm` is intended to assess the goodness of fit of the hidden Markov model for the *observed* states to the data. Tables of observed prevalences of observed states are calculated as described in section 2.12, by assuming that observed states are retained between observation times.

The expected numbers of individuals in each observed state are calculated similarly. Suppose the process begins at a common time for all individuals, and at this time, the probability of occupying *true* state r is f_r. Then given $n(t)$ individuals under observation at time t, the expected number of individuals in true state r at time t is the rth element of the vector $n(t)fP(t)$. Thus the expected number of individuals in *observed* state r is the rth element of the vector $n(t)fP(t)E$, where E is the misclassification probability matrix.

The expected prevalences (not shown) for this example are similar to those forecasted by the model without misclassification, with underestimates of the rates of death from 8 years onwards. To improve this model's long-term prediction ability, it is probably necessary to account for the natural increase in the hazard of death from any cause as people become older.

2.16 Recreating the path through underlying states

In speech recognition and signal processing, *decoding* is the procedure of determining the underlying states that are most likely to have given rise to the observations. The most common method of reconstructing the most likely state path is the *Viterbi* algorithm. Originally proposed by Viterbi [**?**], it is also described by Durbin *et al.* [**?**] and Macdonald and Zucchini [**?**] for discrete-time hidden Markov chains. For continuous-time models it proceeds as follows. Suppose that a hidden Markov model has been fitted and a Markov transition matrix $P(t)$ and misclassification matrix E are known. Let $v_i(k)$ be the probability of the most probable path ending in state k at time t_i.

1. Estimate $v_k(t_1)$ using known or estimated initial-state occupation probabilities.
2. For $i = 1 \ldots N$, calculate $v_l(t_i) = e_{l,O_{t_i}} \max_k v_k(t_{i-1}) P_{kl}(t_i - t_{i-1})$. Let $K_i(l)$ be the maximising value of k.
3. At the final time point N, the most likely underlying state S_N^* is the value of k which maximises $v_k(T_N)$.
4. Retrace back through the time points, setting $S_{i-1}^* = K_i(S_i^*)$.

The computations should be done in log space to prevent underflow. The *msm* package provides the function `viterbi.msm` to implement this method. For example, the following is an extract from a result of calling `viterbi.msm` to determine the most likely underlying states for all patients. The results for patient 100103 are shown, who appeared to 'recover' to a less severe state of disease while in state 3. We assume this is not biologically possible for the true states, so we expect that either the observation of state 3 at time 4.98 was an erroneous observation of state 2, or their apparent state 2 at time 5.94 was actually state 3. According to the expected path constructed using the Viterbi algorithm, it is the observation at time 5.94 which is most probably misclassified.

```
> vit <- viterbi.msm(heartmisc.msm)
> vit[vit$subject == 100103, ]

    subject time observed fitted
567  100103 0.00        1      1
568  100103 2.04        1      1
569  100103 4.08        2      2
570  100103 4.98        3      3
571  100103 5.94        2      3
572  100103 7.01        3      3
573  100103 8.05        3      3
574  100103 8.44        4      4
```

2.17 Fitting general hidden Markov models with *msm*

The *msm* package provides a framework for fitting continuous-time hidden Markov models with general, continuous outcomes. As before, we use the `msm` function itself.

Specifying the hidden Markov model A hidden Markov model consists of two related components:

- the model for the evolution of the underlying Markov chain,
- the set of models for the observed data conditionally on each underlying state.

The model for the transitions between underlying states is specified as before, by supplying a `qmatrix`. The model for the outcomes is specified using the argument `hmodel` to `msm`. This is a list, with one element for each underlying state, in order. Each element of the list should be an object returned by a hidden Markov model *constructor function*. The HMM constructor functions provided with *msm* are listed in Table 1. There is a separate constructor function for each class of outcome distribution, such as uniform, normal or gamma.

Consider a three-state hidden Markov model, with a transition intensity matrix of

$$Q = \begin{pmatrix} -q_{12} & q_{12} & 0 \\ 0 & -q_{23} & q_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

Suppose the outcome distribution for state 1 is Normal(μ_1, σ_1^2), the distribution for state 2 is Normal(μ_2, σ_2^2), and state 3 is exactly observed. Observations of state 3 are given a label of -9 in the data. Here our `hmodel` argument should be a list of objects returned by `hmmNorm` and `hmmIdent` constructor functions.

We must specify initial values for the parameters as the arguments to the constructor functions. For example, we take initial values of $\mu_1 = 90, \sigma_1 = 8, \mu_2 = 70, \sigma_2 = 8$. Initial values for q_{12} and q_{23} are 0.25 and 0.2. Finally suppose the observed data are in a variable called `y`, the measurement times are in `time`, and subject identifiers are in `ptnum`. The call to `msm` to estimate the parameters of this hidden Markov model would then be

```
msm ( y ~ time, subject=ptnum, data = example.df,
qmatrix = rbind( c(0, 0.25, 0), c(0, 0, 0.2), c(0, 0, 0)),
hmodel = list (hmmNorm(mean=90, sd=8), hmmNorm(mean=70, sd=8),
               hmmIdent(-9)) )
```

Covariates on hidden Markov model parameters Most of the outcome distributions can be parameterised by covariates, using a link-transformed linear model. For example, an observation y_{ij} may have distribution f_1 conditionally on underlying state 1. The link-transformed parameter θ_1 is a linear function of the covariate vector x_{ij} at the same observation time.

$$\begin{aligned} y_{ij}|S_{ij} &\sim f_1(y|\theta_1, \gamma_1) \\ g(\theta_1) &= \alpha + \beta^T x_{ij} \end{aligned}$$

Function	Distribution	Parameters		Location (link)	Density for an observation x
`hmmCat`	Categorical	`prob, basecat`	p, c_0	p (logit)	$p_x, x = 1, \ldots, n$
`hmmIdent`	Identity	`x`	x_0		$I_{x=x_0}$
`hmmUnif`	Uniform	`lower, upper`	l, u		$1/(u-l), u \leq x \leq l$
`hmmNorm`	Normal	`mean, sd`	μ, σ	μ (identity)	$\phi(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-(x-\mu)^2/(2\sigma^2))$
`hmmLNorm`	Log-normal	`meanlog, sdlog`	μ, σ	μ (identity)	$\frac{1}{x\sqrt{2\pi\sigma^2}} \exp(-(\log x - \mu)^2/(2\sigma^2))$
`hmmExp`	Exponential	`rate`	λ	λ (log)	$\lambda e^{-\lambda x}, x > 0$
`hmmGamma`	Gamma	`shape, rate`	n, λ	λ (log)	$\frac{\lambda^n}{\Gamma(n)} x^{n-1} \exp(-\lambda x), x > 0, n > 0, \lambda > 0$
`hmmWeibull`	Weibull	`shape, scale`	a, b	b (log)	$\frac{a}{b}(\frac{x}{b})^{a-1} \exp(-(\frac{x}{b})^a), x > 0$
`hmmPois`	Poisson	`rate`	λ	λ (log)	$\lambda^x \exp(-\lambda)/x!, x = 0, 1, 2, \ldots$
`hmmBinom`	Binomial	`size, prob`	n, p	p (logit)	$\binom{n}{x} p^x (1-p)^{n-x}$
`hmmNBinom`	Negative binomial	`disp, prob`	n, p	p (logit)	$\Gamma(x+n)/(\Gamma(n)x!)p^n(1-p)^x$
`hmmTNorm`	Truncated normal	`mean, sd, lower, upper`	μ, σ, l, u	μ (identity)	$\phi(x, \mu, \sigma)/$ $(\Phi(u, \mu, \sigma) - \Phi(l, \mu, \sigma))$, where $\Phi(x, \mu, \sigma) = \int_{-\infty}^{x} \phi(u, \mu, \sigma) du$
`hmmMETNorm`	Normal with truncation and measurement error	`mean, sd, lower, upper, sderr, meanerr`	$\mu_0, \sigma_0, l, u, \sigma_\epsilon, \mu_\epsilon$	μ_ϵ (identity)	$(\Phi(u, \mu_2, \sigma_3) - \Phi(l, \mu_2, \sigma_3))/$ $(\Phi(u, \mu_0, \sigma_0) - \Phi(l, \mu_0, \sigma_0))$ $\times\phi(x, \mu_0 + \mu_\epsilon, \sigma_2)$, $\sigma_2^2 = \sigma_0^2 + \sigma_\epsilon^2$, $\sigma_3 = \sigma_0\sigma_\epsilon/\sigma_2$, $\mu_2 = (x - \mu_\epsilon)\sigma_0^2 + \mu_0\sigma_\epsilon^2$
`hmmMEUnif`	Uniform with measurement error	`lower, upper, sderr, meanerr`	$l, u, \mu_\epsilon, \sigma_\epsilon$	μ_ϵ (identity)	$(\Phi(x, \mu_\epsilon + l, \sigma_\epsilon) - \Phi(x, \mu_\epsilon + u, \sigma_\epsilon))/$ $(u - l)$

Table 1: Hidden Markov model distributions in *msm*.

Specifically, parameters named as the "Location" parameter in Table 1 can be modelled in terms of covariates, with the given link function.

The `hcovariates` argument to `msm` specifies the model for covariates on the hidden Markov outcome distributions. This is a list of the same length as the number of underlying states, and the same length as the `hmodel` list. Each element of the list is a formula, in standard R linear model notation, defining the covariates on the distribution for the corresponding state. If there are no covariates for a certain hidden state, then insert a `NULL` in the corresponding place in the list. For example, in the three-state normal-outcome example above, suppose that the normal means on states 1 and 2 are parameterised by a single covariate x.

$$\mu_1 = \alpha_1 + \beta_1 x_{ij}, \qquad \mu_2 = \alpha_2 + \beta_2 x_{ij}.$$

The equivalent call to `msm` would be

```
msm ( state ~ time, subject=ptnum, data = example.df,
qmatrix = rbind( c(0, 0.25, 0), c(0, 0, 0.2), c(0, 0, 0)),
hmodel = list (hmmNorm(mean=90, sd=8), hmmNorm(mean=70, sd=8),
               hmmIdent(-9)),
hcovariates = list ( ~ x, ~ x, NULL)
).
```

Constraints on hidden Markov model parameters Sometimes it is realistic that parameters are shared between some of the state-specific outcome distributions. For example, the Normally-distributed outcome in the previous example could have a common variance $\sigma_1^2 = \sigma_2^2 = \sigma^2$ between states 1 and 2, but differing means. It would also be realistic for any covariates on the mean to have a common effect $\beta_1 = \beta_2 = \beta$ on the state 1 and 2 outcome distributions.

The argument `hconstraint` to `msm` specifies which hidden Markov model parameters are constrained to be equal. This is a named list. Each element is a vector of constraints on the named hidden Markov model parameter. The vector has length equal to the number of times that class of parameter appears in the whole model. As for the other constraint arguments such as `qconstraint`, identical values of this vector indicate parameters constrained to be equal.

For example consider the three-state hidden Markov model described above, with normally-distributed outcomes for states 1 and 2. To constrain the outcome variance to be equal for states 1 and 2, and to also constrain the effect of `x` on the outcome mean to be equal for states 1 and 2, specify

```
hconstraint = list(sd = c(1,1), x=c(1,1))
```

FEV$_1$ after lung transplants Now we give an example of fitting a hidden Markov model to a real dataset. The data on FEV$_1$ measurements from lung transplant recipients, described in 1.6.4, are provided with the *msm* package in a dataset called `fev`.

```
> data(fev)
```

We fit models Models 1 and 2, each with three states and common Q matrix.

```
> three.q <- rbind(c(0, exp(-6), exp(-9)), c(0,
+     0, exp(-6)), c(0, 0, 0))
```

The simpler Model 1 is specified as follows. Under this model the FEV_1 outcome is Normal with unknown mean and variance, and the mean and variance are different between BOS state 1 and state 2. `hcovariates` specifies that the mean of the Normal outcome depends linearly on acute events. Specifically, this covariate is an indicator for the occurrence of an acute event within 14 days of the observation, denoted `acute` in the data. As an initial guess, we suppose the mean FEV_1 is 100% baseline in state 1, and 54% baseline in state 2, with corresponding standard deviations 16 and 18, and FEV_1 observations coinciding with acute events are on average 8% baseline lower. `hconstraint` specifies that the acute event effect is equal between state 1 and state 2.

Days of death are coded as 999 in the `fev` outcome variable.

```
> hmodel1 <- list(hmmNorm(mean = 100, sd = 16),
+     hmmNorm(mean = 54, sd = 18), hmmIdent(999))
> fev1.msm <- msm(fev ~ days, subject = ptnum, data = fev,
+     qmatrix = three.q, death = 3, hmodel = hmodel1,
+     hcovariates = list(~acute, ~acute, NULL),
+     hcovinits = list(-8, -8, NULL), hconstraint = list(acute = c(1,
+         1)), method = "BFGS")
> fev1.msm

Call:
msm(formula = fev ~ days, subject = ptnum, data = fev, qmatrix = three.q,     hm

Maximum likelihood estimates:
Transition intensity matrix

        State 1
State 1 -0.0007038 (-0.0008333,-0.0005945)
State 2 0
State 3 0
        State 2
State 1 0.0006275 (0.0005201,0.0007572)
State 2 -0.0008011 (-0.001013,-0.0006337)
State 3 0
        State 3
State 1 7.631e-05 (3.967e-05,0.0001468)
State 2 0.0008011 (0.0006337,0.001013)
State 3 0

Hidden Markov model, 3 states
Initial state occupancy probabilities: 1,0,0

State 1 - normal distribution
Parameters:
      estimate   l95   u95
mean      98.0 97.34 98.67
sd        16.2 15.78 16.60
```

```
acute     -8.8 -9.95 -7.63

State 2 - normal distribution
Parameters:
      estimate   l95   u95
mean      51.8 50.76 52.88
sd        17.7 17.08 18.29
acute     -8.8 -9.95 -7.63

State 3 - identity distribution
Parameters:
      estimate l95 u95
which      999  NA  NA


-2 * log-likelihood:  51598
```

```
> sojourn.msm(fev1.msm)

        estimates  SE    L    U
State 1      1421 122 1200 1682
State 2      1248 149  987 1578
```

Printing the *msm* object `fev1.msm` shows estimates and confidence intervals for the transition intensity matrix and the hidden Markov model parameters. The estimated within-state means of FEV_1 are around 98% and 52% baseline respectively. From the estimated transition intensities, individuals spend around 1421 days (3.9 years) before getting BOS, after which they live for an average of 1248 days (3.4 years). FEV_1 is lower by an average of 8% baseline within 14 days of acute events.

Model 2, where the outcome distribution is a more complex two-level model, is specified as follows. We use the distribution defined by equations 15–16. The `hmmMETNorm` constructor defines the truncated normal outcome with an additional normal measurement error. The explicit probability density for this distribution is given in Table 1.

Our initial values are again 100 and 54 for the means of the within-state distribution of *underlying* FEV_1, and 16 and 18 for the standard errors. This time, underlying FEV_1 is truncated normal. The truncation limits `lower` and `upper` are not estimated. We take an initial measurement error standard deviation of `sderr=8`. The extra shift `meanerr` in the measurement error model is fixed to zero and not estimated.

The `hconstraint` specifies that the measurement error variance σ_ϵ^2 is equal between responses in states 1 and 2, as is the effect of short-term acute events on the FEV_1 response.

```
> hmodel2 <- list(hmmMETNorm(mean = 100, sd = 16,
+     sderr = 8, lower = 80, upper = Inf, meanerr = 0),
+     hmmMETNorm(mean = 54, sd = 18, sderr = 8,
+         lower = 0, upper = 80, meanerr = 0), hmmIdent(999))
> fev2.msm <- msm(fev ~ days, subject = ptnum, data = fev,
+     qmatrix = three.q, death = 3, hmodel = hmodel2,
+     hcovariates = list(~acute, ~acute, NULL),
```

```
+     hcovinits = list(-8, -8, NULL), hconstraint = list(sderr = c(1,
+         1), acute = c(1, 1)), method = "BFGS")
> fev2.msm

Call:
msm(formula = fev ~ days, subject = ptnum, data = fev, qmatrix = three.q,     hm

Maximum likelihood estimates:
Transition intensity matrix

        State 1
State 1 -0.0007629 (-0.0008978,-0.0006483)
State 2 0
State 3 0
        State 2
State 1 0.0006912 (0.000578,0.0008266)
State 2 -0.0007507 (-0.0009445,-0.0005967)
State 3 0
        State 3
State 1 7.17e-05 (3.563e-05,0.0001443)
State 2 0.0007507 (0.0005967,0.0009445)
State 3 0

Hidden Markov model, 3 states
Initial state occupancy probabilities: 1,0,0

State 1 - metruncnorm distribution
Parameters:
        estimate    l95   u95
mean       88.70  85.56 91.84
sd         18.99  17.40 20.72
lower      80.00     NA    NA
upper        Inf     NA    NA
sderr       8.88   8.42  9.38
meanerr     0.00     NA    NA
acute     -10.18 -11.24 -9.11

State 2 - metruncnorm distribution
Parameters:
        estimate    l95   u95
mean       59.65  56.96 62.34
sd         20.96  19.08 23.01
lower       0.00     NA    NA
upper      80.00     NA    NA
sderr       8.88   8.42  9.38
meanerr     0.00     NA    NA
```

```
acute     -10.18 -11.24 -9.11

State 3 - identity distribution
Parameters:
      estimate l95 u95
which      999  NA  NA


-2 * log-likelihood:  51411
```

```
> sojourn.msm(fev2.msm)

        estimates  SE    L    U
State 1      1311 109 1114 1543
State 2      1332 156 1059 1676
```

Under this model the standard deviation of FEV_1 measurements caused by measurement error (more realistically, natural short-term fluctuation) is around 9% baseline. The estimated effect of acute events on FEV_1 and sojourn times in the BOS-free state and in BOS before death are similar to Model 1.

The following plot illustrates a trajectory of declining FEV_1 from the first lung transplant recipient in this dataset. This is produced by the following R code. The Viterbi algorithm is used to locate the most likely point at which this individual moved from BOS state 1 to BOS state 2, according to the fitted Model 2. This is illustrated by the vertical dotted line. This is the point at which the individual's lung function started to remain consistently below 80% baseline FEV_1.

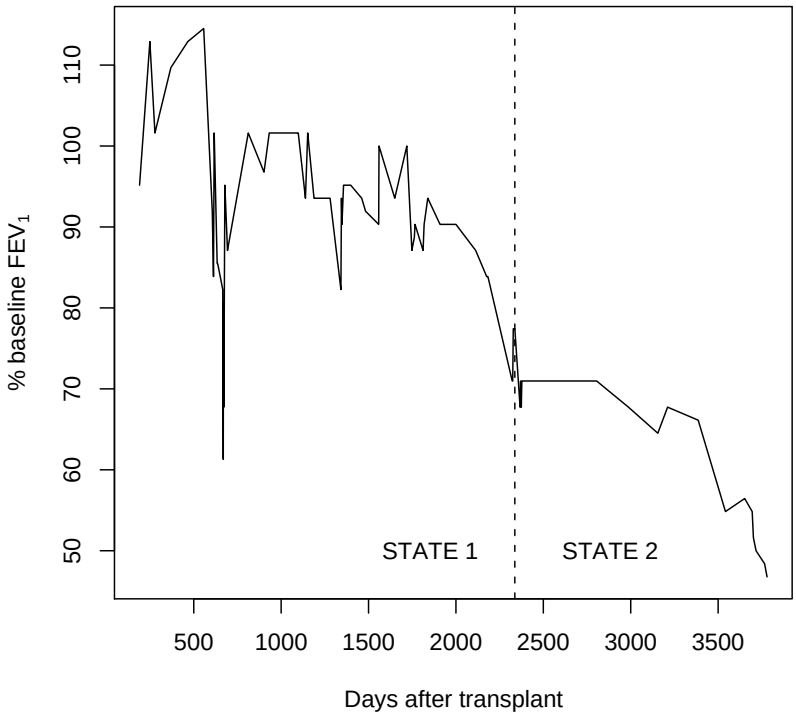
```
> keep <- fev$ptnum == 1 & fev$fev < 999
> plot(fev$days[keep], fev$fev[keep], type = "l",
+     ylab = expression(paste("% baseline ", FEV[1])),
+     xlab = "Days after transplant")
> vit <- viterbi.msm(fev2.msm)[keep, ]
> (max1 <- max(vit$time[vit$fitted == 1]))

[1] 2337

> (min2 <- min(vit$time[vit$fitted == 2]))

[1] 2367

> abline(v = mean(max1, min2), lty = 2)
> text(max1 - 500, 50, "STATE 1")
> text(min2 + 500, 50, "STATE 2")
```



An alternative way of specifying a misclassification model This general framework for specifying hidden Markov models can also be used to specify multi-state models with misclassification. A misclassification model is a hidden Markov model with a categorical outcome distribution. So instead of an `ematrix` argument to `msm`, we can use a `hmodel` argument with `hmmCat` constructor functions.

`hmmCat` takes at least one argument `prob`, a vector of probabilities of observing outcomes of $1, 2, \ldots, n$ respectively, where n is the length of `prob`. All outcome probabilities with an initial value of zero are assumed to be fixed at zero. `prob` is scaled if necessary to sum to one.

The model in section 2.13 specifies that an individual occupying underlying state 1 can be observed as states 2 (and 1), underlying state 2 can be observed as states 1, 2 or 3, and state 3 can be observed as states 2 or 3, and underlying state 4 (death) cannot be misclassified. Initial values of 0.1 are given for the 1-2, 2-1, 2-3 and 3-2 misclassification probabilities.

This is equivalent to the model below, specified using a `hmodel` argument to `msm`. The maximum likelihood estimates should be the same as before (Model 5).

```
> oneway4.q <- rbind(c(0, 0.148, 0, 0.0171), c(0,
+     0, 0.202, 0.081), c(0, 0, 0, 0.126), c(0,
+     0, 0, 0))
```

```
> heartmisc.msm <- msm(state ~ years, subject = PTNUM,
+     data = heart, hmodel = list(hmmCat(c(0.9,
+         0.1, 0, 0)), hmmCat(c(0.1, 0.8, 0.1, 0)),
+         hmmCat(c(0, 0.1, 0.9, 0)), hmmIdent(4)),
+     qmatrix = oneway4.q, death = 4, method = "BFGS")
> heartmisc.msm

Call:
msm(formula = state ~ years, subject = PTNUM, data = heart, qmatrix = oneway4.q,

Maximum likelihood estimates:
Transition intensity matrix

        State 1                  State 2
State 1 -0.142 (-0.1599,-0.1262) 0.1014 (0.08663,0.1186)
State 2 0                        -0.2607 (-0.3162,-0.2149)
State 3 0                        0
State 4 0                        0
        State 3                   State 4
State 1 0                         0.04068 (0.03253,0.05088)
State 2 0.2267 (0.1691,0.3041)    0.03394 (0.008598,0.134)
State 3 -0.3085 (-0.3906,-0.2436) 0.3085 (0.2436,0.3906)
State 4 0                         0

Hidden Markov model, 4 states
Initial state occupancy probabilities: 1,0,0,0

State 1 - categorical distribution
Parameters:
     estimate     l95    u95
P(1)  0.99234      NA     NA
P(2)  0.00766 0.00325 0.0179
P(3)  0.00000      NA     NA
P(4)  0.00000      NA     NA

State 2 - categorical distribution
Parameters:
     estimate    l95    u95
P(1)    0.245 0.1778 0.3276
P(2)    0.704     NA     NA
P(3)    0.051 0.0297 0.0865
P(4)    0.000     NA     NA

State 3 - categorical distribution
Parameters:
     estimate    l95   u95
```

```
P(1)    0.000     NA    NA
P(2)    0.124 0.0625 0.232
P(3)    0.876     NA    NA
P(4)    0.000     NA    NA

State 4 - identity distribution
Parameters:
      estimate l95 u95
which        4  NA  NA


-2 * log-likelihood:  3952
```

2.17.1 Defining a new hidden Markov model distribution

Suppose the hidden Markov model outcome distributions supplied with *msm* (Table 1) are insufficient. We want to define our own univariate distribution, called `hmmNewDist`, taking two parameters `location` and `scale`. Download the source package, for example `msm-0.5.tar.gz` for version 0.5, from CRAN and edit the files in there, as follows.

1. Add an element to `.msm.HMODELPARS` in the file `R/constants.R`, naming the parameters of the distribution. For example

   ```
   newdist = c('location', 'scale')
   ```

2. Add a corresponding element to the C variable `HMODELS` in the file `src/lik.c`. This MUST be in the same position as in the `.msm.HMODELPARS` list. For example,

   ```
   hmmfn HMODELS[] = {
     ...,
     hmmNewDist
   };.
   ```

3. The new distribution is allowed to have one parameter which can be modelled in terms of covariates. Add the name of this parameter to the named vector `.msm.LOCPARS` in `R/constants.R`. For example `newdist = 'location'`. Specify `newdist = NA` if there is no such parameter.

4. Supposed we have specified a parameter with a non-standard name, that is, one which doesn't already appear in `.msm.HMODELPARS`. Standard names include, for example, `'mean'`, `'sd'`, `'shape'` or `'scale'`. Then we should add the allowed range of the parameter to `.msm.PARRANGES`. In this example, we add `meanpars = c(-Inf, Inf)` to `.msm.PARRANGES`. If the parameter should always be fixed during a maximum likelihood estimation, then add its name to `.msm.AUXPARS`.

5. Add an R constructor function for the distribution to `R/hmm-dists.R`. For a simple univariate distribution, this is of the form

```
hmmNewDist <- function(location, scale)
{
  hmmDIST (label = "newdist",
  link = "identity",
  r = function(n) rnewdist(n, location, scale),
  match.call())
}
```

- The `'label'` must be the same as the name you supplied for the new element of `.msm.HMODELPARS`
- `link` is the link function for modelling the location parameter of the distribution as a function of covariates. This should be the quoted name of an R function. A log link is `'log'` and a logit link is `'qlogis'`. If using a new link function other than `'identity'`, `'log'`, or `'qlogis'`, you should add its name to the vector `.msm.LINKFNS` in `R/constants.R`, and add the name of the corresponding inverse link to `.msm.INVLINK`. You should also add the names of these functions to the C array `LINKFNS` in `src/lik.c`, and write the functions if they do not already exist.
- `r` is an R function, of the above format, returning a vector of `n` random values from the distribution. You should write this if it doesn't already exist.

6. Add the name of the new constructor function to the NAMESPACE in the top-level directory of the source package.

7. Write a C function to compute the probability density of the distribution, and put this in `src/hmm.c`, with a declaration in `src/hmm.h`. This must be of the form

   ```
   double hmmNewDist(double x, double *pars)
   ```

   where `*pars` is a vector of the parameters of the distribution, and the density is evaluated at `x`.

8. Update the documentation (`man/hmm-dists.Rd` and the distribution table in `inst/doc/msm-manual.Rnw` if you like.

9. Recompile the package (see the "Writing R Extensions" manual)

Your new distribution will be available to use in the `hmodel` argument to `msm`, as, for example

```
hmodel = list (..., hmmNewDist(location = 0, scale = 1), ...)
```

If your distribution may be of interest to others, ask me nicely (`chris.jackson@imperial.ac.uk`) to include it in a future release.

3 *msm* reference guide

The R help page for `msm` gives details of all the allowed arguments and options to the `msm` function. To view this online in R, type:

```
> help(msm)
```

Similarly all the other functions in the package have help pages, which should always be consulted in case of doubt about how to call them. The web-browser based help interface may often be convenient - type

```
> help.start()
```

and navigate to Packages ... msm, which brings up a list of all the functions in the package with links to their documentation, and a link to this manual in PDF format.

A Changes in the msm package

Version 0.5 is a major update to *msm*, introducing hidden Markov models with general responses. Much of the underlying R and C code was re-organised or re-written. Some syntax changes were made to simplify the process of specifying models.

To convert code written for previous versions of msm to run in 0.5, the following changes will need to be made.

- Replace the `qmatrix` of ones and zeroes with a `qmatrix` containing initial values for transition intensities.
- Replace the `ematrix` of ones and zeroes with a `ematrix` containing initial values for misclassification probabilities.
- Put any initial values for covariate effects in `covinits` or `misccovinits` arguments. These are set to zero if not specified.
- Remove the `inits` argument.
- Remove unnecessary `misc = TRUE` for misclassification models.
- Replace any `fromto`-style datasets with proper longitudinal datasets with one row per observation time. `fromto` support has been withdrawn.
- `covmatch` has been abolished. If you were using `covmatch='next'`, covariates will need to be matched with observations by moving covariates one row back in the data.
- `crudeinits.msm` now takes an formula `state` $\sim$ `time` instead of two arguments `state`, `time`
- `simmulti.msm` takes an argument `covariates` instead of `beta`, for consistency with `msm`.

Saved model objects from previous versions will need to be generated again under 0.5 to work with the post-processing functions from 0.5.

For a more detailed list of the changes introduced in Version 0.5, see the `NEWS` file in the top-level directory of the installed package.

If you use *msm* in published work, please let me know, for my own interest! Or even if you just use it and find it helpful, whatever your field of application.