Multi-state modelling with R: the msm package

Version 1.7

27 November, 2022

Christopher Jackson
MRC Biostatistics Unit
Cambridge, U.K.
chris.jackson@mrc-bsu.cam.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 censored states.

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.

Much of the material in this manual is published, in a more concise form, in Journal of Statistical Software (2011) 38(8):1-29, http://www.jstatsoft.org/v38/i08/

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[1] for a thorough introduction to the theory of continuous-time Markov chains.

In a time-homogeneous continuous-time Markov model, a single period of occupancy (or *sojourn time*) in state r has an exponential distribution, with rate given by $-q_{rr}$, (or mean $-1/q_{rr}$). The remaining elements of the rth row of Q are *proportional to* the probabilities governing the next state after r to which the individual makes a transition. The probability that the individual's next move from state r is to state s is $-q_{rs}/q_{rr}$.

1.2 Disease progression models

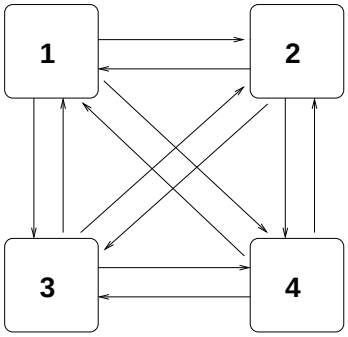
The development of the *msm* package was motivated by applications to disease modelling. Many chronic diseases have a natural interpretation in terms of staged progression. Multi-state Markov models in continuous time are often used to model the course of diseases. A commonly-used model is illustrated 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.*[2]), problems following lung transplantation (Jackson and Sharples[3]), problems following heart transplantation (Sharples[4], Klotz and Sharples[5]), hepatic cancer (Kay[6]), HIV infection and AIDS (Longini *et al.*[7], Satten and Longini[8], Guihenneuc-Jouyaux *et al.*[9], Gentleman *et al.*[10]), diabetic complications (Marshall and Jones[11], Andersen[12]), breast cancer screening (Duffy and Chen[13], Chen *et al.*[14]), cervical cancer screening (Kirby and Spiegelhalter[15]) and liver cirrhosis (Andersen *et al.*[16]). Many of these references also describe the mathematical theory, which will be reviewed in the following sections.

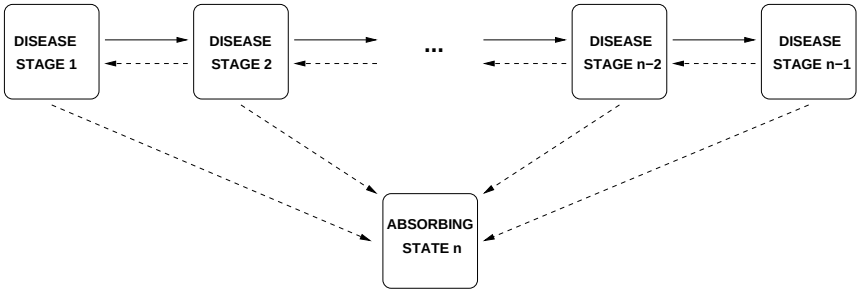
1.3 Arbitrary observation times

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 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



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

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.

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.*[17] 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. Note that *msm* does not deal with informative sampling times. See, e.g. [18] for some methods in this case, which require specialised programming.

1.4 Likelihood for the multi-state model

Kalbfleisch and Lawless[19] and later Kay [6] 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.

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

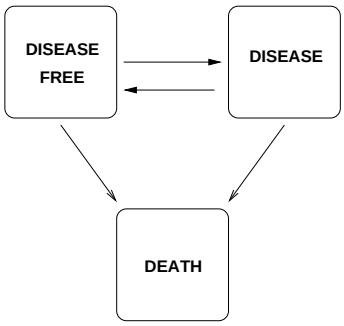


Figure 3: Illness-death model.

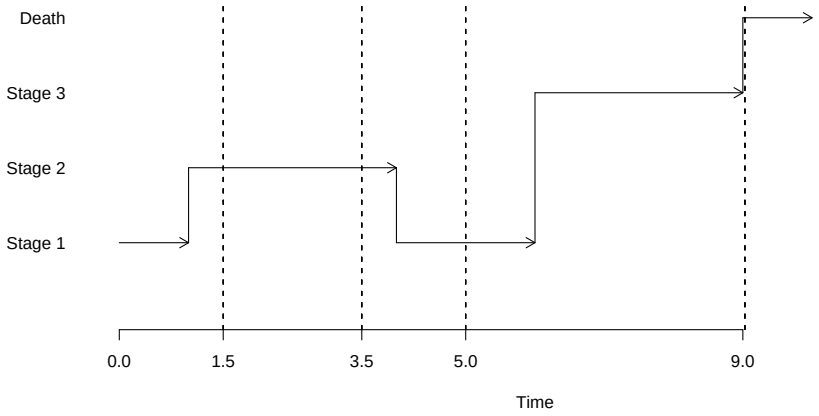


Figure 4: Evolution of a multi-state model. The process is observed on four occasions.

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 [1]).

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

The matrix exponential Exp is different from a scalar exponential. The exponential of a matrix is defined by the same "power series" $\mathrm{Exp}(X) = 1 + X^2/2! + X^3/3! + ...$ as the scalar exponential, except that each term X^k in the series is defined by matrix products, not element-wise scalar multiplication. It is notoriously difficult to calculate reliably, as discussed by Moler and van Loan [20]. 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

$$\begin{aligned}
p_{11}(t) &= e^{-(q_{12}+q_{13})t} \\
p_{12}(t) &= \begin{cases} \frac{q_{12}}{q_{12}+q_{13}-q_{23}}(e^{-q_{23}t} - e^{-(q_{12}+q_{13})t}) & (q_{12}+q_{13} \neq q_{23}) \\ q_{12}te^{(-(q_{12}+q_{13})t} & (q_{12}+q_{13} = q_{23}) \end{cases} \\
p_{13}(t) &= \begin{cases} 1 - e^{-(q_{12}+q_{13})t} - \frac{q_{12}}{q_{12}+q_{13}-q_{23}}(e^{-q_{23}t} - e^{-(q_{12}+q_{13})t}) & (q_{12}+q_{13} \neq q_{23}) \\ (-1 + e^{(q_{12}+q_{13})t} - q_{12}t)e^{-(q_{12}+q_{13})t} & (q_{12}+q_{13} = q_{23}) \end{cases} \\
p_{21}(t) &= 0 \\
p_{22}(t) &= e^{-q_{23}t} \\
p_{23}(t) &= 1 - e^{-q_{23}t} \\
p_{31}(t) &= 0 \\
p_{32}(t) &= 0 \\
p_{33}(t) &= 1
\end{aligned}$$

The *msm* package calculates $P(t)$ analytically for selected 2, 3, 4 and 5-state models, illustrated in Figures 5–8. For other models, which can have any transition structure on any number of states in principle, $P(t)$ is determined from the matrix exponential. This is calculated using eigensystem decomposition (if eigenvalues are distinct) or a method based on Padé approximants with scaling and squaring [20] (if there are repeated eigenvalues). 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 5 a) illustrates the model defined by either $Q = \begin{pmatrix} -q_{12} & q_{12} \\ 0 & 0 \end{pmatrix}$ or $Q = \begin{pmatrix} 0 & 0 \\ q_{21} & -q_{21} \end{pmatrix}$.

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

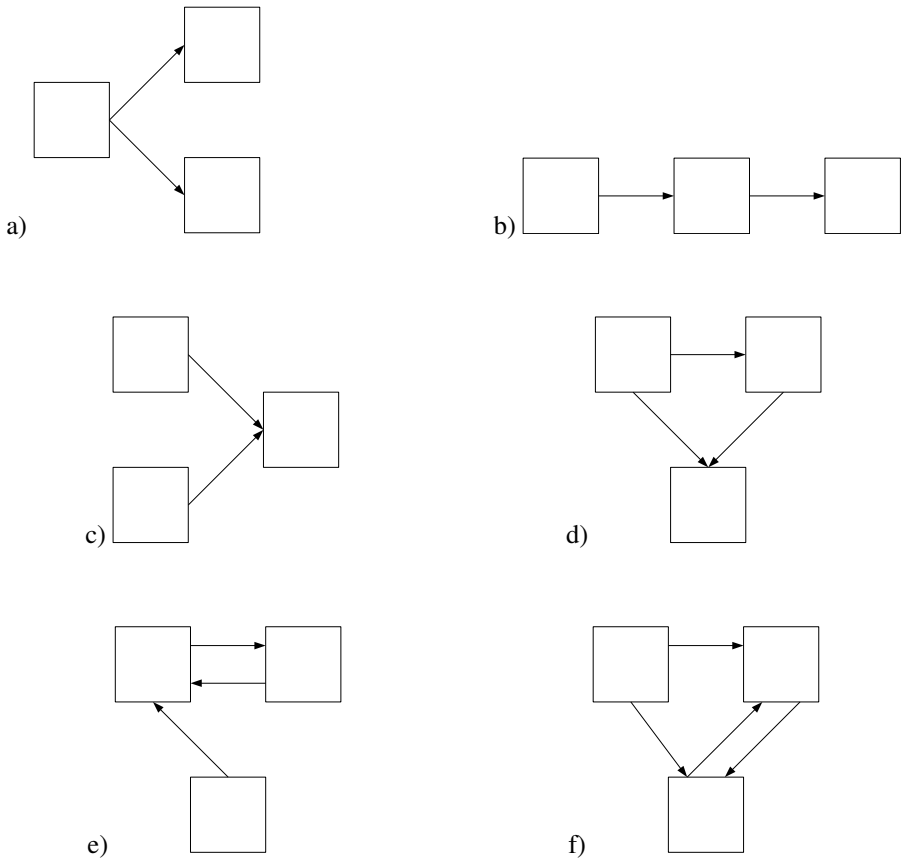


Figure 6: Three-state models fitted using analytic $P(t)$ matrices in *msm*. Implemented for all permutations of state labels 1, 2, 3.

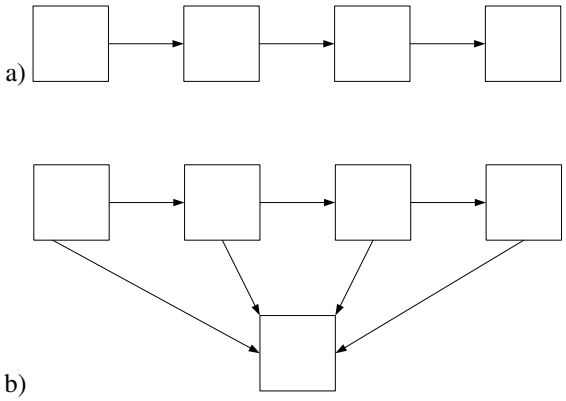


Figure 7: Four-state models fitted using analytic $P(t)$ matrices in *msm*. Implemented for all permutations of state labels 1, 2, 3, 4.

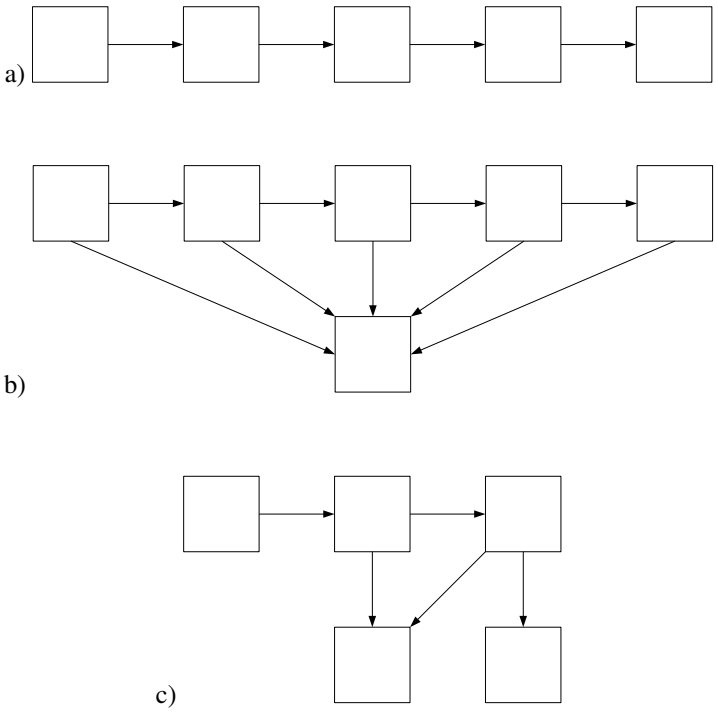


Figure 8: Five-state models fitted using analytic $P(t)$ matrices in *msm*. Implemented for all permutations of state labels 1, 2, 3, 4, 5.

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

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

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 $P(t)$.

Exactly-observed death times 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 instant before death:

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

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

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

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}. ***msm*** is restricted to Markov models, but much richer models are possible for this type of data. For example, Putter *et al.*[21] discussed the *mstate* software for semi-parametric multi-state models with non-parametric baseline hazards and Cox regression. The Markov assumption is restrictive but necessary in general to compute a likelihood for intermittently-observed processes.

Censored states A censored quantity is one whose exact value is unknown, but known to be in a certain interval. For example, in survival analysis, a death time is *right-censored* if the study ends and the patient is still alive, since the death time is known to be greater than the end time. In multi-state models for intermittently-observed processes, the times of changes of state are usually *interval censored*, known to be within bounded intervals. This leads to a likelihood based on equation 3.

In some circumstances, *states* may be censored as well as *event times*. For example, at the end of some chronic disease studies, patients are known to be alive but in an *unknown state*. For such a censored observation $S(t_{j+1})$ $(j + 1 = n)$ known only to be a state in the set C, the equivalent contribution to the likelihood is

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

Note that this special likelihood is not needed if the state is known at the end of the study. In this case, likelihood 3 applies. Although the *survival time* is censored, the *state* at the end of the study is not censored.

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.

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

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,

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

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 (panel data), exactly-observed transition times, exact death times 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 [11] 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. If the main response (the state of the Markov process) is not observed at the times when the covariate changes, it could be considered as a "censored" state (as in Section 1.4).

The *msm* package allows individual-specific or time dependent covariates to be fitted to transition intensities. In order to calculate transition probabilities $P(t)$ on which the likelihood depends, time-dependent covariates are assumed to be piecewise-constant. Models whose intensities change with time are called *time-inhomogeneous*. An important special case handled by *msm* is the model in which intensities change at a series of times common to each individual.

Marshall and Jones [11] 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, or whether the effect on backward transitions is equal to minus the effect on forward transitions.

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 [22] and the analysis of biological sequence data [23]. 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.

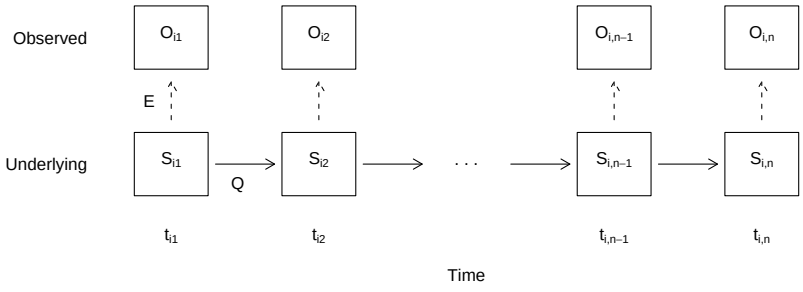


Figure 9: A hidden Markov model in continuous time. Observed states are generated conditionally on an underlying Markov process.

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 e_{rs} of misclassification as state s given underlying state r, a multinomial logistic regression model can be used:

$$\log \frac{e_{rs}(t)}{e_{rs_0}(t)} = \gamma_{rs}^T w(t). \tag{10}$$

where s_0 is some baseline state, usually chosen as the underlying state, or the state with the highest probability (for numerical stability).

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 [24, 25], is commonly used for hidden Markov model estimation in discrete-time applications. See, for example, Durbin *et al.*[23], Albert [26]. A generalisation of this algorithm to continuous time was described by Bureau *et al.*[27].

The *msm* package uses a direct method of calculating likelihoods for continuous-time models based on matrix products. This type of method has been described by Macdonald and Zucchini [28, pp. 77–79], Lindsey [29, p.73] and Guttorp [30]. Satten and Longini [8] 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

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

where the sum is taken over all possible paths of underlying states $S_{i1}, \ldots, S_{in_i}$. 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:

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

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 $j = 2, \ldots, n_i$ let T_{ij} be the $R \times R$ matrix (where R is the number of states) with (r, s) entry

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

Then subject i's likelihood contribution is

$$L_i = f T_{i2} T_{i3}, \ldots T_{in_i} \mathbf{1} \tag{13}$$

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 (r, s) entry of T_{ij} is then

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

Section 2.14 describes how to fit multi-state models with misclassification using the *msm* package, and Section 2.18 describes how to apply general hidden Markov models.

1.6.4 Example of a general hidden Markov model

Jackson and Sharples [3] 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 [31] 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 [3] 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.

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

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

Integrating over y_{ij}^{hid} gives an explicit distribution for y_{ij} conditionally on each underlying state (given in Section 2.18, Table 1). Similar distributions were originally applied by Satten and Longini [8] 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 2.18 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. If the model includes variables with missing values, then the corresponding observations are omitted by `msm` with a warning. If you have missing data, as in any statistical model, it is recommended to ensure these do not result in biases.

An example data set, taken from monitoring a set of heart transplant recipients, is provided with *msm*. (Note: since *msm* version 1.3, the command `data(cav)` is no longer needed to load the data

— it is now "lazy-loaded" when required). Sharples *et al.*[32] 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.

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), `cumrej` (cumulative number of rejection episodes), and `firstobs`, an indicator which is 1 when the observation corresponds to the patient's transplant (the first observation), and 0 when the observation corresponds to a later angiogram.

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

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

```
7          0         4
8          1         1
9          0         1
10         0         3
11         0         4
12         1         1
13         0         1
14         0         1
15         0         1
16         0         1
17         0         1
18         0         2
19         0         3
20         0         3
21         0         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=cav)

    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, we can still have $q_{13} = 0$. The underlying model specifies that the patient must have passed through state 2 in between, rather than jumping straight from 1 to 3. If your data represent the exact and complete transition times of the process, then you must specify `exacttimes=TRUE` or `obstype=2` 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,

```
> 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}.

Such default initial values can be used by supplying `gen.inits=TRUE` in the call to `msm` below, along with a `qmatrix` whose non-zero entries represent the allowed transitions of the model. Alternatively the function `crudeinits.msm` could be used to get this matrix of initial values explicitly as follows. These methods are only available for non-hidden Markov models.

```
> Q.crude  <- crudeinits.msm(state ~ years, PTNUM, data=cav,
+                                    qmatrix=Q)
```

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 initial value matrix `Q` shown above, which we now use to fit the model.

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 cav, a matrix Q indicating the allowed transitions, and initial values. We are ready to run msm.

Model 1: simple bidirectional model

```
> cav.msm <- msm( state ~ years, subject=PTNUM, data = cav,
+                       qmatrix = Q, deathexact = 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 deathexact = 4, to indicate to msm that the entry times into state 4 are observed in this manner. If the model had five states, and states 4 and 5 were two competing causes of death with times recorded exactly in this way, then we would specify deathexact = 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, "exact 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. [1]

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 cav.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. [2] The confidence level can be changed using the cl argument to msm.

```
> cav.msm
```

[1] Note that since version 1.3.2, method="BFGS", is the default optimisation algorithm in msm, since it can use analytic derivatives, which are available for most models.

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

```
Call:
msm(formula = state ~ years, subject = PTNUM, data = cav, qmatrix = Q,     death

Maximum likelihood estimates

Transition intensities
                  Baseline
State 1 - State 1 -0.17037 (-0.19027,-0.15255)
State 1 - State 2  0.12787 ( 0.11135, 0.14684)
State 1 - State 4  0.04250 ( 0.03412, 0.05294)
State 2 - State 1  0.22512 ( 0.16755, 0.30247)
State 2 - State 2 -0.60794 (-0.70880,-0.52143)
State 2 - State 3  0.34261 ( 0.27317, 0.42970)
State 2 - State 4  0.04021 ( 0.01129, 0.14324)
State 3 - State 2  0.13062 ( 0.07952, 0.21457)
State 3 - State 3 -0.43710 (-0.55292,-0.34554)
State 3 - State 4  0.30648 ( 0.23822, 0.39429)

-2 * log-likelihood:  3968.798
```

From the estimated intensities, we see patients are three times as likely to develop symptoms than die without symptoms (transitions from state 1). After disease onset (state 2), progression to severe symptoms (state 3) is 50% more likely than recovery. Once in the severe state, death is more likely than recovery, and a mean of 1 / -0.44 = 2.3 years is spent in state 3 before death or recovery.

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. To specify different initial values, use a `covinits` argument, as described in `help(msm)`. Initial values given in the `qmatrix` represent the intensities with covariate values set to their means in the data. In the following model, all transition intensities are modelled in terms of sex.

Model 2: sex as a covariate

```
> cavsex.msm <- msm( state ~ years, subject=PTNUM, data = cav,
+                       qmatrix = Q, deathexact = 4, covariates = ~ sex)
```

Printing the `msm` object now displays the estimated covariate effects and their confidence intervals (note since version 1.3.2 these are *hazard ratios* $\exp(\beta_{rs})$, not *log hazard ratios* β_{rs} as in previous versions).

```
> cavsex.msm

Call:
msm(formula = state ~ years, subject = PTNUM, data = cav, qmatrix = Q,     covar

Maximum likelihood estimates
Baselines are with covariates set to their means

Transition intensities with hazard ratios for each covariate
                 Baseline
State 1 - State 1 -0.16938 (-1.894e-01,-1.515e-01)
State 1 - State 2  0.12745 ( 1.108e-01, 1.466e-01)
State 1 - State 4  0.04193 ( 3.354e-02, 5.241e-02)
State 2 - State 1  0.22645 ( 1.686e-01, 3.042e-01)
State 2 - State 2 -0.58403 (-1.053e+00,-3.238e-01)
State 2 - State 3  0.33693 ( 2.697e-01, 4.209e-01)
State 2 - State 4  0.02065 ( 2.196e-09, 1.941e+05)
State 3 - State 2  0.13050 ( 7.830e-02, 2.175e-01)
State 3 - State 3 -0.44178 (-5.582e-01,-3.497e-01)
State 3 - State 4  0.31128 ( 2.425e-01, 3.996e-01)
                 sex
State 1 - State 1
State 1 - State 2 0.5632779 (3.333e-01,9.518e-01)
State 1 - State 4 1.1289701 (6.262e-01,2.035e+00)
State 2 - State 1 1.2905854 (4.916e-01,3.388e+00)
State 2 - State 2
State 2 - State 3 1.0765518 (5.194e-01,2.231e+00)
State 2 - State 4 0.0003805 (7.241e-65,1.999e+57)
State 3 - State 2 1.0965531 (1.345e-01,8.937e+00)
State 3 - State 3
State 3 - State 4 2.4135380 (1.176e+00,4.952e+00)

-2 * log-likelihood:  3954.777
```

The sizes of the confidence intervals for some of the hazard ratios suggests there is no information in the data about the corresponding covariate effects, leading to a likelihood that is a flat function of these parameters, and this model should be simplified. The first column shown in the output 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(cavsex.msm, covariates=list(sex=0)) # Male
        State 1
State 1 -0.17747 (-0.19940,-0.15795)
State 2  0.21992 ( 0.16100, 0.30040)
State 3 0
State 4 0
        State 2
State 1  0.13612 ( 0.11789, 0.15718)
State 2 -0.60494 (-0.70972,-0.51563)
State 3  0.12913 ( 0.07728, 0.21578)
State 4 0
        State 3
State 1 0
State 2  0.33409 ( 0.26412, 0.42261)
State 3 -0.41050 (-0.52552,-0.32065)
State 4 0
        State 4
State 1  0.04135 ( 0.03245, 0.05268)
State 2  0.05092 ( 0.01808, 0.14343)
State 3  0.28137 ( 0.21509, 0.36808)
State 4 0
> qmatrix.msm(cavsex.msm, covariates=list(sex=1)) # Female
        State 1
State 1 -1.234e-01 (-1.750e-01,-8.693e-02)
State 2  2.838e-01 ( 1.139e-01, 7.075e-01)
State 3 0
State 4 0
        State 2
State 1  7.667e-02 ( 4.630e-02, 1.270e-01)
State 2 -6.435e-01 (-1.115e+00,-3.714e-01)
State 3  1.416e-01 ( 1.852e-02, 1.083e+00)
State 4 0
        State 3
State 1 0
State 2  3.597e-01 ( 1.804e-01, 7.170e-01)
State 3 -8.207e-01 (-1.587e+00,-4.244e-01)
State 4 0
        State 4
State 1  4.668e-02 ( 2.727e-02, 7.989e-02)
State 2  1.938e-05 ( 3.702e-66, 1.014e+56)
State 3  6.791e-01 ( 3.487e-01, 1.323e+00)
State 4 0
```

Since *msm* version 1.2.3, different transition rates may be easily modelled on different covariates by specifying a named list of formulae as the `covariates` argument. Each element of the list has

a name identifying the transition. In the model below, the transition rate from state 1 to state 2 and the rate from state 1 to state 4 are each modelled on sex as a covariate, but no other intensities have covariates on them.

Model 2a: transition-specific covariates

```
> cavsex.msm <- msm( state ~ years, subject=PTNUM, data = cav,
+                    qmatrix = Q, deathexact = 4,
+                    covariates = list("1-2" = ~ sex, "1-4" = ~sex) )
```

We may also want to constrain the effect of a covariate 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

```
> cav3.msm <- msm( state ~ years, subject=PTNUM, data = cav,
+                  qmatrix = Q, deathexact = 4,
+                  covariates = ~ sex,
+                  constraint = list(sex=c(1,2,3,1,2,3,2)) )
```

This constrains the effect of sex 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. Negative entries can be used to indicate that some effects are equal to minus others: `(1,2,3,-1,2,3,2)` sets the fourth effect to be minus the first.

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 6, 7, that is, q_{32}, q_{34}, to their initial values (0.25 and 0.25, respectively).

Model 4: fixed parameters

```
> cav4.msm <- msm( state ~ years, subject=PTNUM, data = cav,
+                  qmatrix = Q, deathexact = 4,
+                  control = list(trace=2, REPORT=1),
+                  fixedpars = c(6, 7) )
```

A `fixedpars` statement can also be used 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 the estimated 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. Bootstrap confidence intervals are also available for `qmatrix.msm` and for most output functions; these are often more accurate, at the cost of computational time. For more about bootstrapping in *msm*, see Section 2.11.

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(cav.msm, t=10)

           State 1    State 2    State 3   State 4
State 1 0.30940656 0.09750021 0.08787255 0.5052207
State 2 0.17165172 0.06552639 0.07794394 0.6848780
State 3 0.05898093 0.02971653 0.04665485 0.8646477
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.

If `ci="norm"` is specified, then a confidence interval is calculated based on drawing a random sample (default size 1000) from the assumed multivariate normal distribution of the maximum likelihood estimates and covariance matrix, and transforming. If `ci="boot"` is specified, then a bootstrap confidence interval for the transition probability matrix is calculated (see Section 2.11) . However, both of these are computationally intensive, particularly the bootstrap method, so no confidence interval is calculated by default.

Mean sojourn times The function `sojourn.msm` extracts the estimated mean sojourn times in each transient state r, for a given set of covariate values. This is calculated as $-1/\hat{q}_{rr}$, where $\hat{q}_{rr}$ is the rth diagonal entry of the estimated transition intensity matrix.

```
> sojourn.msm(cav.msm)

        estimates        SE        L        U
State 1  5.869552 0.3307930 5.255734 6.555057
State 2  1.644897 0.1288274 1.410825 1.917805
State 3  2.287819 0.2743666 1.808595 2.894023
```

Probability that each state is next The function `pnext.msm` extracts the matrix of probabilities $-q_{rs}/q_{rr}$ that the next state after state r is state s, for each r and s. Together with the mean sojourn times, this gives a more intuitive parameterisation of a continuous-time Markov model than the raw transition intensities q_{rs}. Note these are different from the transition probabilities in a given time t returned by `pmatrix.msm`.

```
> pnext.msm(cav.msm)

        State 1                  State 2
State 1 0                        0.75054 (0.6985,0.7989)
State 2 0.37030 (0.2905,0.4552) 0
State 3 0                        0.29884 (0.1922,0.4281)
State 4 0                        0
        State 3                  State 4
State 1 0                        0.24946 (0.2011,0.3015)
State 2 0.56356 (0.4377,0.6514) 0.06614 (0.0173,0.1996)
State 3 0                        0.70116 (0.5719,0.8078)
State 4 0                        0
```

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.

Bootstrap and asymptotic confidence intervals are available, as for `pmatrix.msm`, but are not calculated by default.

```
> totlos.msm(cav.msm)
```

```
 State 1  State 2  State 3  State 4
8.815917 2.229817 1.747804      Inf
```

Expected first passage times The function `efpt.msm` estimates the expected time until the process first enters a given state or set of states, also called the "hitting time". See its help page for further details.

Expected number of visits The function `envisits.msm` estimates the expected number of visits to a state, computed in a similar way to the total length of stay. See its help page for further details.

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(cav.msm, ind1=c(2,1), ind2=c(1,2))

estimate       se        L        U
1.760527 0.245741 1.339148 2.314497
```

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.

```
> hazard.msm(cavsex.msm)

$sex
                            HR            L            U
State 1 - State 2 0.5632779042 3.333382e-01 9.518320e-01
State 1 - State 4 1.1289701413 6.261976e-01 2.035418e+00
State 2 - State 1 1.2905853501 4.916004e-01 3.388139e+00
State 2 - State 3 1.0765518296 5.193868e-01 2.231408e+00
State 2 - State 4 0.0003804824 7.241465e-65 1.999137e+57
State 3 - State 2 1.0965531163 1.345395e-01 8.937364e+00
State 3 - State 4 2.4135379727 1.176293e+00 4.952139e+00
```

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

```
> qmatrix.msm(cav.msm)
```

then the intensity matrix is evaluated as $Q(\bar{x})$ with all covariates set to their mean values $\bar{x}$ in the data. For factor / categorical variables, the mean of the 0/1 dummy variable for each factor level is used, representing an average over all values in the data, rather than a specific factor level.

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(cavsex.msm, covariates = 0)
```

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

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

Values of categorical covariates must be quoted. For example, consider a covariate `smoke`, representing tobacco smoking status, with three levels, `NON, CURRENT, EX`, representing a non-smoker, current smoker or ex-smoker.

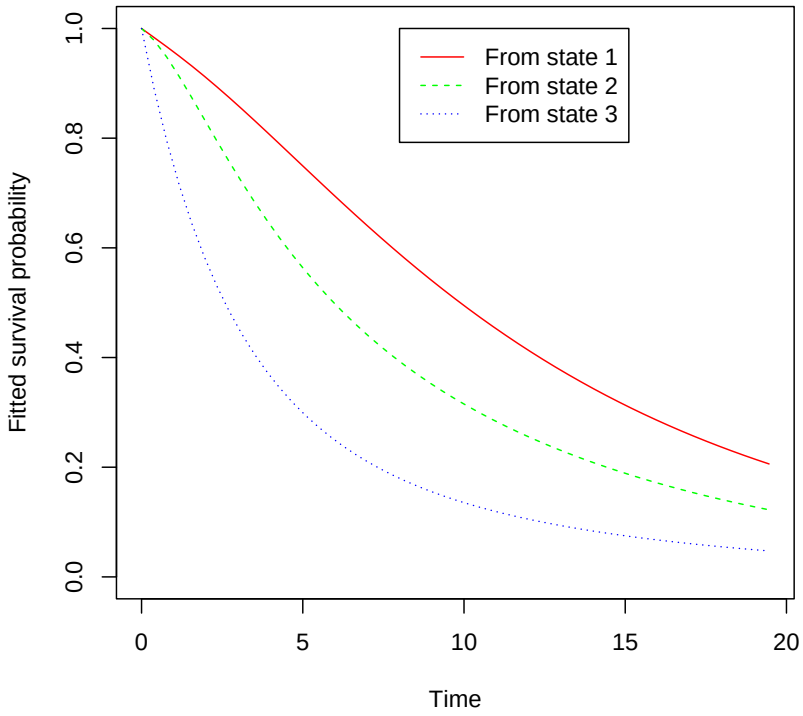
```
qmatrix.msm(example.msm, covariates = list(age = 60, smoke=''CURRENT''))
```

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(cav.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 Bootstrapping

Most of *msm*'s output functions present confidence intervals based on asymptotic standard errors calculated from the Hessian, or transformations of these using the delta method. The asymptotic standard errors are expected to be underestimates of the true standard errors (Cramer-Rao lower bound). For some output functions, such as `pmatrix.msm`, and functions based on `pmatrix.msm` such as `totlos.msm` and `prevalence.msm`, the delta method cannot be used at all to obtain standard errors. In these cases, confidence intervals can be calculated by drawing a random sample from the assumed multivariate normal distribution of the maximum likelihood estimates and covariance matrix, and transforming. However, this is still based on potentially inaccurate asymptotic theory. The *msm* package provides the function `boot.msm` to enable bootstrap refitting of `msm` models, an

alternative way to estimate uncertainty.

For non-hidden Markov models, a bootstrap dataset is drawn by resampling pairs of consecutive states from the full data, i.e. *transitions*. These are assumed to be independent when calculating the likelihood (Section 1.4). For hidden Markov models and models with censoring, a bootstrap dataset is drawn by resampling complete series from independent subjects. The bootstrap datasets have the same number of transitions, or subjects, respectively, as the original data.

For most output extractor functions provided with *msm*, the option `ci="boot"` is available, as a wrapper around `boot.msm`, to enable bootstrap confidence intervals to be calculated. But any user-defined output statistic can be bootstrapped, as follows. The function `boot.msm` is called with the fitted `msm` model as first argument, and an R function specifying the statistic to be bootstrapped as the second argument `stat`. The return value from `boot.msm` is a list of `B` replicates (by default, `B=1000`) of the desired statistic. For example, to bootstrap the transition intensity matrix of the heart transplantation model `cav.msm`:

```
q.list <- boot.msm(cav.msm, stat=function(x){qmatrix.msm(x)$estimates})
```

Note that for `boot.msm` to be able to refit the original model that produced `cav.msm`, all objects used in the original model fit (for example, in this case, `Q`) must be in the working environment. Otherwise, `boot.msm` will give an "object not found" error.

The user can then summarise these replicates by calculating empirical standard deviations or quantile-based intervals. In this example, `q.list` is a list of 1000 4×4 matrices. The following code calculates the bootstrap standard error as the empirical standard deviation of the 1000 replicates, and a similar 95% bootstrap confidence interval.

```
q.array <- array(unlist(q.list), dim=c(4,4,1000))
apply(q.array, c(1,2), sd)
apply(q.array, c(1,2), function(x)quantile(x, c(0.025, 0.975)))
```

Note that when bootstrapping, the refits of the model to the resampled datasets may occasionally fail to converge (as discussed in Section 2.12) even if the original model fit did converge. In these cases, a warning is given, but `boot.msm` simply discards the failed dataset and moves on to the next bootstrap iteration. Unless convergence failure occurs for a large proportion of iterations, this should not affect the accuracy of the final bootstrap confidence intervals.

2.12 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 may be a saddle point rather than the maximum likelihood, or it may be close to the maximum.

Model simplification Firstly, make sure there are not too many parameters in the model. There may not be enough information in the data on a certain transition rate. It is 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`), and do this for any subgroups defining covariates. 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.

Be careful to use a observation scheme and transition matrix appropriate to your data (see Section 1.3). By default, `msm` assumes that the data represent snapshots of the process, and the true state is unknown between observation times. In such circumstances, it is rarely feasible to estimate an intensity matrix with *instantaneous* transitions allowed between every pair of states. This would be easier if the complete course of the process is known `(exacttimes = TRUE)` in the call to `msm`. Understand the difference between *instantaneous* and *interval* transitions - although individuals may be in state 1 at time t_r, and state 3 at time t_{r+1}, that doesn't mean that instantaneous transitions from 1 to 3 should be permitted.

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. It is also sensible to run the model for several different initial values to ensure that the estimation has converged to a global rather than a local optimum.

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: `control = 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", in which `optim()` reports convergence of the optimisation but the Hessian is not positive definite, can sometimes be solved by tightening the criteria (`reltol`, defaults to `1e-08`) for reporting convergence. For example, `control = 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, `control = list(ndeps = rep(1e-6, 5))`

Choice of algorithm By default, since version 1.3.2, `msm` uses the BFGS method of `optim`, which makes use of analytic derivatives. Analytic derivatives are available for all models in msm, apart from hidden Markov models with unknown initial state probabilities, misclassification models with equality constraints on misclassification probabilities, and truncated or measurement-error outcome distributions. This speeds up optimisation. Though alternative algorithms are available such as `method = "CG"`. Or use the `nlm` R function via `msm(..., opt.method`

= "nlm" , ...) Note also the Fisher scoring method available for non-hidden Markov models for panel data, via msm(..., opt.method = "fisher", ...), is expected to be faster than the generic methods, but less robust to bad initial values. Or since version 1.3.2, msm can also use method="bobyqa" from the *minqa* package, a fast derivative-free method.

optim "function cannot be evaluated at initial parameters" To diagnose this problem, run msm again with fixedpars=TRUE set, to calculate the -2 log-likelihood at the initial values. This will probably be Inf. To show the contributions of individual subjects to the overall log likelihood, call logLik.msm(x, by.subject=TRUE), where x is the fitted model object. If only a few subjects give an infinite log-likelihood, then you can check whether their state histories are particularly unusual and conflict with the model. For example, they might appear to make unusually large jumps between states in short periods of time. For models with misclassification, note that the default true initial state distribution initprobs puts all individuals in true state 1 at their first observation. If someone starts in a much higher state, this may result in an infinite log-likelihood, and changing initprobs would be sensible.

2.13 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 multistate 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 $n(t)P(t)_{0,r}$. If the covariates on which $P(t)$ depends vary between individuals, then this can be averaged over the covariates observed in the data.

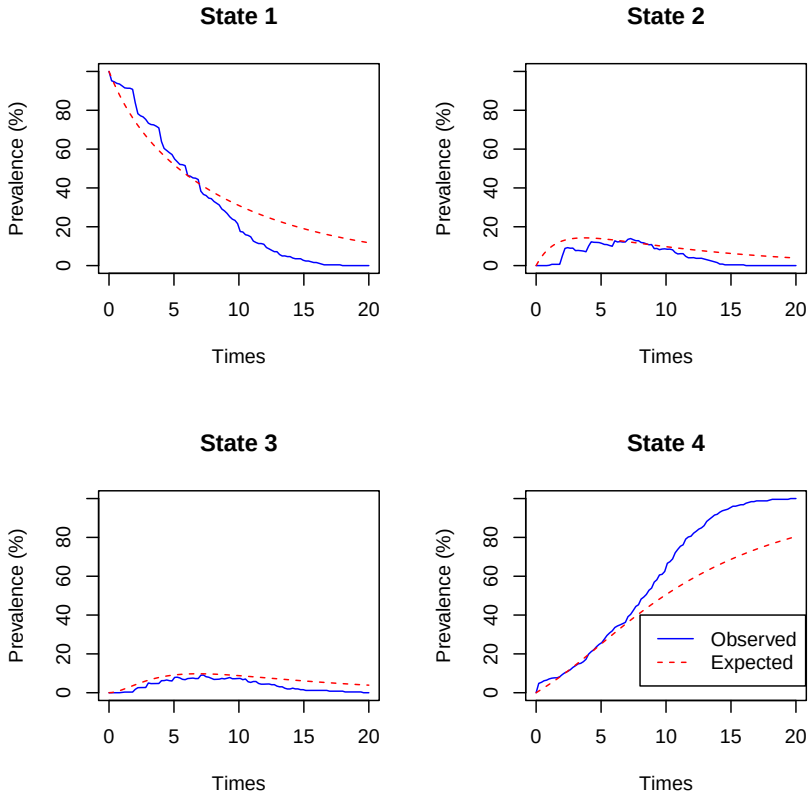
For example, we calculate the observed and expected numbers and percentages at two-yearly intervals up to 20 years after transplant, for the heart transplant model cav.msm. The number of individuals still alive and under observation decreases from 622 to 251 at year 20. The observed and expected percentages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

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

$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      21     190   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   74.50   23.84    52.7   588
4    279.8   68.57   38.83    93.8   481
6    184.2   51.97   38.06   120.8   395
8    129.8   39.31   32.91   141.0   343
10    91.6   28.86   26.01   149.5   296
12    68.6   22.13   20.81   158.5   270
14    54.0   17.66   17.02   169.3   258
16    43.5   14.35   14.04   180.1   252
18    35.7   11.86   11.71   191.7   251
20    29.5    9.83    9.76   201.9   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   7.095   64.19
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    4.05    8.96
4     58.2   14.26    8.07   19.51
6     46.6   13.16    9.64   30.58
8     37.8   11.46    9.59   41.10
10    30.9    9.75    8.79   50.52
12    25.4    8.20    7.71   58.69
```

```
14     20.9     6.84     6.60    65.64
16     17.3     5.69     5.57    71.48
18     14.2     4.73     4.67    76.37
20     11.8     3.92     3.89    80.43
```

```
> plot.prevalence.msm(cav.msm, mintime=0, maxtime=20)
```



Comparing the observed and expected percentages in each state, we see that the predicted number of individuals who die (State 4) is under-estimated by the model from about year 8 onwards. Similarly the number of individuals sill alive and free of CAV (State 1) is over-estimated by the model from about year 8 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. The `pci` argument to `msm` can be used to automatically construct models with transition intensities which are piecewise-constant in time.

In this example, the hazard of death may increase with age, so that 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. Semi-Markov models may in principle be fitted to this type of data using phase-type distributions. Since version 1.4.1 the `phase.states` option to `msm` can be used to define some phase-type models. See `help(msm)` for further details.

However, if it is known that individuals who died would not have been followed up after a certain time, had they survived to that time, then they should not be included in the observed prevalence of the death state after that time. This can be accounted for by passing a vector of maximum potential follow-up times, one for each individual in the same order as the original data, in the `censtime` argument to `prevalence.msm`. Ignoring the potential follow-up times is likely to have resulted in overestimates of the number of deaths at later times for the "observed" prevalences in the CAV example, though these times are not available in the data supplied with *msm*.

Pearson-type goodness-of-fit test Suppose that the true transition times are unknown, and data consist of observations of the process at arbitrary times which differ between individuals (panel data). Assessing goodness of fit by prevalence counts then involves estimating the observed prevalence at a series of points by some form of interpolation. This is only advisable if observation times are close together. An alternative method of assessing goodness-of-fit is to construct tables of observed and expected numbers of transitions, as described by Aguirre-Hernandez and Farewell [33]. This leads to a formal test of goodness-of-fit, analogous to the classical Pearson χ^2 test for contingency tables. The tables are constructed as follows. Each pair of successive observations in the data (*transition*) is classified by

- the starting state r and finishing state s,
- time between the start of the process and the first of the pair of observations (indexed by h),
- time interval between the observations (indexed by l_h, within categories h),
- (if there are fitted covariates) the impact of covariates, as summarised by q_{rr} (indexed by c),
- any other grouping of interest for diagnosing lack of fit (indexed by g).

Groupings of continuous quantities are normally defined by quantiles, so that there are a similar number of observed transitions in each (one-dimensional) category. The observed and expected numbers of transitions in each group are then defined by

$$o_{hl_hrscg} = \sum I(S(t_{i,j+1}) = s, S(t_{ij}) = r)$$

$$e_{hl_hrscg} = \sum P(S(t_{i,j+1}) = s | S(t_{ij}) = r)$$

where $I(A)$ is the indicator function for an event A and the summation is over the set of transitions in the category defined by h, l_h, c, g, over all individuals i. The Pearson-type test statistic is then

$$T = \sum_{hl_hrscg} \frac{(o_{hl_hrscg} - e_{hl_hrscg})^2}{e_{hl_hrscg}}$$

The classical Pearson test statistic is distributed as χ^2_{n-p}, where n is the number of independent cells in the table and p is the number of estimated parameters p. But the null distribution of T is not exactly χ^2, since the time intervals are non-identical, therefore the observed transitions are realizations from a set of independent but non-identical multinomial distributions. Titman [34] showed that the null distribution of T is asymptotically approximated by a weighted sum of χ^2_1 random variables. Aguirre-Hernandez and Farewell [33] also showed that χ^2_{n-p} is a good approximation if there are no fitted covariates. For models with covariates, the null mean of T is higher than $n-p$, but lower than n. Therefore, upper and lower bounds for the true p-value of the statistic can be obtained from the χ^2_{n-p} and χ^2_n distributions. Aguirre-Hernandez and Farewell [33] also described a bootstrap procedure for obtaining an accurate p-value.

Titman and Sharples [35] described modifications to the test to correct for the biases introduced where in addition to the panel-data observation scheme:

- Times of death are known exactly. In this case, transitions ending in death are classified according to the next scheduled observation time after the death, which is estimated by multiple imputation from a Kaplan-Meier estimate of the distribution of time intervals between observations.
- An individual's final observation is censored, so that they are only known to be alive at that point.
- States are misclassified.

The *msm* package provides the function `pearson.msm` to perform the Pearson-type test. By default, three groups are used for each of h, l_h and c. Often the number of groups will need to be reduced in cases where the resulting contingency tables are sparse (thus there are several low expected counts and the variance of T is inflated).

The test is now performed on the model `cav.msm` for the heart transplant dataset (a version of which was also analysed by Titman and Sharples [35]). The default three interval groups are used, and two groups of the time since the start of the process. The `transitions` argument groups the transitions from state 3 to each of states 1, 2 and 3 (the 9th, 10th and 11th transitions) together in the table, since these transitions are infrequent.

```
> options(digits=2)
> pearson.msm(cav.msm, timegroups=2,
+             transitions=c(1,2,3,4,5,6,7,8,9,9,9,10))

Imputing sampling times after deaths...
Calculating replicates of test statistics for imputations...
$Observed
         Time Interval 1-1 1-2 1-3  1-4 2-1 2-2 2-3  2-4
1     [0,4.1)        1 220  27  11 20.1  11  32  14  2.1
2 [4.1,19.46]        1 177  29   3  4.7   9  39  13  3.3
3     [0,4.1)        2 291  33   8 24.8   1   1   0  2.1
4 [4.1,19.46]        2 198  25   4  5.6  17  27  16  3.8
5     [0,4.1)        3 288  42  10 35.1   1   4   0  2.7
6 [4.1,19.46]        3 193  48   8 57.7   7  31  11 33.9
  3-1,3-2,3-3  3-4
```

```
1          24  2.0
2          45  5.4
3           0  2.5
4          34  6.2
5           0  2.4
6          21 36.4

$Expected
         Time Interval 1-1 1-2  1-3  1-4   2-1  2-2  2-3
1     [0,4.1)        1 232  26  5.5 14.4  9.10 33.7 12.0
2 [4.1,19.46]        1 184  18  2.9  8.6  9.53 37.8 12.7
3     [0,4.1)        2 266  45 14.3 31.7  0.92  1.5  1.1
4 [4.1,19.46]        2 198  21  3.6 10.2 10.09 35.7 13.3
5     [0,4.1)        3 272  48 16.8 38.0  1.74  2.6  2.0
6 [4.1,19.46]        3 205  37 15.0 50.1 16.90 24.1 17.5
   2-4 3-1,3-2,3-3  3-4
1  4.3          19  6.8
2  4.2          39 11.6
3  0.7           0  1.1
4  4.7          30 10.3
5  1.4           0  1.1
6 24.3          24 33.0

$`Deviance*sign(O-E)`
         Time Interval    1-1    1-2     1-3   1-4    2-1
1     [0,4.1)        1 -0.644  0.017  5.5468  3.31  0.402
2 [4.1,19.46]        1 -0.291  6.595  0.0068 -2.13 -0.034
3     [0,4.1)        2  2.428 -3.229 -2.7762 -1.90  0.114
4 [4.1,19.46]        2  0.012  0.810  0.0508 -2.39  4.756
5     [0,4.1)        3  1.013 -0.876 -2.7639 -0.65 -0.332
6 [4.1,19.46]        3 -0.670  3.255 -3.2810  1.32 -5.803
     2-2    2-3   2-4 3-1,3-2,3-3   3-4
1 -0.100  0.342 -1.37        1.24 -3.50
2  0.057  0.013 -0.65        1.10 -3.68
3 -0.215 -1.054  3.53        0.00  2.00
4 -2.136  0.557 -0.66        0.67 -1.93
5  0.917 -1.961  1.77        0.00  1.68
6  1.991 -2.447  3.93       -0.55  0.42

$test
 stat df.lower p.lower df.upper p.upper
   98       NA      NA       42 2.4e-06
```

The first two tables in the output show the contingency tables of observed and expected numbers of transitions. Note that the observed number of transitions in certain categories is not a whole number, since these are averaged over multiple imputations of the next scheduled observation time

following deaths. The column `Time` is the group defined by time since the start of the process, and the column `Interval` is the group defined by intervals between observations. The columns indicate the allowed transitions, or pairs of states which can be observed in successive observations.

The third table presents the "deviance", the value of $\frac{(o_{hl_h rscg} - e_{hl_h rscg})^2}{e_{hl_h rscg}}$ for each cell, multipled by the sign of $o_{hl_h rscg} - e_{hl_h rscg}$ to indicate whether there were more or fewer transitions than expected in each cell. These can indicate areas of bad fit. For example, systematic changes in deviance by time or time interval between observations can indicate that a model with time-varying transition intensities is more suitable. Changes in deviance by covariate impact may indicate heterogeneity between individuals which is unexplained by the fitted covariates. Changes in deviance with the length of the interval between observations may also indicate failure of the Markov assumption, and that a semi-Markov model (in which intensities depend on the time spent in the current state) may fit better.

In this example, the test statistic is 100. `p.upper` is an upper bound for the p-value of the statistic based on an asymptotic χ^2_{42} distribution, therefore the model does not fit well. It is not clear from the table of deviances which aspects of the fit are most influental to the test statistic. However, the two-way Markov model itself is not biologically plausible, as discussed in Section 2.14.

For non-hidden Markov models for panel data, `pearson.msm` also presents the accurate analytic p-value of Titman [34]. For all models, `pearson.msm` provides an option for parametric bootstrapping to obtain an accurate p-value.

2.14 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 a 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.

The true state for every patient at the date of transplant is known to be "CAV-free", not misclassified. To indicate this we use the argument `obstrue` to `msm`. This is set to be a variable in the dataset, `firstobs`, indicating where the observed state equals the true state. This takes the value of 1 at the patient's first observation, at the transplant date, and 0 elsewhere.

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 of the probabilities of occupying each true state at the initial observation (equation 13). This can also be a matrix with number of rows equal to the number of subjects, if these probabilities are subject-dependent and known. If not given, all individuals are assumed to be in true state 1 at their initial observation. If `est.initprobs=TRUE` is specified, then these probabilites are estimated as part of the model fit, using a vector `initprobs` as initial values. Covariate effects on these probabilities can also be estimated using a multinomial logistic regression model, if an `initcovariates` argument is specified. See `help(msm)` for further details.

Model 5: multi-state model with misclassification

```
> Qm <- 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))
> cavmisc.msm <- msm(state ~ years, subject = PTNUM, data = cav,
+                    qmatrix = Qm, ematrix = ematrix, deathexact = 4,
+                    obstrue = firstobs)
> cavmisc.msm

Call:
msm(formula = state ~ years, subject = PTNUM, data = cav, qmatrix = Qm,     emat

Maximum likelihood estimates

Transition intensities
                  Baseline
State 1 - State 1 -0.13099 (-0.147787,-0.11610)
State 1 - State 2  0.08963 ( 0.075760, 0.10604)
State 1 - State 4  0.04136 ( 0.033239, 0.05146)
```

```
State 2 - State 2 -0.29196 (-0.357597,-0.23837)
State 2 - State 3  0.25864 ( 0.191525, 0.34928)
State 2 - State 4  0.03331 ( 0.006919, 0.16041)
State 3 - State 3 -0.30758 (-0.389687,-0.24278)
State 3 - State 4  0.30758 ( 0.242779, 0.38969)

Misclassification probabilities
                       Baseline
Obs State 1 | State 1 0.97310 (0.95463,0.98417)
Obs State 2 | State 1 0.02690 (0.01583,0.04537)
Obs State 1 | State 2 0.17491 (0.10066,0.28648)
Obs State 2 | State 2 0.76191 (0.61063,0.86720)
Obs State 3 | State 2 0.06318 (0.03646,0.10730)
Obs State 2 | State 3 0.11510 (0.05732,0.21768)
Obs State 3 | State 3 0.88490 (0.78232,0.94268)

-2 * log-likelihood:  3934
```

Thus there is an estimated probability of about 0.03 that mild/moderate CAV will be diagnosed erroneously, but a rather higher probability of 0.17 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 0.06, 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 8 years is spent disease-free, an average of about 3 years is spent with mild/moderate disease, and periods of severe disease also last about 3 years on average before death.

2.15 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. The linear effects on the log odds of each misclassified state relative to the true state are initialised to zero by default (but this can be changed with the `misccovinits` argument).

Model 6: misclassification model with misclassification probabilities modelled on sex

```
> cavmiscsex.msm <- msm(state ~ years, subject = PTNUM, data = cav,
+                       qmatrix = Qm, ematrix = ematrix,
+                       deathexact = 4, misccovariates = ~sex,
+                       obstrue=firstobs)

> cavmiscsex.msm

Call:
msm(formula = state ~ years, subject = PTNUM, data = cav, qmatrix = Qm,     emat

Maximum likelihood estimates
```

```
Baselines are with covariates set to their means

Transition intensities
                  Baseline
State 1 - State 1 -0.13070 (-0.147075,-0.11614)
State 1 - State 2  0.08936 ( 0.075791, 0.10536)
State 1 - State 4  0.04133 ( 0.033186, 0.05148)
State 2 - State 2 -0.30047 (-0.366866,-0.24609)
State 2 - State 3  0.26674 ( 0.198271, 0.35885)
State 2 - State 4  0.03373 ( 0.006891, 0.16514)
State 3 - State 3 -0.30261 (-0.383636,-0.23870)
State 3 - State 4  0.30261 ( 0.238699, 0.38364)

Misclassification probabilities with odds ratios for each covariate
                       Baseline
Obs State 1 | State 1 0.98708 (7.302e-03,1.0000)
Obs State 2 | State 1 0.01292 (1.260e-06,0.9927)
Obs State 1 | State 2 0.17018 (9.625e-02,0.2831)
Obs State 2 | State 2 0.76366 (6.046e-01,0.8723)
Obs State 3 | State 2 0.06616 (3.801e-02,0.1127)
Obs State 2 | State 3 0.13400 (7.063e-02,0.2396)
Obs State 3 | State 3 0.86600 (7.604e-01,0.9294)
                       sex
Obs State 1 | State 1
Obs State 2 | State 1 0.0002031 (2.064e-39,1.998e+31)
Obs State 1 | State 2 4.6907958 (1.027e+00,2.143e+01)
Obs State 2 | State 2
Obs State 3 | State 2 0.7165202 (8.176e-02,6.279e+00)
Obs State 2 | State 3 6.7620730 (1.198e+00,3.818e+01)
Obs State 3 | State 3

-2 * log-likelihood:  3924
```

The large confidence interval for the odds ratio for 1/2 misclassification suggests there is no information in the data about the difference between genders in the false positive rates for angiography. On the other hand, women have slightly more false negatives.

2.16 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(cavmiscsex.msm, covariates=list(sex=0))
```

```
        State 1
State 1 0.96647 (0.979394,0.99192)
State 2 0.14620 (0.095468,0.28494)
State 3 0
State 4 0
        State 2
State 1 0.03353 (0.008077,0.02061)
State 2 0.78329 (0.601923,0.87350)
State 3 0.11055 (0.067261,0.24927)
State 4 0
        State 3                     State 4
State 1 0                           0
State 2 0.07050 (0.037291,0.11471) 0
State 3 0.88945 (0.750735,0.93274) 0
State 4 0                           1.00000
```

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

```
        State 1
State 1 1.000e+00 (7.773e-34,1.0000)
State 2 4.513e-01 (5.507e-04,0.9871)
State 3 0
State 4 0
        State 2
State 1 7.047e-06 (1.332e-37,1.0000)
State 2 5.155e-01 (1.600e-01,0.9821)
State 3 4.566e-01 (3.028e-02,0.4340)
State 4 0
        State 3                       State 4
State 1 0                             0
State 2 3.324e-02 (9.898e-03,0.3342) 0
State 3 5.434e-01 (5.660e-01,0.9697) 0
State 4 0                             1.000e+00
```

The confidence intervals for the estimates for women are wider, since there are only 87 women in this set of 622 patients.

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

```
odds.msm(cavmiscsex.msm)
```

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 2.13, 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.

Goodness-of-fit test The Pearson-type goodness-of-fit test is performed, as in Section 2.13. The table of deviances indicates that there are more 1-3 and 1-4 transitions than expected in short intervals, and fewer in long intervals. This may indicate some time-dependence in the transition rates. Indeed, Titman [36] found that a model with piecewise-constant transition intensities gave a greatly improved fit to these data.

```
> pearson.msm(cavmisc.msm, timegroups=2,
+             transitions=c(1,2,3,4,5,6,7,8,9,9,9,10))

Imputing sampling times after deaths...
Calculating replicates of test statistics for imputations...
$Observed
         Time Interval 1-1 1-2 1-3  1-4 2-1 2-2 2-3  2-4
1     [0,4.1)        1 220  27  11 19.8  11  32  14  2.0
2 [4.1,19.46]        1 177  29   3  4.6   9  39  13  3.0
3     [0,4.1)        2 291  33   8 25.2   1   1   0  2.5
4 [4.1,19.46]        2 198  25   4  5.7  17  27  16  4.0
5     [0,4.1)        3 288  42  10 35.0   1   4   0  2.5
6 [4.1,19.46]        3 193  48   8 57.7   7  31  11 34.0
  3-1,3-2,3-3  3-4
1          24  2.1
2          45  5.5
3           0  2.4
4          34  6.1
5           0  2.5
6          21 36.4

$Expected
         Time Interval 1-1 1-2  1-3  1-4   2-1  2-2  2-3
1     [0,4.1)        1 232  27  5.7 13.8 13.73 29.3 12.0
2 [4.1,19.46]        1 178  22  4.7  8.4 12.86 32.2 14.4
3     [0,4.1)        2 273  42 12.4 30.0  0.93  1.8  1.1
4 [4.1,19.46]        2 194  24  5.1 10.0 13.16 32.0 14.2
5     [0,4.1)        3 280  45 14.4 35.7  1.55  2.8  1.9
6 [4.1,19.46]        3 198  41 18.0 50.6 10.70 27.0 19.5
```

```
    2-4 3-1,3-2,3-3   3-4
1  3.97          20  6.16
2  4.49          39 11.15
3  0.69           0  0.93
4  4.72          30 10.33
5  1.22           0  1.03
6 25.72          23 34.17

$`Deviance*sign(O-E)`
         Time Interval    1-1    1-2   1-3  1-4    2-1
1     [0,4.1)        1 -0.647  0.014  4.94  3.6 -0.547
2 [4.1,19.46]        1 -0.016  1.940 -0.64 -2.1 -1.162
3     [0,4.1)        2  1.217 -1.820 -1.59 -1.3  0.097
4 [4.1,19.46]        2  0.103  0.063 -0.22 -2.2  1.131
5     [0,4.1)        3  0.266 -0.192 -1.33 -0.4 -0.208
6 [4.1,19.46]        3 -0.121  1.390 -5.55  1.1 -1.284
    2-2   2-3   2-4 3-1,3-2,3-3  3-4
1  0.26  0.35 -1.31        0.88 -2.9
2  1.44 -0.15 -0.82        0.89 -3.1
3 -0.40 -1.13  5.47        0.00  2.3
4 -0.78  0.25 -0.54        0.69 -2.0
5  0.62 -1.87  1.76        0.00  2.0
6  0.62 -3.75  2.82       -0.28  0.2

$test
 stat df.lower p.lower df.upper p.upper
   77       NA      NA       42 0.00085
```

2.17 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 [37], it is also described by Durbin *et al.*[23] and Macdonald and Zucchini [28] 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 $v_k(t_i)$ 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 t_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(cavmisc.msm)
> vit[vit$subject==100103,]
```

```
    subject time observed fitted pstate.1 pstate.2 pstate.3
567  100103  0.0        1      1  1.00000  0.00000  0.00000
568  100103  2.0        1      1  0.87731  0.12269  0.00000
569  100103  4.1        2      2  0.01091  0.89490  0.09418
570  100103  5.0        3      3  0.00000  0.37517  0.62483
571  100103  5.9        2      3  0.00000  0.33416  0.66584
572  100103  7.0        3      3  0.00000  0.02262  0.97738
573  100103  8.0        3      3  0.00000  0.00092  0.99908
574  100103  8.4        4      4  0.00000  0.00000  0.00000
    pstate.4
567  0.00000
568  0.00000
569  0.00000
570  0.00000
571  0.00000
572  0.00000
573  0.00000
574  1.00000
```

2.18 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.

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

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)),
```

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)	$\Gamma(x+n)/(\Gamma(n)x!)p^n(1-p)^x$
`hmmBetaBinom`	Beta-binomial	`size`, `meanp`, `sdp`	n, μ, σ	μ (logit)	$\binom{n}{x} Beta(x+a, n-x+b)/Beta(a,b)$ where $a = \mu/\sigma, b = (1-\mu)/\sigma$
`hmmBeta`	Beta	`shape1`, `shape2`	a, b		$\Gamma(a+b)/(\Gamma(a)\Gamma(b))x^{a-1}(1-x)^{b-1}$
`hmmT`	Student t	`mean`, `scale`, `df`	μ, σ, k	μ (identity)	$\frac{\Gamma((k+1)/2)}{\Gamma(k/2)} \sqrt{\frac{1}{k\pi\sigma^2}} \left\{1 + \frac{1}{k\sigma^2}(x-\mu)^2\right\}^{-(k+1)/2}$
`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*.

```
  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))
```

Parameters of the outcome distributions may also be constrained within specific ranges. If chosen carefully, this may improve identifiability of hidden Markov states. For example to constrain the mean for state 1 to be between 80 and 110, and the mean for state 2 to be between 50 and 80, specify

```
hranges = list(mean=list(lower=c(80,50), upper=c(110,80)))
```

Maximum likelihood estimation is then performed on the appropriate log or logit-transformed scale so that these constraints are satisfied. See the `msm` help page for further details. Note that initial values should be strictly within the ranges, and not on the range boundary.

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`. 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,
+                 deathexact=3, hmodel=hmodel1,
+                 hcovariates=list(~acute, ~acute, NULL),
+                 hcovinits = list(-8, -8, NULL),
+                 hconstraint = list(acute = c(1,1)))
> fev1.msm

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

Maximum likelihood estimates
Baselines are with covariates set to their means

Transition intensities
                  Baseline
State 1 - State 1 -7.037e-04 (-8.332e-04,-0.0005944)
State 1 - State 2  6.274e-04 ( 5.200e-04, 0.0007570)
State 1 - State 3  7.633e-05 ( 3.969e-05, 0.0001468)
State 2 - State 2 -8.008e-04 (-1.012e-03,-0.0006335)
State 2 - State 3  8.008e-04 ( 6.335e-04, 0.0010124)

Hidden Markov model, 3 states
State 1 - normal distribution
Parameters:
      Estimate LCL  UCL
mean      97.9  97 98.6
sd        16.2  16 16.6
acute     -8.8 -10 -7.6

State 2 - normal distribution
Parameters:
      Estimate LCL  UCL
mean      51.8  51 52.8
sd        17.7  17 18.3
acute     -8.8 -10 -7.6

State 3 - identity distribution
Parameters:
      Estimate LCL UCL
which      999  NA  NA


-2 * log-likelihood:  51598

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

```
        estimates  SE    L    U
State 1      1421 122 1200 1682
State 2      1249 149  988 1579
```

Printing the *msm* object `fev1.msm` shows estimates and confidence intervals for the transition intensities 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 90 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.

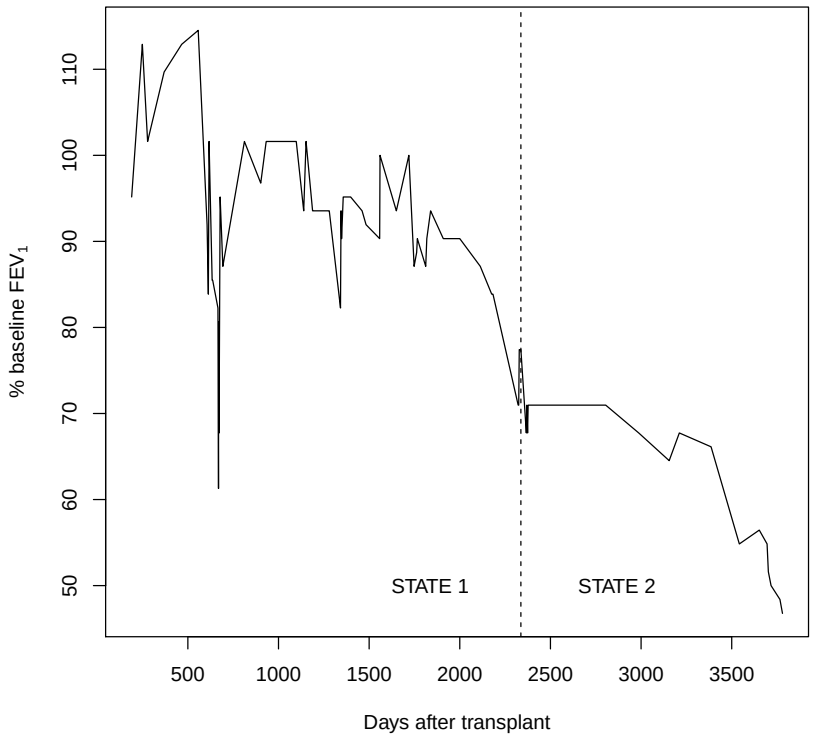
The convergence of maximum likelihood estimation in this example is particularly sensitive to the optimisation method and options, initial values, the unit of the time variable and whether covariates are centered, probably because the likelihood surface is irregular near to the true maximum.

```
hmodel2 <- list(hmmMETNorm(mean=90, 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,
                deathexact=3, hmodel=hmodel2,
                hcovariates=list(~acute, ~acute, NULL),
                hcovinits = list(-8, -8, NULL),
                hconstraint = list(sderr = c(1,1), acute = c(1,1)),
                control=list(maxit=10000), center=TRUE)
```

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 code will create a plot that illustrates a trajectory of declining FEV_1 from the first lung transplant recipient in this dataset. 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 a 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]))
(min2 <- min(vit$time[vit$fitted==2]))
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 2.14 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) if the `obstrue=firstobs` is removed from the `msm()` call. `obstrue` has a slightly different meaning for models specified with `hmodel`. If supplied, the variable indicated by `obstrue` should contain the true state if this is known, and `NA` otherwise, whereas the `state` variable contains an observation generated from the HMM outcome model given the (unobserved) true state. For models specified with `ematrix`, the `state` variable contains the (observed) true state itself. Thus the `hmodel` specification is not strictly suitable for the CAV data, since the true state is both *known* and *observed* at the time of transplant.

```
Qm <- 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))
cavmisc.msm <- msm(state ~ years, subject = PTNUM, data = cav,
                   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 = Qm, deathexact = 4)
cavmisc.msm
```

2.18.1 Hidden Markov models with multivariate outcomes

Since version 1.5.2, *msm* can fit models where at each time point, there are multiple outcomes generated conditionally on a single hidden Markov state. The outcomes must be independent conditionally on the hidden state, but they may be generated from the same or different univariate distributions.

See `help(hmmMV)` for detailed documentation and a worked example.

2.18.2 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-0.7.2.tar.gz` for version 0.7.2, 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`. This ensures that the optimisation to estimate the parameter takes place on a suitable scale, for example, a log scale for a parameter constrained to be positive. If the parameter should be fixed during maximum likelihood estimation (for example, the denominator of a binomial distribution) 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 linear 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. (Optionally) Write a C function to compute the derivatives of the probability density with respect to the parameters, and put this in `src/hmmderiv.c`, with a declaration in `src/hmm.h`. Then add the model to `DHMODELS` in `lik.c` (in the same position as in `HMODELS`) and `.msm.HMODELS.DERIV` in `R/constants.R`. This will generally double the speed of maximum likelihood estimation, but analytic derivatives will not be available for all distributions.

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

10. 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 (`chris.jackson@mrc-bsu.cam.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

For a detailed list of the changes in recent versions of *msm*, see the `NEWS` file in the top-level directory of the installed package.

Development versions of *msm* are to be found on GitHub `https://github.com/chjackson/msm`. These are often more recent than the version released on CRAN, so if you think you have found a bug, then please check first to see whether it has been fixed on the GitHub version.

B Get in touch

If you use *msm* in your work, whatever your field of application, please let me know, for my own interest! Suggestions for improvement are welcome.

References

[1] D. R. Cox and H. D. Miller. *The Theory of Stochastic Processes*. Chapman and Hall, London, 1965.

[2] C. H. Jackson, L. D. Sharples, S. G. Thompson, S. W. Duffy, and E. Couto. Multistate Markov models for disease progression with classification error. *Journal of the Royal Statistical Society, Series D - The Statistician*, 52(2):193–209, July 2003.

[3] C. H. Jackson and L. D. Sharples. Hidden Markov models for the onset and progression of bronchiolitis obliterans syndrome in lung transplant recipients. *Statistics in Medicine*, 21:113–128, 2002.

[4] L. D. Sharples. Use of the Gibbs sampler to estimate transition rates between grades of coronary disease following cardiac transplantation. *Statistics in Medicine*, 12:1155–1169, 1993.

[5] J. H. Klotz and L. D. Sharples. Estimation for a Markov heart transplant model. *The Statistician*, 43(3):431–436, 1994.

[6] R. Kay. A Markov model for analysing cancer markers and disease states in survival studies. *Biometrics*, 42:855–865, 1986.

[7] I. M. Longini, W. S. Clark, R. H. Byers, J. W. Ward, W. W. Darrow, G. F. Lemp, and H. W. Hethcote. Statistical analysis of the stages of HIV infection using a Markov model. *Statistics in Medicine*, 8:851–843, 1989.

[8] G. A. Satten and I. M. Longini. Markov chains with measurement error: Estimating the 'true' course of a marker of the progression of human immunodeficiency virus disease. *Applied Statistics - Journal of the Royal Statistical Society Series C*, 45(3):275–295, 1996.

[9] C. Guihenneuc-Jouyaux, S. Richardson, and I. M. Longini. Modelling markers of disease progression by a hidden Markov process: Application to characterising CD4 cell decline. *Biometrics*, 56:733–741, 2000.

[10] R. C. Gentleman, J. F. Lawless, J. C. Lindsey, and P. Yan. Multi-state Markov models for analysing incomplete disease history data with illustrations for HIV disease. *Statistics in Medicine*, 13(3):805–821, 1994.

[11] G. Marshall and R. H. Jones. Multi-state Markov models and diabetic retinopathy. *Statistics in Medicine*, 14, 1995.

[12] P. K. Andersen. Multistate models in survival analysis: a study of nephropathy and mortality in diabetes. *Statistics in Medicine*, 7(6):661–670, 1988.

[13] S. W. Duffy and H. H. Chen. Estimation of mean sojourn time in breast cancer screening using a Markov chain model of entry to and exit from preclinical detectable phase. *Statistics in Medicine*, 14:1531–1543, 1995.

[14] H. H. Chen, S. W. Duffy, and L. Tabar. A Markov chain method to estimate the tumour progression rate from preclinical to clinical phase, sensitivity and positive predictive value for mammography in breast cancer screening. *The Statistician*, 45(3):307–317, 1996.

[15] A. J. Kirby and D. J. Spiegelhalter. Statistical modelling for the precursors of cervical cancer. In *Case Studies in Biometry*. Wiley, New York, 1994.

[16] P. K. Andersen, L. S. Hansen, and N. Keiding. Assessing the influence of reversible disease indicators on survival. *Statistics in Medicine*, 10:1061–1067, 1991.

[17] J. Grüger, R. Kay, and M. Schumacher. The validity of inferences based on incomplete observations in disease state models. *Biometrics*, 47:595–605, 1991.

[18] M. J. Sweeting, V. Farewell, and D. De Angelis. Multi-State Markov Models for Disease Progression in the Presence of Informative Examination Times: an Application to Hepatitis C. *Statistics in Medicine*, 29(11):1161–1174, 2010.

[19] J.D. Kalbfleisch and J.F. Lawless. The analysis of panel data under a Markov assumption. *Journal of the American Statistical Association*, 80(392):863–871, 1985.

[20] C. Moler and C. van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[21] H. Putter, M. Fiocco, and R. B. Geskus. Tutorial in biostatistics: competing risks and multi-state models. *Statistics in Medicine*, 26:2389–2430, 2007.

[22] B. H. Juang and L. R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33:251–272, 1991.

[23] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[24] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[25] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970.

[26] P. S. Albert. A mover-stayer model for longitudinal marker data. *Biometrics*, 55(4):1252–1257, 1999.

[27] A. Bureau, J. P. Hughes, and S. C. Shiboski. An S-Plus implementation of hidden Markov models in continuous time. *Journal of Computational and Graphical Statistics*, 9:621–632, 2000.

[28] I. L. Macdonald and W. Zucchini. *Hidden Markov and Other Models for Discrete-Valued Time Series*. Chapman and Hall, London, 1997.

[29] J. K. Lindsey. *Models for Repeated Measurements*. Oxford Statistical Science Series. Oxford University Press, second edition, 1999.

[30] P. Guttorp. *Stochastic Modeling of Scientific Data*. Chapman and Hall, London, 1995.

[31] C. H. Jackson. *Statistical models for the latent progression of chronic diseases using serial biomarkers*. PhD thesis, University of Cambridge, 2002.

[32] L.D. Sharples, C.H. Jackson, J. Parameshwar, J. Wallwork, and S.R. Large. Diagnostic accuracy of coronary angiography and risk factors for post-heart-transplant cardiac allograft vasculopathy. *Transplantation*, 76(4):679–682, 2003.

[33] R. Aguirre-Hernandez and V. Farewell. A Pearson-type goodness-of-fit test for stationary and time-continuous Markov regression models. *Statistics in Medicine*, 21:1899–1911, 2002.

[34] A.C. Titman. Computation of the asymptotic null distribution of goodness-of-fit tests for multistate models. *Lifetime Data Analysis*, 15(4):519–533, 2009.

[35] A. Titman and L. D. Sharples. A general goodness-of-fit test for Markov and hidden Markov models. *Statistics in Medicine*, 27(12):2177–95, 2008.

[36] A. Titman. *Model diagnostics in multi-state models of biological systems*. PhD thesis, University of Cambridge, 2008.

[37] J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.