A Users Guide to **panel**
Technical Report 12
Department of Statistics
University of Auckland

R. C. Gentleman
Department of Statistics
University of Auckland

August 7, 2001

Abstract

Documentation for the R/S/Splus function **panel**. Included are installation instructions, a brief description of the problem and some problem solving hints. It is assumed that you know how to program R/S/Splus and some experience with the foreign function interface would be helpful. Panel uses some FORTRAN code and it calls EISPACK so you will need a FORTRAN compiler and some means of loading the compiled code into R/S/Splus. In addition it is helpful if you know about maximum likelihood estimation and stochastic processes. References for these last two are Cox and Hinkley (1974) and Cox and Miller (1965).

1 Introduction

The function **panel** fits a multi–state Markov model to panel data. Such data arise in many areas such as sociology, psychology and medical research. The basic situation is that individuals are observed at some number of points in time such as yearly or monthly intervals. At those times the subjects are interviewed (or examined) and their current *state* is determined. This may be voting preference or it may be their current disease status. In some cases, particularly disease history, a number of covariates may also be measured at each inspection time. Since individuals are not under continuous surveillance not all transitions are recorded. The data really consist only of the inspection times and the state of the individual at the inspection times. There are a great number of examples of such data, see for example Kalbfleisch and Lawless (1985), Gentleman, Lawless, Lindsey and Yan (1994), Gentleman (1994) and Bartholomew (1985).

The software was written to deal with the case where there are n individuals with the i^{th} individual inspected at times $t_{i,0}, \ldots, t_{i,m_i}$. In situations where all individuals are inspected at the same time points some computational savings would be possible but these have not been incorporated. At each inspection time the individual is determined to be in one of k states, some of which may be absorbing. The methodology is based on modeling the transition intensities, i.e. the instantaneous rate of change from one state to another. For a model with k states the transition intensities can be described in terms of a k by k matrix called the transition intensity matrix which will be denoted $\mathbf{Q}$. Covariate information is included by modeling these transition intensities as functions of the covariate (Kalbfleisch and Lawless, 1985). There are many similarities between multi–state models and the proportional hazards model which can be exploited; especially when considering time dependent covariates (Gentleman, 1994).

Section 2 presents the methodology used and defines the notation needed. In Section 3 the algorithms, both computational and modeling, and data structures used in the function and associated software are detailed. Also in this section several code examples are provided. All the examples should be loaded into R/S/SPlus when you install `panel`. Section 4 discusses some of the issues regarding the use of covariates. Finally Section 5 discusses the implementation, installation and gives some hints on problem solving.

2 Methodology

2.1 Markov Processes

Let $Y(t)$ be a stochastic process which indicates the state of an individual at time t. We do not observe $Y(t)$ continuously but rather for each individual we observe $Y(t)$ at a finite set of points $\{t_{ij}\}_{i=1}^{m_i}$. An important assumption is that the inspection times are probabilistically independent of the underlying process $Y(t)$. By that I mean that there is no information about the change of state in the inspection time sequence. In the health care setting this assumption would be violated if the inspection times coincided with hospital visits and the visits were caused by a change in disease status.

The basis for inference in this setting is the probability that an individual in state l at time t_1 is later in state k at time t_2. In general this probability depends on the entire past history of the individual but for a special class of models, the Markov models, it is assumed to depend only on the previously observed state. While this assumption simplifies the situation greatly you should be aware that it is a very strong assumption that is seldom met.

A multi-state model $\{Y(t): \quad t \geq 0\}$ is Markov[1] if

$$\begin{aligned} Pr\{Y(t_2) = v | Y(t_1) = u,\ Y(t),\ \ 0 \leq t < t_1\} &= Pr\{Y(t_2) = v | Y(t_1) = u\} \\ &= p_{uv}(t_1, t_2) \end{aligned} \tag{1}$$

for all $u,\ v \in \{1, 2, \dots, k\}$ and $t_1 < t_2$. The transition intensity functions for the process are

$$q_{uv}(t) = \lim_{\Delta t \downarrow 0} \frac{Pr\{Y(t + \Delta t) = v | Y(t) = u\}}{\Delta t}, \quad u \neq v, \tag{2}$$

and for convenience we define $q_{uu}(t) = -\sum_{v \neq u} q_{uv}(t)$ with the $k \times k$ transition intensity matrix $\mathbf{Q}(t) = (q_{uv}(t))$. When none of the transition intensities $q_{uv}(t)$ depend on time, the model is called time homogeneous.

We assume henceforth that $\mathbf{Q}$ is specified in terms of a $b \times 1$ vector $\underline{\theta}$ of unknown parameters. For example, the transition intensity matrix, $\mathbf{Q}$, for the model in Figure 1 is

$$\begin{pmatrix} -\theta_1 & \theta_1 & 0 & 0 \\ \theta_2 & -\theta_2 - \theta_3 - \theta_6 & \theta_3 & \theta_6 \\ 0 & \theta_4 & -\theta_4 - \theta_5 & \theta_5 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3}$$

In this case the transition probabilities (1) are stationary, and we write $p_{uv}(t_1, t_2) = p_{uv}(0, t_2 - t_1) = p_{uv}(t_2 - t_1)$ with $\mathbf{P}(t) = (p_{uv}(t))$. $\mathbf{P}(t)$ can be estimated using the methods described in Kalbfleisch and Lawless (1985). I have chosen to use the parameterization $\underline{\gamma} = \log(\underline{\theta})$ for reasons given later.

Consider a time homogeneous Markov model with $k \times k$ transition intensity matrix $\mathbf{Q}$ and transition probability matrix $\mathbf{P}(t)$. Assume that $\mathbf{Q}$ is specified in terms of a $b \times 1$ vector $\underline{\theta}$ of unknown parameters. It is well known that (Cox and Miller, 1965):

$$\mathbf{P}(t) = \exp\{\mathbf{Q}t\} = \sum_{r=0}^{\infty} \frac{\mathbf{Q}^r t^r}{r!}. \tag{4}$$

When $\mathbf{Q}$ is diagonalizable, i.e. $\mathbf{Q}$ has k linearly independent eigenvectors, an efficient means of computing the transition probabilities, given the matrix of transition intensities, is as follows. Let $d_1, d_2, \dots, d_k$ be the eigenvalues of $\mathbf{Q}$ and let $\mathbf{A}$ denote the $k \times k$ matrix of rank k whose v^{th} column is the right eigenvector corresponding to d_v, then

$$\mathbf{Q} = \mathbf{A}\mathbf{D}\mathbf{A}^{-1},$$

with $\mathbf{D} = diag(d_1, d_2, \ldots, d_k)$. Thus,

$$\mathbf{P}(t) = \mathbf{A} diag(e^{d_1 t}, e^{d_2 t}, \ldots, e^{d_k t}) \mathbf{A}^{-1}, \tag{5}$$

where the dependence of $\mathbf{Q}$, $\mathbf{P}(t)$, $\mathbf{A}$ and the d_v's on $\underline{\theta}$ is suppressed for notational convenience. Once $\mathbf{A}$ and $\mathbf{D}$ are obtained, transition probabilities may be rapidly computed from (5). Additional comments on computation are given in Kalbfleisch and Lawless (1985, 1989) and Gentleman *et al* (1994).

2.2 Covariates

To include covariate information we are going to make the very simple assumption that $\mathbf{P}$ depends not only on t but also on the value of a covariate, **z**, which may be vector valued. There are many ways in which the covariate can affect the probability transition matrix. We will only consider discrete covariates. The assumption that I have used is that there is a different **P** matrix for each different combination of the covariates. These **P** matrices can be independent or linked, in terms of their parameterization. When specifying the model in R/S/SPlus you must provide a function that evaluates to give the **Q** matrices and one that gives the derivatives of the **Q** matrices with respect to the parameters. It is essential that these be correct. A separate decomposition is computed for each value of the covariate and for each transition the correct decomposition is used for the current value of the covariate.

Consider the simple case where the covariate takes only two values and assume a general model in which $\mathbf{P}(t; z = 0)$ and $\mathbf{P}(t; z = 1)$ are completely unrelated. To fit this we use two $\mathbf{Q}$ matrices, one for the $z = 0$ transitions and the other for the $z = 1$ transitions.

$$\mathbf{Q}(z=0) = \begin{pmatrix} -\theta_1 & \theta_1 & 0 & 0 \\ \theta_2 & -\theta_2 - \theta_3 - \theta_6 & \theta_3 & \theta_6 \\ 0 & \theta_4 & -\theta_4 - \theta_5 & \theta_5 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{6}$$

$$\mathbf{Q}(z=1) = \begin{pmatrix} -\theta_7 & \theta_7 & 0 & 0 \\ \theta_8 & -\theta_8 - \theta_9 - \theta_{12} & \theta_9 & \theta_{12} \\ 0 & \theta_{10} & -\theta_{10} - \theta_{11} & \theta_{11} \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{7}$$

Notice that the parameters in $\mathbf{Q}(z = 1)$ are completely independent of those in $\mathbf{Q}(z = 0)$. Time dependent covariates are easily handled. When an individual has covariate $z = 0$ the first $\mathbf{Q}$ matrix controls the transitions and when $z = 1$ it is the second $\mathbf{Q}$ matrix. Various sub–models can easily be examined and tested via the likelihood ratio test. For example, replacing θ_7 by θ_1 in $\mathbf{Q}(z = 1)$ would allow us to test whether the transition intensities from state one to state two differ for different values of z.

In the interest of simplicity I have assumed that there is only one discrete covariate. If you have two or more you will need to arrange them so that there is only one. For example if variable one comes at 2 levels and variable two comes at 3 levels then you should make up a composite variable that has six level (one for each possible combination of the two variables).

An ordinal variable that comes at 3 levels could be handled (in a regression setting) by modeling the **Q** matrix by

$$\mathbf{Q}(z) = \mathbf{Q_o} exp(\beta z)$$

with $z = 0, 1, 2$ depending on the level of the covariate and $\mathbf{Q_o}$ being a standard **Q** matrix. Provided that you provide correct functions for computing **Q** and its derivatives with respect to all the parameters (including β) you can fit this model. See the third example in Section 3.

2.3 The Likelihood

The likelihood function for $\underline{\theta} = (\theta_1, \ldots, \theta_b)'$, assuming a time homogeneous Markov process $Y_i(t)$ observed intermittently at times

$$t_{i,0} < t_{i,1} < \cdots < t_{i,m_i-1} < t_{i,m_i}, \quad i = 1, ..., n \text{ individuals}$$

with states

$$(y_0^i, \mathbf{z}_0^i), \quad (y_1^i, \mathbf{z}_1^i), \quad \ldots, \quad (y_{m_i}^i, \mathbf{z}_{m_i}^i)$$

conditional upon the state occupied at $t_{i,0}$, is

$$L(\underline{\theta}) = \prod_{i=1}^{n} \prod_{r=1}^{m_i} p_{y_{r-1}^i, y_r^i}(t_{i,r} - t_{i,r-1};\ \underline{\theta}, \mathbf{z}_r^i).$$

Notice that while $\mathbf{z}$ may depend on t our assumption of time homogeneity of the probability transition matrix is not violated. We make the assumption that if at inspection time t the individual has covariate value $\mathbf{z}_0$ then the transition intensity matrix $\mathbf{Q}(\mathbf{z}_0)$ applies until the next inspection time at which point the covariate may have changed. Unlike the proportional hazards model assumptions do not have to be made regarding the covariate values of other individuals at inspection times in order to be able to find the maximum likelihood estimates.

3 Data Structures and Algorithms

3.1 Data Structures

The data are stored as a list. The list has one node for each individual and each node has 4 components. Within a node the components are **time**, a vector of inspection times in whatever units you are using, **stage**, a vector of stages occupied by the individual at the corresponding inspection times, **cov**, a vector of covariate value(s) at the corresponding inspection times, and **len**, the length of the time vector.

The covariate is used to index the array **Q(z)** so it is important that it be of type integer and it should take values from 1 to the number of levels (`ncov`). If the covariate is of type factor you should replace it with the levels of the factor rather than use the factor itself. In particular, $\mathbf{Q}[1,,]$ is the transition intensity matrix for the covariate level 1. To fit a model without covariates simply make up a covariate vector for each individual that is always 1 and set `ncov` to 1.

To elucidate several of the points it is perhaps best to have a concrete example in mind. Therefore, consider the following simple example from Gentleman, Lawless, Lindsey and Yan (1994). The model has four disease states which are defined in terms of the individual's CD4 level at the time of inspection. Figure 1 portrays such a model used with HIV disease. The arrows indicate the possible instantaneous transitions between the states. The assumption that instantaneous transitions are only permitted between adjacent states reflects the belief that CD4 behaves approximately like a continuous function of time.

An example of the first element from the data structure for this model is given below.

```
> t4iga.data[[1]]
$time:
 [1]  586  680  771  866  968 1066 1166 1248 1397 1692 2038

$stage:
 [1] 3 3 3 3 3 2 3 3 3 3 3

$cov:
 [1] 2 2 2 2 2 3 2 1 2 2 2

$len:
[1] 11
```

This indicates, among other things, that at time 586 (days) the individual concerned was in stage 3 and had a covariate value of 2. There were 11 observations on this individual at times ranging from 586 days to 2038 days. The individual was in state 3 at most inspection times appearing in state 2 on the sixth inspection only. As indicated previously this does not imply that the individual visited state 2 only once but rather that it was observed in state 2 only once. They may have visited state 2 many times but never been observed there; in fact they could visit every non–absorbing state many times between inspections.

Next we need to examine the `panel` function itself.

```
function(indata, qmatf, gamma, qderivf, npar, nstage, ncov, verbose = F,
        tol = 0.001)
```

The arguments are as follows:

indata The data in the list structure described above.

qmatf A function which takes the parameter vector $\underline{\gamma}$ as an argument and returns the **Q** matrix. This is a three–way array. The first dimension codes the levels of the covariate and within each of these the two–way array is the **Q** matrix.

gamma The vector of parameter estimates. Recall that $\theta_i = exp(\gamma_i)$.

qderivf This is a function that takes the parameter vector $\underline{\gamma}$ as an argument and returns a four–way matrix. The matrix contains the derivatives of the **Q** matrix with respect to each of the γ_i. Since the **Q** matrix is three–way we need an extra dimension to handle this.

npar The number of parameters, ie. the length of $\underline{\gamma}$.

nstage The number of states or stages.

ncov The number of levels of the covariate. Currently you can only have one covariate.

verbose If true the function prints out diagnostic information as it is running. This is essential for debugging.

tol This is the tolerance on convergence. Setting it smaller means that more iterations (typically) will be needed.

3.2 Some Examples

Working versions are stored in the `test` directory.

Let's consider the model described in Figure 1 with a covariate which comes at two levels. We will first let the two **Q** matrices be completely independent. This means that we will have 12 parameters. Here I will give the correct **qmatf**, **qderivf** for this model.

```
> qfun.basic <- function(gamma)
{
        qarr <- array(0, dim = c(2, 4, 4))
        theta <- exp(gamma)
        qarr[1, 1, 1] <-  - theta[1]
        qarr[1, 1, 2] <- theta[1]
        qarr[1, 2, 1] <- theta[2]
        qarr[1, 2, 2] <-  - theta[2] - theta[3] - theta[6]
        qarr[1, 2, 3] <- theta[3]
        qarr[1, 2, 4] <- theta[6]
        qarr[1, 3, 2] <- theta[4]
        qarr[1, 3, 3] <-  - theta[4] - theta[5]
        qarr[1, 3, 4] <- theta[5]
        qarr[2, 1, 1] <-  - theta[7]
        qarr[2, 1, 2] <- theta[7]
        qarr[2, 2, 1] <- theta[8]
        qarr[2, 2, 2] <-  - theta[8] - theta[9] - theta[12]
        qarr[2, 2, 3] <- theta[9]
        qarr[2, 2, 4] <- theta[12]
        qarr[2, 3, 2] <- theta[10]
        qarr[2, 3, 3] <-  - theta[10] - theta[11]
        qarr[2, 3, 4] <- theta[11]
        return(qarr)
}
```

Notice that `theta.in` is exponentiated because we want the **Q** matrix in terms of $\underline{\theta}$ but the optimization in terms of $\underline{\gamma}$. The dimensions of the return matrix are 2 (for the two levels of the covariate) and 4 by 4. Since there are 4 stages the **Q** matrix is 4 by 4. The derivative matrix is more complicated.

```
> qderivf <- function(gamma)
{
        rmat <- array(0, c(12, 2, 4, 4))
        theta <- exp(gamma)
        rmat[1, 1, 1, 1] <- ( - theta[1])
        rmat[1, 1, 1, 2] <- theta[1]
        rmat[7, 2, 1, 1] <- ( - theta[7])
        rmat[7, 2, 1, 2] <- theta[7]
        rmat[2, 1, 2, 1] <- theta[2]
        rmat[2, 1, 2, 2] <- ( - theta[2])
        rmat[8, 2, 2, 1] <- theta[8]
        rmat[8, 2, 2, 2] <- ( - theta[8])
        rmat[3, 1, 2, 2] <- ( - theta[3])
        rmat[3, 1, 2, 3] <- theta[3]
        rmat[9, 2, 2, 2] <- ( - theta[9])
        rmat[9, 2, 2, 3] <- theta[9]
        rmat[4, 1, 3, 2] <- theta[4]
        rmat[4, 1, 3, 3] <- ( - theta[4])
        rmat[10, 2, 3, 2] <- theta[10]
        rmat[10, 2, 3, 3] <- ( - theta[10])
        rmat[5, 1, 3, 3] <- ( - theta[5])
        rmat[5, 1, 3, 4] <- theta[5]
        rmat[11, 2, 3, 3] <- ( - theta[11])
        rmat[11, 2, 3, 4] <- theta[11]
        rmat[6, 1, 2, 4] <- theta[6]
        rmat[6, 1, 2, 2] <- ( - theta[6])
        rmat[12, 2, 2, 4] <- theta[12]
        rmat[12, 2, 2, 2] <- ( - theta[12])
        return(rmat)
}
```

First we make up a big array that is filled with zeros. This is important since most of the entries will be zero. The first dimension indicates the parameter, which has 12 levels in our example, the second indicates the covariate, which has 2 levels in our example, and the last two dimensions indicate the position in the relevant **Q** matrix.

The only non–zero entries in rmat are rmat$[i, j, k, l]$ where $\mathbf{Q}(\mathbf{z} = \mathbf{j})[\mathbf{k}, \mathbf{l}] = \theta_\mathbf{i}$. The only place that θ_8 appears is in $\mathbf{Q}(\mathbf{z} = \mathbf{2})$ in row 2 columns 1 and 2 so in `rmat` the only entries of the submatrix `rmat[8,,,]` that should be non–zero are the `rmat[8,2,2,1]` entry and the `rmat[8,2,2,2]` entry. The first of these has θ_8 as an entry, this is simply $\exp(\gamma_8)$ and the derivative of this with respect to γ_8 is $\exp(\gamma_8) = \theta_8$ and that is what we put there. Since the sign on `rmat[8,2,2,2]` is negative we need a negative θ_8 in this entry.

For a second example, suppose that we assumed that the transitions to higher numbered states were the only transitions affected by the covariate. That means that θ_2 and θ_4 would be the same in the two matrices since they refer to transitions to lower states. Making this change and renumbering the parameters so they go from θ_1 to θ_{10} yields the following:

$$\mathbf{Q}(z=0) = \begin{pmatrix} -\theta_1 & \theta_1 & 0 & 0 \\ \theta_2 & -\theta_2 - \theta_3 - \theta_6 & \theta_3 & \theta_6 \\ 0 & \theta_4 & -\theta_4 - \theta_5 & \theta_5 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{8}$$

$$\mathbf{Q}(z=1) = \begin{pmatrix} -\theta_7 & \theta_7 & 0 & 0 \\ \theta_2 & -\theta_2 - \theta_8 - \theta_{10} & \theta_8 & \theta_{10} \\ 0 & \theta_4 & -\theta_4 - \theta_9 & \theta_9 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{9}$$

The appropriate versions of the `qfun` and `qderiv` functions are given below.

```
> qfun.m2 <- function(gamma)
{
        qarr <- array(0, dim = c(2, 4, 4))
        theta <- exp(gamma)
        qarr[1, 1, 1] <-  - theta[1]
        qarr[1, 1, 2] <- theta[1]
        qarr[1, 2, 1] <- theta[2]
        qarr[1, 2, 2] <-  - theta[2] - theta[3] - theta[4]
        qarr[1, 2, 3] <- theta[3]
        qarr[1, 2, 4] <- theta[4]
        qarr[1, 3, 2] <- theta[5]
        qarr[1, 3, 3] <-  - theta[5] - theta[6]
        qarr[1, 3, 4] <- theta[6]
        qarr[2, 1, 1] <-  - theta[7]
        qarr[2, 1, 2] <- theta[7]
        qarr[2, 2, 1] <- theta[2]
        qarr[2, 2, 2] <-  - theta[2] - theta[8] - theta[10]
        qarr[2, 2, 3] <- theta[8]
        qarr[2, 2, 4] <- theta[10]
        qarr[2, 3, 2] <- theta[4]
        qarr[2, 3, 3] <-  - theta[4] - theta[9]
        qarr[2, 3, 4] <- theta[9]
        return(qarr)
}
```

```
> qderiv.m2 <- function(gamma)
{
        rmat <- array(0, c(10, 2, 4, 4))
        theta <- exp(gamma)
        rmat[1, 1, 1, 1] <- ( - theta[1])
        rmat[1, 1, 1, 2] <- theta[1]
        rmat[2, 1, 2, 1] <- theta[2]
        rmat[2, 1, 2, 2] <- ( - theta[2])
        rmat[2, 2, 2, 1] <- theta[2]
        rmat[2, 2, 2, 2] <- ( - theta[2])
        rmat[3, 1, 2, 2] <- ( - theta[3])
        rmat[3, 1, 2, 3] <- theta[3]
        rmat[4, 1, 3, 2] <- theta[4]
        rmat[4, 1, 3, 3] <- ( - theta[4])
        rmat[4, 2, 3, 2] <- theta[4]
        rmat[4, 2, 3, 3] <- ( - theta[4])
        rmat[5, 1, 3, 3] <- ( - theta[5])
        rmat[5, 1, 3, 4] <- theta[5]
        rmat[6, 1, 2, 4] <- theta[6]
        rmat[6, 1, 2, 2] <- ( - theta[6])
        rmat[7, 2, 1, 1] <- ( - theta[7])
        rmat[7, 2, 1, 2] <- theta[7]
        rmat[8, 2, 2, 2] <- ( - theta[8])
        rmat[8, 2, 2, 3] <- theta[8]
        rmat[9, 2, 3, 3] <- ( - theta[9])
        rmat[9, 2, 3, 4] <- theta[9]
        rmat[10, 2, 2, 4] <- theta[10]
        rmat[10, 2, 2, 2] <- ( - theta[10])
        return(rmat)
}
```

A third example has an ordinal covariate at three levels (coded 0, 1, 2) and we want to fit a regression model to the upward transition out of state 1. This yields a model with 8 parameters, the seven θ_i and the regression parameter, β.

The appropriate versions of the `qfun` and `qderiv` functions are given below.

```
> qfun.ord <- function(gamma)
{
        qarr <- array(0, dim = c(3, 4, 4))
        theta <- exp(gamma)
        qarr[1, 1, 1] <-  - theta[1]
        qarr[1, 1, 2] <- theta[1]
        qarr[1, 2, 1] <- theta[2]
        qarr[1, 2, 2] <-  - theta[2] - theta[3] - theta[4]
        qarr[1, 2, 3] <- theta[3]
        qarr[1, 2, 4] <- theta[4]
        qarr[1, 3, 2] <- theta[5]
        qarr[1, 3, 3] <-  - theta[5] - theta[6]
        qarr[1, 3, 4] <- theta[6]
        qarr[2,  ,  ] <- qarr[1,  ,  ]
        qarr[3,  ,  ] <- qarr[1,  ,  ]
        qarr[2, 1, 2] <- qarr[2, 1, 2] * exp(gamma[7])
        qarr[2, 1, 1] <- qarr[2, 1, 1] * exp(gamma[7])
        qarr[3, 1, 1] <- qarr[3, 1, 1] * exp(2 * gamma[7])
        qarr[3, 1, 2] <- qarr[3, 1, 2] * exp(2 * gamma[7])
        return(qarr)
}

> qderiv.ord <- function(gamma)
{
        rmat <- array(0, c(7, 3, 4, 4))
        theta <- exp(gamma)
        rmat[1, 1, 1, 1] <- ( - theta[1])
        rmat[1, 1, 1, 2] <- theta[1]
        rmat[1, 2, 1, 1] <- ( - theta[1] * exp(gamma[7]))
        rmat[1, 2, 1, 2] <- theta[1] * exp(gamma[7])
        rmat[1, 3, 1, 1] <- ( - theta[1] * exp(2 * gamma[7]))
        rmat[1, 3, 1, 2] <- theta[1] * exp(2 * gamma[7])
        rmat[2,  , 2, 1] <- theta[2]
        rmat[2,  , 2, 2] <- ( - theta[2])
        rmat[3,  , 2, 2] <- ( - theta[3])
        rmat[3,  , 2, 3] <- theta[3]
        rmat[4,  , 2, 4] <- theta[4]
        rmat[4,  , 2, 2] <- ( - theta[4])
        rmat[5,  , 3, 3] <- ( - theta[5])
        rmat[5,  , 3, 2] <- theta[5]
        rmat[6,  , 3, 4] <- theta[6]
        rmat[6,  , 3, 3] <- ( - theta[6])
        rmat[7, 2, 1, 1] <- ( - theta[1] * exp(gamma[7]))
        rmat[7, 2, 1, 2] <- theta[1] * exp(gamma[7])
        rmat[7, 3, 1, 1] <- (-2 * theta[1] * exp(2 * gamma[7]))
        rmat[7, 3, 1, 2] <- 2 * theta[1] * exp(2 * gamma[7])
        return(rmat)
}
```

Some simulated data from this model has been included and is in the file *simord.data*. The data and commands for this example are stored in the `tests` directory of the `panel` package.

```
> sim.fit<-panel(sim.ord,qfun.ord, gamma.ord, qderiv.ord, 7, 4, 3, T)
```

In Section 6 of Kalbfleisch and Lawless (1985) some data simulated from a three-state Markov model is given. The transition intensity matrix they used (modulo a small typographic error) is given by

$$\mathbf{Q} = \begin{pmatrix} q_{11}(z) & \exp(a_{12} + z_1\beta_1 + z_2\beta_2) & \exp(a_{13}) \\ \exp(a_{21} + z_1\beta_3 + z_2\beta_4) & q_{22}(z) & \exp(a_{23}) \\ 0 & 0 & 0 \end{pmatrix}.$$

They started 15 individuals in each of States 1 and 2 for each of four groups with regression parameters (z_1, z_2) = (0,0), (0,1), (1,0) and (1,1). Individuals were followed for 5 transitions at equidistant time intervals. The initial values, parameter estimates etc. from their Table 2 are repeated in Table 1. The data is supplied in the bundle with **panel** as are these functions and you should be able to simply execute the two commands below to fit this model.

```
> library("panel")
> klfitted <-panel(kldata,qfun.kl,theta.kl,qderivs.kl,8,3,4,T)
```

The `qfun` and `qderiv` functions for coding this model are given below.

Table 1: True Values, MLE's and Estimated Standard Errors

Parameter	True Value	MLE	Est'd SE
$a_{12} = \gamma_1$	-2.30	-2.177	0.356
$\beta_1 = \gamma_5$	0.5	0.700	0.406
$\beta_2 = \gamma_6$	-0.50	-0.772	0.406
$a_{13} = \gamma_2$	-2.30	-2.659	0.235
$a_{21} = \gamma_3$	-1.90	-1.389	0.278
$\beta_3 = \gamma_7$	0.50	0.284	0.307
$\beta_4 = \gamma_8$	0.50	0.111	0.304
$a_{23} = \gamma_4$	-2.30	-2.246	0.254

```
qfun.kl<- function(gamma)
{
        rmat <- array(0, c(4, 3, 3))
        theta<- exp(gamma)
        rmat[1, 1, 1] <- ( - theta[1] - theta[2])
        rmat[1, 1, 2] <- theta[1]
        rmat[1, 1, 3] <- theta[2]
        rmat[1, 2, 1] <- theta[3]
        rmat[1, 2, 2] <- ( - theta[3] - theta[4])
        rmat[1, 2, 3] <- theta[4]
        rmat[2, 1, 1] <- ( - theta[1] * theta[6] - theta[2])
        rmat[2, 1, 2] <- theta[1] * theta[6]
        rmat[2, 1, 3] <- theta[2]
        rmat[2, 2, 1] <- theta[3] * theta[8]
        rmat[2, 2, 2] <- ( - theta[3] * theta[8] - theta[4])
        rmat[2, 2, 3] <- theta[4]
        rmat[3, 1, 1] <- ( - theta[1] * theta[5] - theta[2])
        rmat[3, 1, 2] <- theta[1] * theta[5]
        rmat[3, 1, 3] <- theta[2]
        rmat[3, 2, 1] <- theta[3] * theta[7]
        rmat[3, 2, 2] <- ( - theta[3] * theta[7] - theta[4])
        rmat[3, 2, 3] <- theta[4]
        rmat[4, 1, 1] <- ( - theta[1] * theta[5] * theta[6] - theta[2])
        rmat[4, 1, 2] <- theta[1] * theta[5] * theta[6]
        rmat[4, 1, 3] <- theta[2]
        rmat[4, 2, 1] <- theta[3] * theta[7] * theta[8]
        rmat[4, 2, 2] <- ( - theta[3] * theta[7] * theta[8] - theta[4])
        rmat[4, 2, 3] <- theta[4]
        return(rmat)
}

qderivs.kl<- function(gamma)
{
        rmat <- array(0, c(8, 4, 3, 3))
        theta <- exp(gamma)
        rmat[1, 1, 1, 1] <- ( - theta[1])
        rmat[1, 1, 1, 2] <- theta[1]
        rmat[1, 2, 1, 1] <- ( - theta[1] * theta[6])
        rmat[1, 2, 1, 2] <- theta[1] * theta[6]
        rmat[1, 3, 1, 1] <- ( - theta[1] * theta[5])
        rmat[1, 3, 1, 2] <- theta[1] * theta[5]
        rmat[1, 4, 1, 1] <- ( - theta[1] * theta[6] * theta[5])
        rmat[1, 4, 1, 2] <- theta[1] * theta[6] * theta[5]
        rmat[2, 1, 1, 1] <- ( - theta[2])
        rmat[2, 1, 1, 3] <- theta[2]
        rmat[2, 2, 1, 1] <- ( - theta[2])
        rmat[2, 2, 1, 3] <- theta[2]
        rmat[2, 3, 1, 1] <- ( - theta[2])
        rmat[2, 3, 1, 3] <- theta[2]
        rmat[2, 4, 1, 1] <- ( - theta[2])
        rmat[2, 4, 1, 3] <- theta[2]
        rmat[3, 1, 2, 1] <- theta[3]
        rmat[3, 1, 2, 2] <- ( - theta[3])
        rmat[3, 2, 2, 1] <- theta[3] * theta[8]
        rmat[3, 2, 2, 2] <- ( - theta[3] * theta[8])
        rmat[3, 3, 2, 1] <- theta[3] * theta[7]
        rmat[3, 3, 2, 2] <- ( - theta[3] * theta[7])
        rmat[3, 4, 2, 1] <- theta[3] * theta[7] * theta[8]
        rmat[3, 4, 2, 2] <- ( - theta[3] * theta[7] * theta[8])
        rmat[4, 1, 2, 2] <- ( - theta[4])
        rmat[4, 1, 2, 3] <- theta[4]
        rmat[4, 2, 2, 2] <- ( - theta[4])
        rmat[4, 2, 2, 3] <- theta[4]
        rmat[4, 3, 2, 2] <- ( - theta[4])
        rmat[4, 3, 2, 3] <- theta[4]
        rmat[4, 4, 2, 2] <- ( - theta[4])
        rmat[4, 4, 2, 3] <- theta[4]
        rmat[5, 3, 1, 1] <- ( - theta[1] * theta[5])
        rmat[5, 3, 1, 2] <- theta[1] * theta[5]
        rmat[5, 4, 1, 1] <- ( - theta[1] * theta[6] * theta[5])
        rmat[5, 4, 1, 2] <- theta[1] * theta[6] * theta[5]
        rmat[6, 2, 1, 1] <- ( - theta[1] * theta[6])
        rmat[6, 2, 1, 2] <- theta[1] * theta[6]
        rmat[6, 4, 1, 1] <- ( - theta[1] * theta[6] * theta[5])
        rmat[6, 4, 1, 2] <- theta[1] * theta[6] * theta[5]
        rmat[7, 3, 2, 1] <- theta[3] * theta[7]
        rmat[7, 3, 2, 2] <- ( - theta[3] * theta[7])
        rmat[7, 4, 2, 1] <- theta[3] * theta[7] * theta[8]
        rmat[7, 4, 2, 2] <- ( - theta[3] * theta[7] * theta[8])
        rmat[8, 2, 2, 1] <- theta[3] * theta[8]
        rmat[8, 2, 2, 2] <- ( - theta[3] * theta[8])
        rmat[8, 4, 2, 1] <- theta[3] * theta[7] * theta[8]
        rmat[8, 4, 2, 2] <- ( - theta[3] * theta[7] * theta[8])
        return(rmat)
}
```

3.3 Optimization Algorithms

To avoid some problems with the optimization rather than work with the θ_i directly they are transformed to $\theta_i = \exp(\gamma_i)$. The main reason being that θ_i are constrained to be positive (requiring constrained optimization) while the γ_i are not constrained and standard Newton–Raphson can be employed. Contrary to popular belief this does not typically alleviate convergence problems and can actually introduce some. If one of the θ_i is close to $zero$ then the corresponding γ_i is close to $-\infty$ and the information matrix will not be stable. The reader is referred to Gentleman, Lawless, Lindsey and Yan for information regarding the testing of hypotheses of the type $H_o : \theta_i = 0$ since these are non–standard. You are testing whether the parameter lies on the boundary of the parameter space.

The optimization algorithm used is a modified step size Newton–Raphson procedure. At each iteration the increase in the log likelihood predicted by the quadratic approximation is computed and then the actual change in the log likelihood is computed. If these two differ substantially the step size is halved and the comparison performed again. The purpose of such a procedure is to guard against extremely large steps being taken in the wrong direction when the quadratic approximation to the likelihood function is not good. The cost of this approach is that one computes the value of the likelihood several times and for large data sets this computation can be time consuming.

In the examples that I have run the convergence is very good for the first few steps and then slows down (sometimes by an extreme amount) subsequently. Convergence problems have always been found to involve either an extremely small transition intensity or an error in coding on my part. If the transition intensity is extremely small you will probably have to remove it from your model or reparameterize it to stabilize the information matrix.

4 Covariates

The modeling of covariates in a multi–state Markov model is problematic. The form of the mathematical relationship between the covariate and the transition intensities must be determined. Multiplicative models are easiest but may not necessarily be appropriate. In addition decisions regarding the modeling of the covariates will have to be made. Should the covariate affect all transition intensities or only specified subsets of the transition intensities. And, given several covariates which should define the states and which should model the intensities.

The easiest model for the transition intensities is an exponential model. The form, $q_{ij}(z) = \exp(z'\beta_{ij})$, where $q_{ij}(z)$ is the transition intensity from state i to state j for an individual with covariate value z, was proposed by Kalbfleisch and Lawless (1985). This model has several computational advantages over other models. In particular the estimated transition intensities will always be positive while for other models some form of constrained estimation will be required. However, it is not necessarily the case that a log linear model is appropriate in all situations and its use should not be automatic.

Multi–state Markov models allow very general covariate models to be fit to the data. Covariates can affect single transition intensities, all transition intensities, transition intensities that move towards a healthier state, or virtually any other effect desired. The fitting of a general covariate model allows us to examine the nature of the covariate effect. For example, in an HIV study, with CD4 defining the states, the effect of AZT on transition intensities can easily be obtained. A model which allows for one set of transition intensities for those on AZT and another set for those not on AZT will allow us to compare them. One can see not only whether AZT has an effect but, to some extent, the nature of the effect. One can gain evidence on whether, for example, the effect of AZT is to increase the intensities of transitions towards a healthier state and to decrease the intensities of transitions towards AIDS.

The covariates that are available will come in many forms. There are fixed covariates, such as treatment or sex, and there are time dependent covariates such as blood glucose level or CD4 count. The issue of whether a variable should be used to define the stages for the Markov model or should be used as a covariate affecting the transition intensities must be addressed. This is similar to determining whether a covariate should be used to define strata or used as a covariate in a proportional hazards model (Kalbfleisch and Prentice, 1980, p. 87). The answer to this question will depend on the relative performance of the covariate in both roles. One method of assessing this performance is suggested in Gentleman (1994). As is the case in the proportional hazards model, inference will only really be possible for variables used as covariates and not for variables used to define the states. We also note that the inference that is drawn will, in some sense, be conditional on the observed covariate paths. The extent to which these are atypical usually cannot be judged from the available data.

Some problems may be encountered with internal covariates. Kalbfleisch and Prentice (1980) define these to be the output of a stochastic process that is generated by the individual. If the covariate process is affected by treatment then it may be difficult to assess the value of the treatment when the covariate is used to define states. For example, suppose that AZT is a good treatment for HIV infection and that its effect is entirely through CD4. If CD4 is used to define the states the treatment effect may be difficult to detect as it may be absorbed in the state *effect*. This suggests that including covariates which are affected by the treatment is problematic and may disguise the true treatment effect. It may also be the case that individuals under study monitor the values of some variables themselves and attempt to alter the levels of the variable through various self–administered interventions. If these variables are included in a model the conclusions drawn would be suspect.

While there are many similarities between the multi–state Markov models with time dependent covariates and the proportional hazards model with time dependent covariates there are also some large differences and it is worth amplifying these. At each observed failure time covariate information on all individuals still at risk is required in order to be able to fit a proportional hazards model. This ensures that the partial likelihood can be determined at that time point. Lin and Ying (1994) discuss various methods for dealing with incomplete covariate measurements for the proportional hazards model. Under a multi–state Markov model there is no such requirement. Multi–state Markov models assume that the waiting time in a state is exponentially distributed. The only data required to form the likelihood are the inspection times and the states occupied at those times. It is important to keep in mind that the times that individuals entered various states are typically unknown. We only know which states were occupied at the inspection times. It is possible that other non–absorbing states were visited between inspection times.

4.1 Basics of the Estimation Process

For the current estimate of the parameters the **Q** matrices are determined and their decomposition computed. Then we loop over the list calling the FORTRAN subroutine CMPSCORE (contained in dcscore.panel.f) once for each individual and compute their contribution to the score vector and information matrix. Then the log likelihood is computed for the current parameter estimates using the FORTRAN subroutine COMPLIKE. This is also done iteratively over individuals (see the S function *dclike.panel*). Then the suggested improvement in $\underline{\theta}$ is computed. The likelihood is computed at this new point, l1, and the estimated value of the likelihood assuming that the quadratic approximation given by the score and information are correct is computed, l2. If l1 and l2 are similar then the step is taken and we loop back to the start to compute the score and information for these values of $\underline{\theta}$. This process is iterated until convergence is attained. Convergence is measured by the maximum of the score vector (in absolute value) being less than *tol*.

The computation of the score and likelihood is pretty inefficient in some ways since we loop through the list of observations calling a FORTRAN subroutine in each iteration. Given the S/SPlus problem with loops this can be pretty slow, the same problems don't occur in R though. It would be possible to store the data in a ragged array, keep track of the number of observations in some other vector and use that to handle the looping in FORTRAN. This would probably make the whole process much faster but I like the list data structure for other reasons and have adopted it.

4.2 Problems and Solutions

I have had relatively few problems with the software. I hope to expand this section as people use the program and report difficulties to me at `rgentlem@stat.auckland.ac.nz`. Please use the word `panel` in the subject line.

The only convergence problems that I have experienced were caused by one of two things.

Either there was one transition intensity that was several orders of magnitude different from the others or (more commonly) I had not written either the `qfun` or `qderiv` functions correctly. If the first situation eventuates you should think about whether such a transition is important for modeling the situation. You will need to either remove this transition from the model or reparameterize it. When I was originally fitting the AIDS data I started by allowing instantaneous transitions between all states. This caused convergence problems and I gradually restricted allowable transitions to be only between neighboring states.

Sometimes re parameterizing the problem can help when there are convergence problems. Scaling the variables so that they are all of approximately the same order of magnitude should cure some of the problems.

References

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

[2] Gentleman, R. ‘The Use of Covariate Information in Multi–State Markov Models’, Department of Statistics Technical Report, University of Auckland, 1994.

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

[4] Kalbfleisch, J.D. and Lawless, J.F. ‘The analysis of panel data under a Markov assumption’, *Journal of the American Statistical Association*, **80**, 863-871, (1985).

[5] Kalbfleisch, J.D. and Lawless, J.F. ‘Some statistical methods for panel life history data’, *Proceedings of the Statistics Canada Symposium on the Analysis of Data in Time*, 185-192, Ottawa: Statistics Canada, 1989.

[6] Kalbfleisch, J. D. and Prentice, R. L. (1980). *The Statistical Analysis of Failure Time Data*, John Wiley and Sons.

[7] Lin, D. Y. and Ying, Z. ‘Cox Regression with incomplete covariate measurements’, *Journal of the American Statistical Association*, **88**, 1341–1349, (1994).

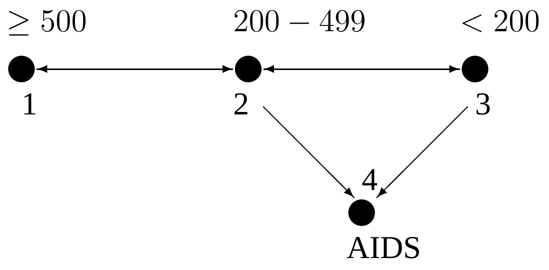

Figure 1: A model with states defined by CD4 counts.