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-281/54c]TJ 0lel

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,

aware that it is a very strong assumption that is seldom met.

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

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

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

where the dependence of $\mathbf{Q}$, $\mathbf{P}(t)$, $\mathbf{A}$ and the d_v's on _ is suppressed for notational convenience. Once $\mathbf{A}$ and $\mathbf{D}$ are obtained, transition probabilities may be rapidly computed from (5). Additional

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.

```
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 _ 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 $_i = exp($

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

```
> qfun.m2 <- function(gamma)
```

```
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])
```

```
        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

given by

$$Q = \begin{pmatrix} 0 & & q \\ B & & \\ @ & & \end{pmatrix}$$

```
        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]
```

3.3 Optimization Algorithms

To avoid some problems with the optimization rather than work with the $_i$ directly they are transformed to $_i = \exp(_i)$. The main reason being that $_i$ are constrained to be positive (requiring constrained optimization) while the $_i$ are not constrained and standard Newton–Raphson c55 Tf92(be)]TJ -

always be positive while for other models some form of constrained estimation will be required.

large differences and it is worth amplifying these. At each observed failure time covariate informa-

Either there was one transition intensity that was several orders of magnitude different from the

y
1