

STAT 778 HW 3

Tom Wallace

April 4, 2018

Program Organization and Compilation

Source code is contained in `hw3.c`. The program can be compiled with the following command:

```
gcc hw3.c -o hw3 -lm
```

Executing the program requires one argument, the name of the input file. An example command is:

```
./hw3 HW2_2018.dat
```

Technical Notes

This program finds maximum partial likelihood estimates (MPLE) for coefficients β in the Cox proportional hazards model. This section provides mathematical background on the computation of these estimates.

The log-likelihood function for the Cox proportional hazards model is:

$$\log L(\beta) = \sum_{i=1}^n \delta_i \left(\beta' \mathbf{x}_i - \log \sum_{l \in R(t_i)} \exp(\beta' \mathbf{x}_l) \right) \quad (1)$$

with:

- $t_1 \dots t_n$ = unique observation times
- δ_i = indicator function, which returns 0 if observation t_i is right-censored
- \mathbf{x}_i = vector of covariates associated with individual observed at t_i
- β = vector of coefficients associated with covariates
- $R(t_i)$ = risk set at t_i

The first order partial derivatives are:

$$\frac{\partial \log L}{\partial \beta_j} = \sum_{i=1}^n \delta_i \left(x_{ij} - \frac{\sum_{l \in R(t_i)} x_{lj} \exp(\beta' \mathbf{x}_l)}{\sum_{l \in R(t_i)} \exp(\beta' \mathbf{x}_l)} \right) \quad (2)$$

It can be shown that these functions are concave, and so the their roots—i.e., the values of β for which the above functions equal $\mathbf{0}$ —are the MPLE estimators $\hat{\beta}$. The Newton-Raphson algorithm is used to numerically estimate the roots (there is no closed-form solution). This algorithm also requires the derivatives of the function to be solved. Thus, we also need the second-order partial derivatives of the log likelihood function.

$$\frac{\partial^2 \log L}{\partial \beta_j \partial \beta_k} = - \sum_{i=1}^n \delta_i \left(\frac{\sum_{l \in R(t_i)} x_{lj} x_{lk} \exp(\beta' \mathbf{x}_l)}{\sum_{l \in R(t_i)} \exp(\beta' \mathbf{x}_l)} - \frac{\left(\sum_{l \in R(t_i)} x_{lj} \exp(\beta' \mathbf{x}_l) \right) \left(\sum_{l \in R(t_i)} x_{lk} \exp(\beta' \mathbf{x}_l) \right)}{\left(\sum_{l \in R(t_i)} \exp(\beta' \mathbf{x}_l) \right)^2} \right) \quad (3)$$

Once we have the Jacobian \mathbf{J} and Hessian \mathbf{H} , the multivariate Newton-Raphson algorithm is:

$$\beta_{n+1} = \beta_n - \mathbf{H}^{-1}(\beta_n) \mathbf{J}(\beta_n) \quad (4)$$

In our case of two covariates:

$$\begin{bmatrix} \beta_{1,n+1} \\ \beta_{2,n+1} \end{bmatrix} = \begin{bmatrix} \beta_{1,n} \\ \beta_{2,n} \end{bmatrix} - \begin{bmatrix} \frac{\partial^2 \log L}{\partial \beta_1^2}(\beta_{1,n}, \beta_{2,n}) & \frac{\partial^2 \log L}{\partial \beta_1 \partial \beta_2}(\beta_{1,n}, \beta_{2,n}) \\ \frac{\partial^2 \log L}{\partial \beta_2 \partial \beta_1}(\beta_{1,n}, \beta_{2,n}) & \frac{\partial^2 \log L}{\partial \beta_2^2}(\beta_{1,n}, \beta_{2,n}) \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial \log L}{\partial \beta_1}(\beta_{1,n}, \beta_{2,n}) \\ \frac{\partial \log L}{\partial \beta_2}(\beta_{1,n}, \beta_{2,n}) \end{bmatrix} \quad (5)$$

Note that \mathbf{H} is square and symmetric. If it is positive definite (and it is), the Cholesky decomposition can be used to efficiently obtain the inverse.

The algorithm iterates until an arbitrary quality of approximation is obtained:

$$\beta_n - \beta_{n-1} \leq \epsilon \quad (6)$$

Verification and Validation

This program's results for $\hat{\beta}$ were compared to those obtained using the `coxph` function from the `survival` package in R. As depicted in Table 1, the two are essentially identical.

Table 1: Verification and validation

	$\hat{\beta}_1$	$SE(\hat{\beta}_1)$	$\hat{\beta}_2$	$SE(\hat{\beta}_2)$
R survival package				
This C program				