STAT 778 HW 3

Tom Wallace

April 10, 2018

Program Organization and Compilation

Source code is contained in `hw3.c`. The program requires the GNU Scientific Library (GSL), an open-source numerical library. It can be obtained from `www.gnu.org/software/gsl`; or, it can installed from most standard Linux package managers. An example command to achieve the latter is:

```
sudo apt-get install gsl-bin libgsl-dev
```

Compilation of `hw3.c` is best achieved in two steps. First, use the below command to compile the program but not link it. You may need to change the argument passed to the `-I` flag to wherever the `gsl` header files live on your computer.

```
gcc -I/usr/include -c hw3.c
```

This command should create an object file `hw3.o`. Link this object file to relevant libraries with the following command. You may need to change the argument passed to the `-L` flag to wherever `libgsl` lives on your computer.

```
gcc -L/usr/lib hw3.o -o hw3 -lgsl -lgslcblas -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 $\boldsymbol{\beta}$ 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:

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

with:

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

The first order partial derivatives are:

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

It can be shown that these functions are concave, and so the their roots—i.e., the values of $\boldsymbol{\beta}$ for which the above functions equal $\mathbf{0}$—are the MPLE estimators $\hat{\boldsymbol{\beta}}$. 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.

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

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

$$\boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_n - \boldsymbol{H}^{-1}(\boldsymbol{\beta}_n)\boldsymbol{J}(\boldsymbol{\beta}_n) \tag{4}$$

In our case of two covariates:

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

The inverse Hessian $\boldsymbol{H}^{-1}$ is obtained using LU decomposition. The algorithm iterates until an arbitrary quality of approximation is obtained: $10e^{-6}$ was used.

$$\boldsymbol{\beta}_n - \boldsymbol{\beta}_{n-1} \leq \epsilon \tag{6}$$

Verification and Validation

This program's output was compared to that obtained using the `coxph` function from the `survival` package in R. As depicted in Table 1, the two produce essentially identical estimated coefficients. I suspect that the difference in log-likelihood estimates is due to a difference in optimization procedures: R often seeks to minimize the negative log-likelihood (c.f. documentation for the `mle` function), whereas my program maximizes the log-likelihood.

Table 1: Verification and validation

	$\hat{\beta}_1$	$\hat{\beta}_2$	Log-likelihood
R `survival` package	0.0405	0.0607	-664.2725
This C program	0.0405	0.0605	-5.2081