Homework#3 Survival Analysis

Name: Huang, Xinwei 黄昕蔚

Question:

Suppose that data $(t_i, \delta_i, x_i), i = 1, \ldots, n$ follow the model $S(t|x_i) = \exp(-\lambda t e^{\beta x_i})$, where $\lambda > 0$ and $-\infty < \beta < \infty$. Let $m = \sum_{i=1}^{n} \delta_i$ be the number of deaths.

(1) Write down the log-likelihood function $\ell(\lambda, \beta) = \log L(\lambda, \beta)$.

(2) Derive the score functions $\partial \ell(\lambda, \beta)/\partial \lambda$ and $\partial \ell(\lambda, \beta)/\partial \beta$.

(3) Derive the fixed-point iteration algorithm and apply it to the data of Example 1.

(4) Derive the Hessian matrix of $\ell(\lambda, \beta)$.

(5) Derive the Newton-Raphson algorithm and apply it to the data of Example 1.

(6) Derive the Newton-Raphson algorithm under the transformed parameter $\tilde{\lambda} = \log(\lambda)$ and apply it to the data of Example 1.

(7) Compare the numbers of iterations in all the three algorithms.

Solution:

(1)

Data: $(t_i, \delta_i, x_i), i = 1, \ldots, n$; $x_i = \begin{cases} 0 & \text{category1} \\ 1 & \text{category2} \end{cases}$; $\delta_i = \begin{cases} 0 & \text{censored} \\ 1 & \text{observed} \end{cases}$;

Model: $S(t|x_i) = \exp(-\lambda t e^{\beta x_i})$, where $\lambda > 0$ and $-\infty < \beta < \infty$

Observed survival time: T

The cumulative density function is $F(t|x_i) = 1 - S(t|x_i) = 1 - \exp(-\lambda t e^{\beta x_i})$, the probability density function is $f(t|x_i) = \frac{\partial F(t|x_i)}{\partial t} = \lambda e^{\beta x_i} \exp(-\lambda t e^{\beta x_i})$, the hazard function is $h(t|x_i) = \frac{f(t|x_i)}{S(t|x_i)} = \frac{\lambda e^{\beta x_i} \exp(-\lambda t e^{\beta x_i})}{\exp(-\lambda t e^{\beta x_i})} = \lambda e^{\beta x_i} = h_0(t) e^{\beta x_i}$, where $h_0(t) = \lambda$ is the baseline hazard function, and the cumulative hazard function is

$$H(t|x_i) = \int_0^t h(u|x_i)du = \int_0^t h_0(t)e^{\beta x_i}du = \lambda t e^{\beta x_i}.$$

For a cox model, the likelihood function of death time t_i could be defined as

$$L(\lambda,\beta) = \prod_{i=1}^{n}\left[h_T(t_i|x_i)^{\delta_i}\exp\left\{-H_T(t_i|x_i)\right\}\right]$$

$$= \prod_{i=1}^{n}\left[\left(\lambda e^{\beta x_i}\right)^{\delta_i}\exp\left\{-\lambda t_i e^{\beta x_i}\right\}\right].$$

Then the log-likelihood function is

$$\ell(\lambda,\beta) = \sum_{i=1}^{n}\left[\delta_i\left\{\log\lambda + \beta x_i\right\} - \lambda t_i e^{\beta x_i}\right]$$

$$= \sum_{i=1}^{n}\delta_i\log\lambda + \sum_{i=1}^{n}\delta_i\beta x_i - \sum_{i=1}^{n}\lambda t_i e^{\beta x_i}$$

$$= m\log\lambda + \sum_{i=1}^{n} I\left(x_i = 1|\delta_i = 1\right)\beta - \lambda\sum_{i=1}^{n} t_i e^{\beta x_i}$$

$$= m\log\lambda + m_1\beta - \lambda\sum_{i=1}^{n} t_i e^{\beta x_i},$$

where $m = \sum_{i=1}^{n}\delta_i$ and $m_1 = \sum_{i=1}^{n} I\left(x_i = 1|\delta_i = 1\right).$

(2)

The score functions are

$$\frac{\partial\ell(\lambda,\beta)}{\partial\lambda} = \frac{m}{\lambda} - \sum_{i=1}^{n} t_i e^{\beta x_i} = \frac{m}{\lambda} - e^{\beta}S_1 - S_0$$

and

$$\frac{\partial\ell(\lambda,\beta)}{\partial\beta} = m_1 - \lambda\sum_{i=1}^{n} t_i x_i e^{\beta x_i} = m_1 - \lambda e^{\beta}S_1$$

where $S_1 = \sum_{x_i=1} t_i$ and $S_0 = \sum_{x_i=0} t_i$.

(3)

To solve the score functions, setting the functions equal to zero

$$\begin{cases} \frac{m}{\lambda} - e^{\beta} S_1 - S_0 = 0 \\ m_1 - \lambda e^{\beta} S_1 = 0 \end{cases},$$

the solution is $\hat{\lambda} = \frac{m - m_1}{S_0} = 0.0008928571$ and $\hat{\beta} = \log\left\{ \frac{m_1}{m - m_1} \frac{S_0}{S_1} \right\} = -0.4054651.$

By the Fixed-point iteration method, we can derive the relationship from the score functions

$$\begin{cases} \lambda = \frac{m}{e^{\beta} S_1 + S_0} \\ \beta = \log\left(\frac{m_1 / \lambda}{S_1} \right) \end{cases}.$$

Fixed-point iteration algorithm:

Step 1: Set the initial point $\beta^{(0)}$;

Step 2: $\lambda^{(k+1)} = \frac{m}{e^{\beta^{(k)}} S_1 + S_0}$, $\beta^{(k+1)} = \log\left(\frac{m_1 / \lambda^{(k+1)}}{S_1} \right)$;

Step 3: If $\left| \beta^{(k+1)} - \beta^{(k)} \right| < 10^{-5}$ and $\left| \lambda^{(k+1)} - \lambda^{(k)} \right| < 10^{-5}$, then stop. $\hat{\beta} = \beta^{(k+1)}$ and $\hat{\lambda} = \lambda^{(k+1)}$; Else go Step 2.

```
> fixed.Point(iteration.Function4, c(0.00001,0), 10^(-5))
Iteration 1 :  0.0007142857 -0.1823216
Iteration 2 :  0.0007936508 -0.2876821
Iteration 3 :  0.0008403361 -0.3448405
Iteration 4 :  0.0008658009 -0.3746934
Iteration 5 :  0.0008791209 -0.3899609
Iteration 6 :  0.0008859358 -0.397683
Iteration 7 :  0.000889383 -0.4015665
Iteration 8 :  0.0008911167 -0.4035139
Iteration 9 :  0.0008919861 -0.404489
Iteration 10 :  0.0008924214 -0.4049769
Iteration 11 :  0.0008926392 -0.405221
Iteration 12 :  0.0008927482 -0.405343
Iteration 13 :  0.0008928027 -0.4054041
Iteration 14 :  0.0008928299 -0.4054346
Iteration 15 :  0.0008928435 -0.4054498
Iteration 16 :  0.0008928503 -0.4054575
```

(4)

The second differentiate of the log-likelihood function are

$$\frac{\partial^2 \ell(\lambda, \beta)}{\partial \lambda^2} = -\frac{m}{\lambda^2}$$

$$\frac{\partial^2 \ell(\lambda, \beta)}{\partial \beta^2} = -\lambda e^{\beta} S_1$$

$$\frac{\partial^2 \ell(\lambda,\beta)}{\partial\lambda\partial\beta} = -e^{\beta} S_1,$$

where $S_1 = \sum_{x_i=1} t_i$ and $S_0 = \sum_{x_i=0} t_i$.

Thus the hessian matrix is

$$H(\lambda,\beta) = \begin{bmatrix} -\frac{m}{\lambda^2} & -e^{\beta} S_1 \\ -e^{\beta} S_1 & -\lambda e^{\beta} S_1 \end{bmatrix}.$$

(5)

Newton-Raphson algorithm 1:

Step 1: Set the initial point $\beta^{(0)}$ and $\lambda^{(0)}$;

Step 2: $$\begin{pmatrix} \beta^{(k+1)} \\ \lambda^{(k+1)} \end{pmatrix} = \begin{pmatrix} \beta^{(k)} \\ \lambda^{(k)} \end{pmatrix} - \begin{bmatrix} -\frac{m}{\lambda^{(k)2}} & -e^{\beta^{(k)}} S_1 \\ -e^{\beta^{(k)}} S_1 & -\lambda^{(k)} e^{\beta^{(k)}} S_1 \end{bmatrix}^{-1} \begin{pmatrix} \frac{m}{\lambda^{(k)}} - e^{\beta^{(k)}} S_1 - S_0 \\ m_1 - \lambda e^{\beta^{(k)}} S_1 \end{pmatrix};$$

Step 3: If $\left|\beta^{(k+1)} - \beta^{(k)}\right| < 10^{-5}$ and $\left|\lambda^{(k+1)} - \lambda^{(k)}\right| < 10^{-5}$, then stop. $\hat{\beta} = \beta^{(k+1)}$ and $\hat{\lambda} = \lambda^{(k+1)}$; Else go Step 2.

```
> Newton.Raphson(Score1, Hessian1, c(0.0001,0), 10^(-5))
Iteration 1 :  0.0001484716 4.467665
Iteration 2 :  0.0001422006 3.555903
Iteration 3 :  0.0001234207 2.8075
Iteration 4 :  4.932354e-05 2.698948
Iteration 5 :  0.0001099746 1.281182
Iteration 6 :  0.0001822218 1.127339
Iteration 7 :  0.0003197144 0.430823
Iteration 8 :  0.0004945833 0.09397844
Iteration 9 :  0.0006975007 -0.2207391
Iteration 10 :  0.0008414338 -0.3629226
Iteration 11 :  0.0008891024 -0.402656
Iteration 12 :  0.0008928361 -0.4054494
Iteration 13 :  0.0008928571 -0.4054651
Iteration 14 :  0.0008928571 -0.4054651
```

(6)

Substitute $\log\lambda$ by $\tilde{\lambda}$,

$$\ell(\tilde{\lambda},\beta) = m\log\lambda + m_1\beta - \lambda\sum_{i=1}^{n} t_i e^{\beta x_i}$$

$$= m\tilde{\lambda} + m_1\beta - e^{\tilde{\lambda}}\sum_{i=1}^{n} t_i e^{\beta x_i}.$$

The score functions are

$$\frac{\partial \ell(\tilde{\lambda},\beta)}{\partial \tilde{\lambda}} = m - e^{\tilde{\lambda}} \sum_{i=1}^{n} t_i e^{\beta x_i} = m - e^{\tilde{\lambda}} e^{\beta} S_1 - e^{\tilde{\lambda}} S_0$$

and

$$\frac{\partial \ell(\tilde{\lambda},\beta)}{\partial \beta} = m_1 - e^{\tilde{\lambda}} \sum_{i=1}^{n} t_i x_i e^{\beta x_i} = m_1 - e^{\tilde{\lambda}} e^{\beta} S_1 .$$

The second differentiate of the log-likelihood function are

$$\frac{\partial^2 \ell(\tilde{\lambda},\beta)}{\partial \tilde{\lambda}^2} = -e^{\tilde{\lambda}} e^{\beta} S_1 - e^{\tilde{\lambda}} S_0$$

$$\frac{\partial^2 \ell(\tilde{\lambda},\beta)}{\partial \beta^2} = -e^{\tilde{\lambda}} e^{\beta} S_1$$

$$\frac{\partial^2 \ell(\tilde{\lambda},\beta)}{\partial \tilde{\lambda} \partial \beta} = -e^{\tilde{\lambda}} e^{\beta} S_1 .$$

Thus the hessian matrix is

$$H(\tilde{\lambda},\beta) = \begin{bmatrix} -e^{\tilde{\lambda}} e^{\beta} S_1 - e^{\tilde{\lambda}} S_0 & -e^{\tilde{\lambda}} e^{\beta} S_1 \\ -e^{\tilde{\lambda}} e^{\beta} S_1 & -e^{\tilde{\lambda}} e^{\beta} S_1 \end{bmatrix}.$$

Newton-Raphson algorithm 2:

Step 1: Set the initial point $\beta^{(0)}$ and $\tilde{\lambda}^{(0)}$;

Step 2:

$$\begin{pmatrix} \beta^{(k+1)} \\ \tilde{\lambda}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \beta^{(k)} \\ \tilde{\lambda}^{(k)} \end{pmatrix} - \begin{bmatrix} -e^{\tilde{\lambda}^{(k)}} e^{\beta^{(k)}} S_1 - e^{\tilde{\lambda}^{(k)}} S_0 & -e^{\tilde{\lambda}^{(k)}} e^{\beta^{(k)}} S_1 \\ -e^{\tilde{\lambda}^{(k)}} e^{\beta^{(k)}} S_1 & -e^{\tilde{\lambda}^{(k)}} e^{\beta^{(k)}} S_1 \end{bmatrix}^{-1} \begin{pmatrix} m - e^{\tilde{\lambda}^{(k)}} e^{\beta^{(k)}} S_1 - e^{\tilde{\lambda}^{(k)}} S_0 \\ m_1 - e^{\tilde{\lambda}^{(k)}} e^{\beta^{(k)}} S_1 \end{pmatrix}$$

Step 3: If $\left|\beta^{(k+1)} - \beta^{(k)}\right| < 10^{-5}$ and $\left|\tilde{\lambda}^{(k+1)} - \tilde{\lambda}^{(k)}\right| < 10^{-5}$, then stop. $\hat{\beta} = \beta^{(k)}$ and $\hat{\tilde{\lambda}} = \tilde{\lambda}^{(k)}$; Else go Step 2.

```
> Newton.Raphson(Score2, Hessian2, c(log(0.0001),0), 10^(-5))
Iteration 1 :  -1.281769 -2.97619
Iteration 2 :  -2.278552 -2.937345
Iteration 3 :  -3.269835 -2.836432
Iteration 4 :  -4.246347 -2.59286
Iteration 5 :  -5.183981 -2.099423
Iteration 6 :  -6.024703 -1.392073
Iteration 7 :  -6.65549 -0.7710117
Iteration 8 :  -6.961705 -0.4648436
Iteration 9 :  -7.019355 -0.4071936
Iteration 10 :  -7.021082 -0.4054666
Iteration 11 :  -7.021084 -0.4054651
```

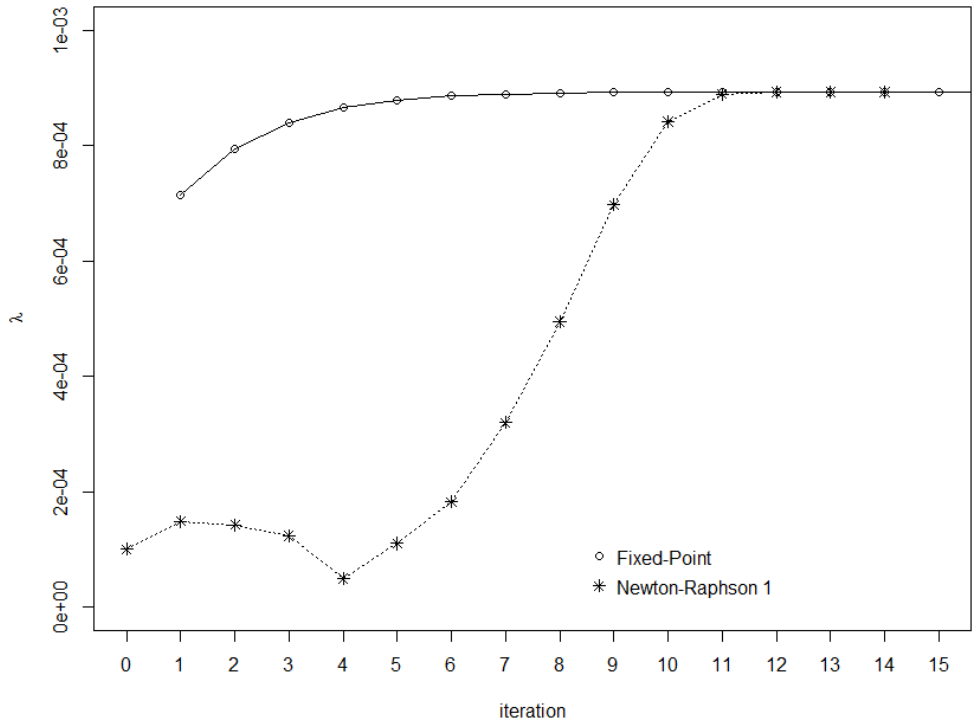
(7)

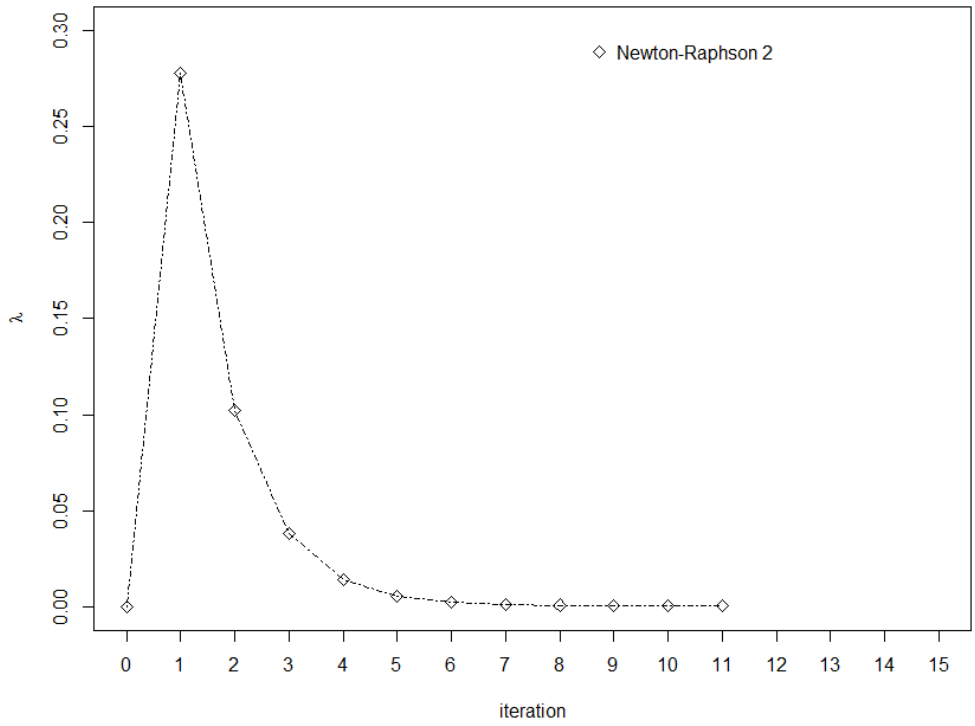
Iteration	Fixed-point iteration		Newton-Raphson 1		Newton-Raphson 2	
	λ	β	λ	β	$\tilde{\lambda}$	β
0	-	0	0.0001	0	log(0.0001)	0
1	0.000714286	-0.1823216	0.000148472	4.467665	-1.281769	-2.97619
2	0.000793651	-0.2876821	0.000142201	3.555903	-2.278552	-2.937345
3	0.000840336	-0.3448405	0.000123421	2.8075	-3.269835	-2.836432
4	0.000865801	-0.3746934	0.000049324	2.698948	-4.246347	-2.59286
5	0.000879121	-0.3899609	0.000109975	1.281182	-5.183981	-2.099423
6	0.000885936	-0.397683	0.000182222	1.127339	-6.024703	-1.392073
7	0.000889383	-0.4015665	0.000319714	0.430823	-6.65549	-0.7710117
8	0.000891117	-0.4035139	0.000494583	0.09397844	-6.961705	-0.4648436
9	0.000891986	-0.404489	0.000697501	-0.2207391	-7.019355	-0.4071936
10	0.000892421	-0.4049769	0.000841434	-0.3629226	-7.021082	-0.4054666
11	0.000892639	-0.405221	0.000889102	-0.402656	-7.021084	-0.4054651
12	0.000892748	-0.405343	0.000892836	-0.4054494	-	-
13	0.000892803	-0.4054041	0.000892857	-0.4054651	-	-
14	0.00089283	-0.4054346	0.000892857	-0.4054651	-	-
15	0.000892844	-0.4054498	-	-	-	-
16	0.00089285	-0.4054575	-	-	-	-

For fixed –point algorithm, it took 15 iterations to converge to the exact value calculated by MLEs. For Newton-Raphson algorithms, if we substitute $\log\lambda$ by $\tilde{\lambda}$, the number of iterations could reduce from 13 to 10, it seems to be more efficiency.

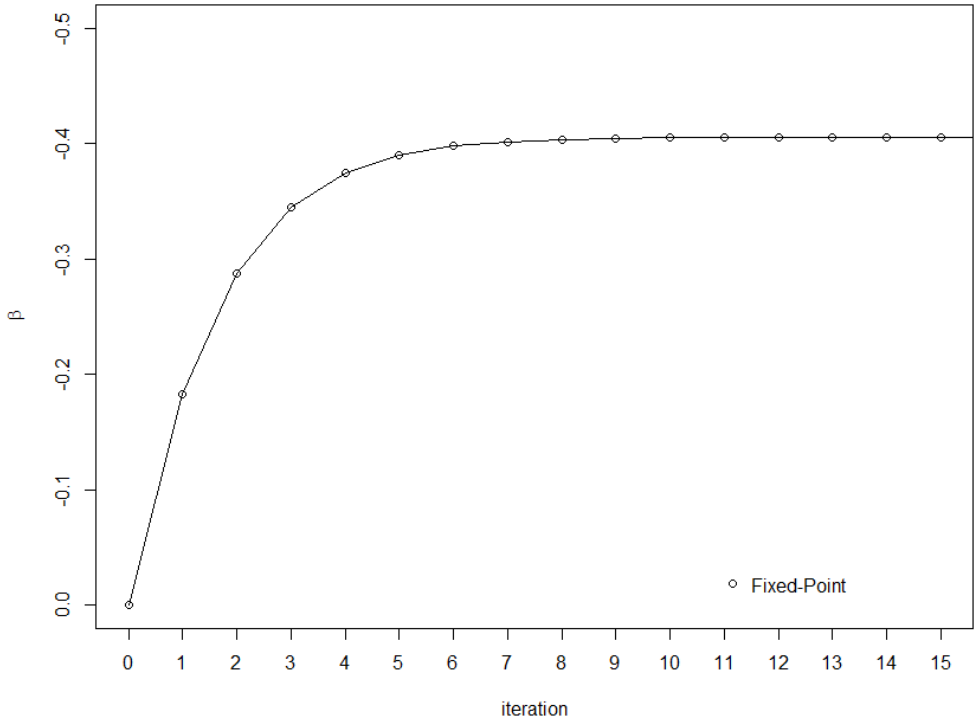
About choosing initial value, it is hard to decide which start point to use, since the data is quite small, any change on initial value will take different results. In these two Newton-Raphson algorithms, I choose $\lambda = 0.0001$ and $\beta = 0$, $\tilde{\lambda} = \log\lambda = \log 0.0001$ and $\beta = 0$ to be the initial points. The result seems good, $\tilde{\lambda}^{(11)}$ and $\lambda^{(14)}$ can transform to each other.

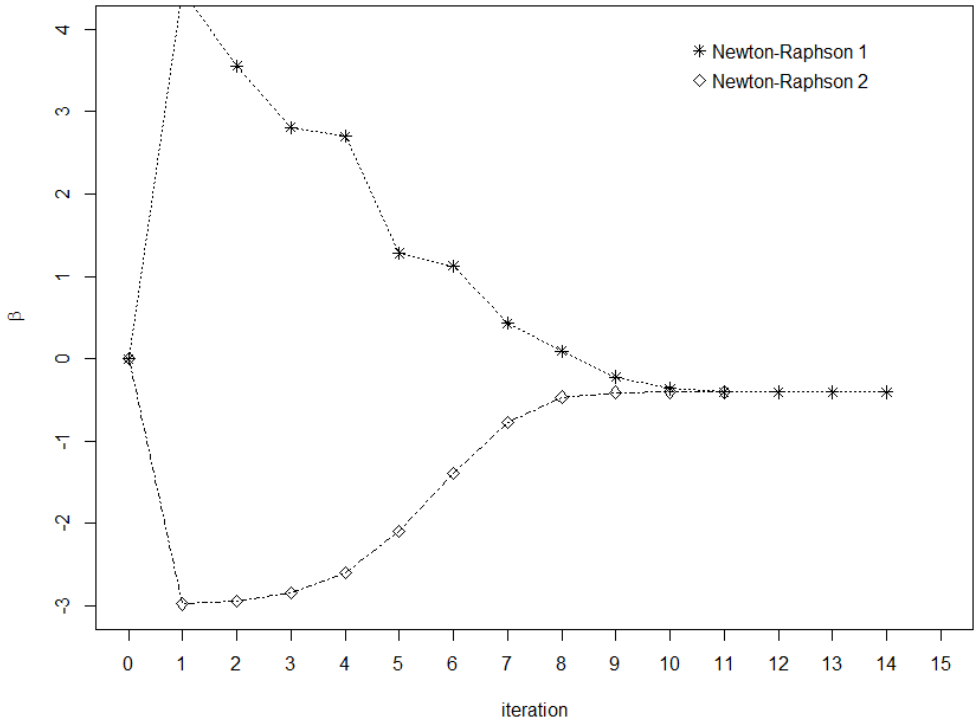
λ converge plot





β converge plot





Appendix R code

```
data = as.data.frame(
   cbind(death.time =c(1650, 30, 720, 450, 510, 1110, 210, 1380, 1800, 540),
            delta = c(0, 1, 0, 1, 1, 0, 1, 1, 0, 1),
            x = c(0,0,0,0,0,1,1,1,1,1)))
data = data[order(data$death.time,decreasing=F),]
for (i in c(1:nrow(data))){
   data$ni0[i] = sum(1-data$x[i:nrow(data)])
   data$ni1[i] = sum(data$x[i:nrow(data)])
}
m = sum(data$delta)
m1 = sum(data$delta[data$x == 1])
S0 = sum(data$death.time[data$x == 0])
S1 = sum(data$death.time[data$x == 1])
############Algorithm####################
###Fixed point
fixed.Point = function(func, initial, critirien){
   difference = rep(1, length(initial))
   j = 1
   input.Value = initial
   while (any(difference > critirien)){
      cat("Iteration",j,":    ")
      j = j + 1
      output.Value = func(input.Value)
      cat(output.Value, "\n")
      difference = abs(output.Value - input.Value)
      input.Value = output.Value
   }
}
```

```
###Newton Raphson
Newton.Raphson = function(S.Function, H.Function, initial, critirien){
   difference = rep(1, length(initial))
   input.Value = initial
   j = 1
   while (any(difference > critirien)){
      cat("Iteration",j,":    ")
      j = j + 1
      output.Value = input.Value - solve(H.Function(input.Value)) %*% S.Function(input.Value)
      cat(output.Value, "\n")
      difference = abs(output.Value - input.Value)
      input.Value = output.Value
   }
}
##############################################
###Fixed point test
iteration.Function4 = function(theta){
   lambda = theta[1]
   beta = theta[2]
   lambda.new = m / (exp(beta) * S1 + S0)
   beta.new = log(m1 / (lambda.new * S1))
   return(c(lambda.new, beta.new))
}
fixed.Point(iteration.Function4, c(0.00001,0), 10^(-5))
###Newton Rapson test
Hessian1 = function(theta){
   lambda = theta[1]
   beta = theta[2]
   h = matrix(c(-m / lambda^2, -exp(beta) * S1,
                -exp(beta) * S1, -lambda * exp(beta) * S1), nrow = 2)
```

```
   return(h)
}
Score1 = function(theta){
   lambda = theta[1]
   beta = theta[2]
   h = matrix(c(m / lambda - exp(beta) * S1 - S0,
                m1 - lambda * exp(beta) * S1), nrow = 2)
   return(h)
}
Newton.Raphson(Score1, Hessian1, c(0.0001,0), 10^(-5))
Hessian2 = function(theta){
   lambda = theta[1]
   beta = theta[2]
   h = matrix(c(- exp(lambda + beta) * S1 - exp(lambda) * S0,
                - exp(lambda + beta) * S1, - exp(lambda + beta) * S1,
                - exp(lambda + beta) * S1), nrow = 2)
   return(h)
}
Score2 = function(theta){
   lambda = theta[1]
   beta = theta[2]
   h = matrix(c(m - exp(lambda + beta) * S1 - exp(lambda) * S0,
                m1 - exp(lambda + beta) * S1), nrow = 2)
   return(h)
}
Newton.Raphson(Score2, Hessian2, c(log(0.0001),0), 10^(-5))
```