Supplement_3   Survival Analysis I

NAME: JIA-HAN SHIH

Problem 1.   (Exercise 4, p.31)

Suppose that data (t_i, δ_i, x_i), $i=1,2,\ldots,n$, follow the model $S(t\,|\,x_i)=\exp(-\lambda t e^{\beta x_i})$, where $\lambda>0$ and $-\infty<\beta<\infty$. Let $m=\Sigma_{i=1}^n \delta_i$ be the number of deaths.

(1) Write down the log-likelihood function $\ell(\lambda,\beta)=\log L(\lambda,\beta)$ (under the Clayton copula with dependent censoring).

Solution (1).

The Clayton copula is defined as

$$C_\alpha(w,v)=(w^{-\alpha}+v^{-\alpha}-1)^{-1/\alpha}, \qquad 0<w,v<1,\ \ \alpha>0\,.$$

We assume the common margins for survival time (T) and censoring time (U), that is

$$S_T(t\,|\,x_i)=S_U(t\,|\,x_i)=\exp(-\lambda t e^{\beta x_i})\,.$$

Then, the joint survival function is

$$S_{TU,\alpha}(t,u)=C_\alpha\{S_T(t\,|\,x),S_U(u\,|\,x)\}=\{S_T(t\,|\,x)^{-\alpha}+S_U(u\,|\,x)^{-\alpha}-1\}^{-1/\alpha}$$

$$=\{\exp(\alpha\lambda t e^{\beta x})+\exp(\alpha\lambda u e^{\beta x})-1\}^{-1/\alpha}.$$

The sub-density functions are (they are the same due to the common margins assumption)

$$f^{\#}_{T,\alpha}(t\,|\,x)=f^{\#}_{U,\alpha}(t\,|\,x)=-\frac{\partial}{\partial y}S_{TU,\alpha}(t,y\,|\,x)\bigg|_{y=t}=\frac{\lambda\exp(\beta x+\alpha\lambda t e^{\beta x})}{\{2\exp(\alpha\lambda t e^{\beta x})-1\}^{1/\alpha+1}}\,.$$

Therefore, based on data (t_i,δ_i,x_i), $i=1,2,\ldots,n$, the log-likelihood function is

$$\ell(\lambda,\beta)=\sum_{i=1}^n \delta_i\log f^{\#}_{T,\alpha}(t_i\,|\,x_i)+\sum_{i=1}^n(1-\delta_i)\log f^{\#}_{U,\alpha}(t_i\,|\,x_i)=\sum_{i=1}^n\log f^{\#}_{T,\alpha}(t_i\,|\,x_i)$$

$$=n\log\lambda+\beta\sum_{i=1}^n x_i+\alpha\lambda\sum_{i=1}^n t_i e^{\beta x_i}-\frac{\alpha+1}{\alpha}\sum_{i=1}^n\log\{2\exp(\alpha\lambda t_i e^{\beta x_i})-1\}.$$

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

Solution (2).

By straightforward calculations, we have

$$S_1(\lambda,\beta)=\frac{\partial}{\partial\lambda}\ell(\lambda,\beta)=\frac{n}{\lambda}+\alpha\sum_{i=1}^{n}t_i e^{\beta x_i}-\frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{2\alpha t_i\exp(\beta x_i+\alpha\lambda t_i e^{\beta x_i})}{2\exp(\alpha\lambda t_i e^{\beta x_i})-1},$$

$$S_2(\lambda,\beta)=\frac{\partial}{\partial\beta}\ell(\lambda,\beta)=\sum_{i=1}^{n}x_i+\alpha\lambda\sum_{i=1}^{n}t_i x_i e^{\beta x_i}-\frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{2\alpha\lambda t_i x_i\exp(\beta x_i+\alpha\lambda t_i e^{\beta x_i})}{2\exp(\alpha\lambda t_i e^{\beta x_i})-1}.$$

Thus, the score vector is

$$\mathbf{S}(\lambda,\beta)=[\,S_1(\lambda,\beta),S_2(\lambda,\beta)\,]^{\mathrm{T}}.$$

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

Solution (3).

One can obtain the maximum likelihood estimator (MLE) by applying the fixed-point iteration algorithm as follows:

Algorithm 1: The fixed-point iteration algorithm

Step 1 Choose initial value $\lambda^{(0)}$ and $\beta^{(0)}$.

Step 2 Update the value λ by

$$\lambda^{(k+1)}=\left\{\frac{\alpha+1}{\alpha n}\sum_{i=1}^{n}\frac{2\alpha t_i\exp(\beta^{(k)}x_i+\alpha\lambda^{(k)}t_i e^{\beta^{(k)}x_i})}{2\exp(\alpha\lambda^{(k)}t_i e^{\beta^{(k)}x_i})-1}-\frac{\alpha}{n}\sum_{i=1}^{n}t_i e^{\beta^{(k)}x_i}\right\}^{-1}.$$

Step 3 Update the value β by $\beta^{(k+1)}$ which is the solution of equation

$$\frac{\partial}{\partial\beta}\ell(\lambda^{(k+1)},\beta)=\sum_{i=1}^{n}x_i+\alpha\lambda^{(k+1)}\sum_{i=1}^{n}t_i x_i e^{\beta x_i}-\frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{2\alpha\lambda^{(k+1)}t_i x_i\exp(\beta x_i+\alpha\lambda^{(k+1)}t_i e^{\beta x_i})}{2\exp(\alpha\lambda^{(k+1)}t_i e^{\beta x_i})-1}=0.$$

Step 4 Repeat Step 2 and Step 3 as $k=0,1,2,\ldots$.

- If $\max\{\,|\lambda^{(k+1)}-\lambda^{(k)}|,|\beta^{(k+1)}-\beta^{(k)}|\,\}\le 10^{-6}$ then stop and $(\lambda^{(k)},\beta^{(k)})$ is the MLE.
- The equation in Step 3 is solved by R function *uniroot*.

We apply Algorithm 1 to the data in Example 1 with the initial value given by $(\lambda^{(0)}, \beta^{(0)}) = (0.0001, 0)$, $(0.001, 0)$, $(0.001, 0)$ for $\alpha = 0.01, 2, 8$, respectively. The results are given in Table 1. It shows that the case $\alpha = 0.01$ produces a larger log-likelihood value. This may indicate that the dependence between survival time and censoring time is weak.

Table 1. The results (Algorithm 1) based on the data in Example 1 under $\alpha = 0.01, 2, 8$.

α	$\hat{\beta}$	$\hat{\lambda}$	Log-likelihood	Iteration number (k)
0.01	0.000746	-0.407045	-84.07	19
2.00	0.001171	-0.448781	-84.85	29
8.00	0.001417	-0.406267	-84.31	20

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

Solution (4).

By straightforward calculations, we have

$$H_{11}(\lambda, \beta) = \frac{\partial}{\partial \lambda^2}\ell(\lambda, \beta) = -\frac{n}{\lambda^2} - \frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{2\alpha^2 t_i^2 \exp(2\beta x_i + \alpha\lambda t_i e^{\beta x_i})}{2\exp(\alpha\lambda t_i e^{\beta x_i}) - 1}$$
$$+ \frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{4\alpha^2 t_i^2 \exp(2\beta x_i + 2\alpha\lambda t_i e^{\beta x_i})}{\{2\exp(\alpha\lambda t_i e^{\beta x_i}) - 1\}^2},$$

$$H_{12}(\lambda, \beta) = \frac{\partial}{\partial\lambda\partial\beta}\ell(\lambda, \beta) = \alpha\sum_{i=1}^{n} t_i x_i e^{\beta x_i} - \frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{2\alpha t_i x_i \exp(\beta x_i + \alpha\lambda t_i e^{\beta x_i})(1 + \alpha\lambda t_i e^{\beta x_i})}{2\exp(\alpha\lambda t_i e^{\beta x_i}) - 1}$$
$$+ \frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{4\alpha^2 t_i^2 \exp(\beta x_i + \alpha\lambda t_i e^{\beta x_i})(1 + \lambda x_i)}{\{2\exp(\alpha\lambda t_i e^{\beta x_i}) - 1\}^2},$$

$$H_{22}(\lambda, \beta) = \frac{\partial}{\partial\beta^2}\ell(\lambda, \beta) = \frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{2\alpha\lambda t_i x_i \exp(\beta x_i + \alpha\lambda t_i e^{\beta x_i})(x_i + \alpha\lambda t_i x_i e^{\beta x_i})}{2\exp(\alpha\lambda t_i e^{\beta x_i}) - 1}$$
$$+ \alpha\lambda\sum_{i=1}^{n} t_i x_i^2 e^{\beta x_i} - \frac{\alpha+1}{\alpha}\sum_{i=1}^{n}\frac{4\alpha^2\lambda^2 t_i^2 x_i^2 \exp(2\beta x_i + 2\alpha\lambda t_i e^{\beta x_i})}{\{2\exp(\alpha\lambda t_i e^{\beta x_i}) - 1\}^2}.$$

Thus, the Hessian matrix is

$$\mathbf{H}(\lambda,\beta)=\begin{bmatrix} H_{11}(\lambda,\beta) & H_{12}\ell(\lambda,\beta) \\ H_{12}\ell(\lambda,\beta) & H_{22}\ell(\lambda,\beta) \end{bmatrix}.$$

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

Solution (5).

One can obtain the MLE by using the Newton-Raphson algorithm as follows:

Algorithm 2: The Newton-Raphson algorithm

Step 1. Choose initial value $(\lambda^{(0)},\beta^{(0)})$.

Step 2. Repeat the Newton-Raphson iteration

$$\begin{bmatrix} \lambda^{(k+1)} \\ \beta^{(k+1)} \end{bmatrix}=\begin{bmatrix} \lambda^{(k)} \\ \beta^{(k)} \end{bmatrix}-\mathbf{H}(\lambda^{(k)},\beta^{(k)})^{-1}\mathbf{S}(\lambda^{(k)},\beta^{(k)}),$$

where the expressions of $\mathbf{S}(\lambda,\beta)$ and $\mathbf{H}(\lambda,\beta)$ are given in (2) and (4).

- If $\max\{|\lambda^{(k+1)}-\lambda^{(k)}|,|\beta^{(k+1)}-\beta^{(k)}|\}\le 10^{-6}$ then stop and $(\lambda^{(k)},\beta^{(k)})$ is the MLE.

We apply Algorithm 2 to the data in Example 1 with the initial value given by $(\lambda^{(0)},\beta^{(0)})=(0.0001,0)$, $(0.001,0)$, $(0.001,0)$ for $\alpha=0.01,2,8$, respectively. The results are given in Table 2 and it agrees with Table 1.

Table 2. The results (Algorithm 2) based on the data in Example 1 under $\alpha=0.01,2,8$.

α	$\hat{\beta}$	$\hat{\lambda}$	Log-likelihood	Iteration number (k)
0.01	0.000746	-0.407042	-84.07	11
2.00	0.001171	-0.448782	-84.85	4
8.00	0.001417	-0.406265	-84.31	4

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

Solution (6).

Under the transformed parameter, the log-likelihood becomes

$$\tilde{\ell}(\tilde{\lambda}, \beta) = \ell(e^{\tilde{\lambda}}, \beta).$$

By applying the Chain Rule, the score vector becomes

$$\tilde{\mathbf{S}}(\tilde{\lambda}, \beta) = \left[\frac{\partial}{\partial \tilde{\lambda}} \ell(e^{\tilde{\lambda}}, \beta) \quad \frac{\partial}{\partial \beta} \ell(e^{\tilde{\lambda}}, \beta) \right]^{\mathrm{T}} = \left[\frac{\partial}{\partial e^{\tilde{\lambda}}} \ell(e^{\tilde{\lambda}}, \beta) \frac{\partial}{\partial \tilde{\lambda}} e^{\tilde{\lambda}} \quad S_2(e^{\tilde{\lambda}}, \beta) \right]^{\mathrm{T}}$$
$$= [\, S_1(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} \quad S_2(e^{\tilde{\lambda}}, \beta) \,]^{\mathrm{T}}.$$

Similarly, the Hessian matrix becomes

$$\tilde{\mathbf{H}}(\tilde{\lambda}, \beta) = \begin{bmatrix} \frac{\partial}{\partial \tilde{\lambda}^2} \ell(e^{\tilde{\lambda}}, \beta) & \frac{\partial}{\partial \tilde{\lambda} \partial \beta} \ell(e^{\tilde{\lambda}}, \beta) \\ \frac{\partial}{\partial \tilde{\lambda} \partial \beta} \ell(e^{\tilde{\lambda}}, \beta) & \frac{\partial}{\partial \beta^2} \ell(e^{\tilde{\lambda}}, \beta) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \tilde{\lambda}} \{ S_1(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} \} & \frac{\partial}{\partial \beta} S_1(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} \\ \frac{\partial}{\partial \beta} S_1(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} & H_{22}(e^{\tilde{\lambda}}, \beta) \end{bmatrix}$$
$$= \begin{bmatrix} \frac{\partial}{\partial \tilde{\lambda}} S_1(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} + S_1(e^{\tilde{\lambda}}, \beta) \frac{\partial}{\partial \tilde{\lambda}} e^{\tilde{\lambda}} & H_{12}(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} \\ H_{12}(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} & H_{22}(e^{\tilde{\lambda}}, \beta) \end{bmatrix}$$
$$= \begin{bmatrix} \frac{\partial}{\partial e^{\tilde{\lambda}}} S_1(e^{\tilde{\lambda}}, \beta) \frac{\partial}{\partial \tilde{\lambda}} e^{\tilde{\lambda}} e^{\tilde{\lambda}} + S_1(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} & e^{\tilde{\lambda}} H_{12}(e^{\tilde{\lambda}}, \beta) \\ e^{\tilde{\lambda}} H_{12}(e^{\tilde{\lambda}}, \beta) & H_{22}(e^{\tilde{\lambda}}, \beta) \end{bmatrix}$$
$$= \begin{bmatrix} H_{11}(e^{\tilde{\lambda}}, \beta) e^{2\tilde{\lambda}} + S_1(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} & H_{12}(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} \\ H_{12}(e^{\tilde{\lambda}}, \beta) e^{\tilde{\lambda}} & H_{22}(e^{\tilde{\lambda}}, \beta) \end{bmatrix}.$$

Then, one can obtain the MLE by the Newton-Raphson algorithm with transformed parameter as follows:

Algorithm 3: The Newton-Raphson algorithm with transformed parameter

Step 1. Choose initial value $(\tilde{\lambda}^{(0)}, \beta^{(0)})$.

Step 2. Repeat the Newton-Raphson iteration

$$\begin{bmatrix} \tilde{\lambda}^{(k+1)} \\ \beta^{(k+1)} \end{bmatrix} = \begin{bmatrix} \tilde{\lambda}^{(k)} \\ \beta^{(k)} \end{bmatrix} - \tilde{\mathbf{H}}(\tilde{\lambda}^{(k)}, \beta^{(k)})^{-1} \tilde{\mathbf{S}}(\tilde{\lambda}^{(k)}, \beta^{(k)}) .$$

- If $\max\{ |\tilde{\lambda}^{(k+1)} - \tilde{\lambda}^{(k)}|, |\beta^{(k+1)} - \beta^{(k)}| \} \le 10^{-6}$ then stop and $(\tilde{\lambda}^{(k)}, \beta^{(k)})$ is the MLE.

We apply Algorithm 3 to the data in Example 1 with the initial value given by $(\lambda^{(0)}, \beta^{(0)}) = (0.0001, 0)$, $(0.001, 0)$, $(0.001, 0)$ for $\alpha = 0.01, 2, 8$, respectively. The results are given in Table 3 and it agrees with Tables 1 and 2.

Table 3. The results (Algorithm 3) based on the data in Example 1 under $\alpha = 0.01, 2, 8$.

α	$\hat{\beta}$	$\hat{\lambda}$	Log-likelihood	Iteration number (k)
0.01	0.000746	-0.407045	-84.07	9
2.00	0.001171	-0.448782	-84.85	3
8.00	0.001417	-0.406265	-84.31	4

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

Solution (7).

Among these three algorithms, the convergence speed of fixed-point iteration algorithm is the slowest. The Newton-Raphson algorithm with the transformed parameter converges slightly quicker (0 - 1 iterations) than the untransformed algorithm.

In the aspect of convergence speed, one may think that there is only small improvement by using the Newton-Raphson algorithm with the transformed parameter. However, it can reduce the sensitivity of the initial value. For example, Algorithm 3 can converge properly under the choice of $(\lambda^{(0)}, \beta^{(0)}) = (1, 0)$ but Algorithm 2 cannot.

Appendix 1 R codes for Algorithm 1

```
logL.func = function(para) {

  beta.para    = para[1]
  lambda.para = para[2]

  C1 = alpha*lambda.para*sum(t.event*exp(beta.para*x))
  C2 = (1/alpha+1)*sum(log(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

  return(n*log(lambda.para)+beta.para*sum(x)+C1-C2)

}
opt.logL.func = function(para) {

  beta.para    = para[1]
  lambda.para = para[2]

  C1 = alpha*lambda.para*sum(t.event*exp(beta.para*x))
  C2 = (1/alpha+1)*sum(log(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

  return(-(n*log(lambda.para)+beta.para*sum(x)+C1-C2))

}
fp1.func = function(para) {

  beta.para    = para[1]
  lambda.para = para[2]

  C3 = alpha*sum(t.event*exp(beta.para*x))
  c4 = 2*alpha*t.event*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*x)
  C4 = (1/alpha+1)*sum(c4/(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

  return(1/(C4/n-C3/n))

}
```

```
fp2.func = function(beta.para) {

   C1 = alpha*lambda.para*sum(t.event*x*exp(beta.para*x))
   c2 = 2*alpha*lambda.para*t.event*x*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*x)
   C2 = (1/alpha+1)*sum(c2/(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

   return(sum(x)+C1-C2)

}

t.event = c(1650,30,720,450,510,1110,210,1380,1800,540)
x = c(0,0,0,0,0,1,1,1,1,1)
n = length(t.event)
epsilon = 1e-6

#alpha = 0.01; ini.para = c(0,0.0001)
alpha = 2; ini.para = c(0,0.001)
#alpha = 8; ini.para = c(0,0.001)

k = 0
para.old = ini.para
para.new = c(0,0)
repeat{

   cat("k = ",k,", para = ",round(para.old,6),", log.L = ",round(logL.func(para.old),2),"\n")
   para.new[2] = fp1.func(para.old)
   lambda.para = para.new[2]
   para.new[1] = uniroot(fp2.func,c(-1,1))$root

   if(max(abs(para.new-para.old)) < epsilon) {break}
   k = k+1
   para.old = para.new

}
```

Appendix 2 R codes for Algorithm 2

```
logL.func = function(para) {

   beta.para     = para[1]
   lambda.para = para[2]

   C1 = alpha*lambda.para*sum(t.event*exp(beta.para*x))
   C2 = (1/alpha+1)*sum(log(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

   return(n*log(lambda.para)+beta.para*sum(x)+C1-C2)

}
opt.logL.func = function(para) {

   beta.para     = para[1]
   lambda.para = para[2]

   C1 = alpha*lambda.para*sum(t.event*exp(beta.para*x))
   C2 = (1/alpha+1)*sum(log(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

   return(-(n*log(lambda.para)+beta.para*sum(x)+C1-C2))

}
score.func = function(para) {

   beta.para     = para[1]
   lambda.para = para[2]

   C1 = alpha*lambda.para*sum(t.event*x*exp(beta.para*x))
   c2                                                                                                  =
2*alpha*lambda.para*t.event*x*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*
x)
   C2 = (1/alpha+1)*sum(c2/(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

   S1 = sum(x)+C1-C2
```

```
    C3 = alpha*sum(t.event*exp(beta.para*x))
    c4 = 2*alpha*t.event*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*x)
    C4 = (1/alpha+1)*sum(c4/(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

    S2 = n/lambda.para+C3-C4

    return(c(S1,S2))

}
hessian.func = function(para) {

    beta.para      = para[1]
    lambda.para = para[2]

    C1 = alpha*lambda.para*sum(t.event*x^2*exp(beta.para*x))
    c21 = 2*alpha*lambda.para*t.event*x*(x+alpha*lambda.para*t.event*x*exp(beta.para*x))
    c22 = exp(beta.para*x+alpha*lambda.para*t.event*exp(beta.para*x))
    c23 = 2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1
    c24 = 4*alpha^2*lambda.para^2*t.event^2*x^2
    c25 = exp(2*beta.para*x+2*alpha*lambda.para*t.event*exp(beta.para*x))
    C2                                                                                    =
(1/alpha+1)*sum((c21*c22*c23-c24*c25)/(2*exp(alpha*lambda.para*t.event*exp(beta.para*
x))-1)^2)

    H1 = C1-C2

    C3 = alpha*sum(t.event*x*exp(beta.para*x))
    c41 = 2*alpha*t.event*x*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*x)
    c42 = 1+alpha*lambda.para*t.event*exp(beta.para*x)
    c43 = 2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1
    c44                                                                                   =
2*alpha*lambda.para*t.event*x*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*
x)
    c45 = 2*alpha*t.event*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*x)
    C4                                                                                    =
(1/alpha+1)*sum((c41*c42*c43-c44*c45)/(2*exp(alpha*lambda.para*t.event*exp(beta.para*
x))-1)^2)
```

```
    H2 = C3-C4

    c51 = 2*alpha^2*t.event^2*exp(alpha*lambda.para*t.event*exp(beta.para*x)+2*beta.para*x)
    c52 = 2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1
    c53 = 4*alpha^2*t.event^2*exp(2*alpha*lambda.para*t.event*exp(beta.para*x)+2*beta.para*x)
    C5 = (1/alpha+1)*sum((c51*c52-c53)/(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1)^2)

    H3 = -n/lambda.para^2-C5

    return(matrix(c(H1,H2,H2,H3),2,2))

}

#library(numDeriv)
#t.event = c(1.2,2.2)
#x = c(0.3,0.5)
#n = length(x)
#alpha = 2
#para = c(0.7,0.5)
#score.func(para)
#grad(logL.func,para)
#hessian.func(para)
#hessian(logL.func,para)

t.event = c(1650,30,720,450,510,1110,210,1380,1800,540)
x = c(0,0,0,0,0,1,1,1,1,1)
n = length(t.event)
epsilon = 1e-6

#alpha = 0.01; ini.para = c(0,0.0001)
#alpha = 2; ini.para = c(0,0.001)
alpha = 8; ini.para = c(0,0.001)
```

```
k = 0
para.old = ini.para
repeat{

  cat("k = ",k,", para = ",round(para.old,6),", log.L = ",round(logL.func(para.old),2),"\n")
  para.new = para.old-solve(hessian.func(para.old))%*%score.func(para.old)

  if(max(abs(para.new-para.old)) < epsilon) {break}
  k = k+1
  para.old = para.new

}
```

Appendix 3 R codes for Algorithm 3

```
t.logL.func = function(para) {

   beta.para    = para[1]
   lambda.para = exp(para[2])

   C1 = alpha*lambda.para*sum(t.event*exp(beta.para*x))
   C2 = (1/alpha+1)*sum(log(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

   return(n*log(lambda.para)+beta.para*sum(x)+C1-C2)

}
t.opt.logL.func = function(para) {

   beta.para    = para[1]
   lambda.para = exp(para[2])

   C1 = alpha*lambda.para*sum(t.event*exp(beta.para*x))
   C2 = (1/alpha+1)*sum(log(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

   return(-(n*log(lambda.para)+beta.para*sum(x)+C1-C2))

}
t.score.func = function(para) {

   beta.para    = para[1]
   lambda.para = exp(para[2])

   C1 = alpha*lambda.para*sum(t.event*x*exp(beta.para*x))
   c2                                                                                          =
2*alpha*lambda.para*t.event*x*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*
x)
   C2 = (1/alpha+1)*sum(c2/(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

   S1 = sum(x)+C1-C2
```

```
    C3 = alpha*sum(t.event*exp(beta.para*x))
    c4 = 2*alpha*t.event*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*x)
    C4 = (1/alpha+1)*sum(c4/(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1))

    S2 = lambda.para*(n/lambda.para+C3-C4)

    return(c(S1,S2))

}
t.hessian.func = function(para) {

    beta.para      = para[1]
    lambda.para = exp(para[2])

    C1 = alpha*lambda.para*sum(t.event*x^2*exp(beta.para*x))
    c21 = 2*alpha*lambda.para*t.event*x*(x+alpha*lambda.para*t.event*x*exp(beta.para*x))
    c22 = exp(beta.para*x+alpha*lambda.para*t.event*exp(beta.para*x))
    c23 = 2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1
    c24 = 4*alpha^2*lambda.para^2*t.event^2*x^2
    c25 = exp(2*beta.para*x+2*alpha*lambda.para*t.event*exp(beta.para*x))
    C2                                                                                          =
(1/alpha+1)*sum((c21*c22*c23-c24*c25)/(2*exp(alpha*lambda.para*t.event*exp(beta.para*
x))-1)^2)

    H1 = C1-C2

    C3 = alpha*sum(t.event*x*exp(beta.para*x))
    c41 = 2*alpha*t.event*x*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*x)
    c42 = 1+alpha*lambda.para*t.event*exp(beta.para*x)
    c43 = 2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1
    c44                                                                                         =
2*alpha*lambda.para*t.event*x*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*
x)
    c45 = 2*alpha*t.event*exp(alpha*lambda.para*t.event*exp(beta.para*x)+beta.para*x)
    C4                                                                                          =
(1/alpha+1)*sum((c41*c42*c43-c44*c45)/(2*exp(alpha*lambda.para*t.event*exp(beta.para*
x))-1)^2)
```

```
    H2 = lambda.para*(C3-C4)

    c51 = 2*alpha^2*t.event^2*exp(alpha*lambda.para*t.event*exp(beta.para*x)+2*beta.para*x)
    c52 = 2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1
    c53 = 4*alpha^2*t.event^2*exp(2*alpha*lambda.para*t.event*exp(beta.para*x)+2*beta.para*x)
    C5 = (1/alpha+1)*sum((c51*c52-c53)/(2*exp(alpha*lambda.para*t.event*exp(beta.para*x))-1)^2)

    H3 = lambda.para^2*(-n/lambda.para^2-C5)+t.score.func(para)[2]

    return(matrix(c(H1,H2,H2,H3),2,2))

}

#library(numDeriv)
#t.event = c(1.2,2.2)
#x = c(0.3,0.5)
#n = length(x)
#alpha = 2
#para = c(0.7,0.5)
#t.score.func(para)
#grad(t.logL.func,para)
#t.hessian.func(para)
#hessian(t.logL.func,para)

t.event = c(1650,30,720,450,510,1110,210,1380,1800,540)
x = c(0,0,0,0,0,1,1,1,1,1)
n = length(t.event)
epsilon = 1e-6

alpha = 0.01; ini.para = c(0,log(0.0001))
#alpha = 2; ini.para = c(0,log(0.001))
#alpha = 8; ini.para = c(0,log(0.001))
```

```
#alpha = 0.01; ini.para = c(0,0)
#alpha = 2; ini.para = c(0,-3)
#alpha = 8; ini.para = c(0,-4)

k = 0
para.old = ini.para
repeat{

   cat("k = ",k,", para = ",round(c(para.old[1],exp(para.old[2])),6),
         ", t.para = ",round(para.old,6),", log.L = ",round(t.logL.func(para.old),2),"\n")
   para.new = para.old-solve(t.hessian.func(para.old))%*%t.score.func(para.old)

   if(max(abs(para.new-para.old)) < epsilon) {break}
   k = k+1
   para.old = para.new

}
```