07-30-2023

Hengde Ouyang

2023-07-30

1. AFT Model Simulation

General Formulation:

$$\tilde{T}_i = \beta_0 + \beta^T Z_i + \sigma e_i \ , \ E(e_i) = 0$$

$$\tilde{T}_i = log(T_i)$$

$$e_i \sim N(0,1), \ T_i \sim logNormal$$

$$e_i \sim standard \ EV, \ T_i \sim Weibull$$

$$e_i \sim standard \ logistic, \ T_i \sim log - logistic$$

Code (Log Normal):

```
# Log Normal

AFT_Simulation <- function(n,beta,lower,upper,
                           lower_cens, upper_cens){
  x = runif(length(beta)*n,lower,upper)
  X = matrix(x,n,length(beta))


  time = exp(X%*%beta+rnorm(n))
  cens_time <- runif(n, min=lower_cens, max=upper_cens)
  Y <- pmin(time, cens_time)
  delta <- ifelse(time <= cens_time, 1, 0)
  return(list(Y=Y, delta=delta, Time = time,
              X = X))
}


tst <- AFT_Simulation(500, beta=c(1,2,3), lower=0, upper=1,
                      lower_cens=0, upper_cens=300)
```

```r
table(tst$delta)
```

```
##
##   0   1
##  83 417
```

```r
AFT_EXP = summary(survreg(Surv(tst$Y, tst$delta) ~ tst$X,
        dist="lognormal"),scale=1)
AFT_EXP
```

```
##
## Call:
## survreg(formula = Surv(tst$Y, tst$delta) ~ tst$X, dist = "lognormal")
##              Value Std. Error     z       p
## (Intercept) 0.1909     0.1449  1.32 0.18756
## tst$X1      0.6156     0.1729  3.56 0.00037
## tst$X2      2.0351     0.1728 11.78 < 2e-16
## tst$X3      3.0226     0.1664 18.17 < 2e-16
## Log(scale)  0.0521     0.0352  1.48 0.13845
##
## Scale= 1.05
##
## Log Normal distribution
## Loglik(model)= -1774.4   Loglik(intercept only)= -1960.3
##  Chisq= 371.75 on 3 degrees of freedom, p= 2.9e-80
## Number of Newton-Raphson Iterations: 5
## n= 500
```

2. Ranking Credible Interval

```
system.time({

  result_GP = MH_GP_Sampling(tti,Y,Y.test,delta,delta.test,tau,
                         A,A.all,beta0,alpha0,v0,kappa,
                         m,B,eta,K.all,
                         Wmat_option=0)



})
```

```
##    user  system elapsed
## 4944.86  187.27 6456.03
```

```
get_quantiles <- function(x) {
  quantile(x, c(0.025, 0.975))

}

Quan =  apply(result_GP$Rank,2,get_quantiles)
All_Rank = data.frame(True_Rank =
                      rank(tst$X[(tti+1):length(tst$Time),]%*%c(1,2,3)),
                      Lower = Quan[1,],
                      Upper = Quan[2,])
All_Rank
```

```
##     True_Rank  Lower   Upper
## 1          19  7.000  22.000
## 2          80 45.000  89.000
## 3          23 10.000  28.000
## 4          87 87.000 100.000
## 5          12  3.000  17.000
## 6          40 22.000  63.000
## 7          52 24.000  76.000
## 8          43 27.000  69.000
## 9          46 22.000  72.000
## 10         64 37.000  92.000
## 11         54 26.000  75.000
## 12         79 43.000  92.000
## 13         13  2.000  36.000
## 14          1  1.000  21.000
## 15         47 27.000  73.000
## 16         24 13.000  36.000
## 17         30 16.000  51.000
## 18          3  1.000  10.000
## 19         81 30.000  92.000
## 20          2  1.000  12.000
## 21         73 58.000  92.000
## 22         53 35.000  80.000
## 23         78 31.000  88.000
```

```
## 24         88 62.000  97.000
## 25         34 15.000  75.000
## 26         42 29.000  70.000
## 27         89 55.000  98.000
## 28         55 30.000  73.000
## 29         90 45.000  96.000
## 30         99 63.000 100.000
## 31         86 55.000  95.000
## 32         20 10.000  41.000
## 33         35 21.000  74.000
## 34         48 37.000  83.000
## 35         85 26.000  88.000
## 36         21  9.000  37.000
## 37         65 30.000  94.000
## 38         95 58.000  99.000
## 39         84 77.000  99.000
## 40          7  1.000  18.000
## 41         36 17.000  51.000
## 42         93 78.000 100.000
## 43         11  3.000  22.525
## 44         63 39.000  83.000
## 45         26 17.000  63.000
## 46         10  2.000  27.000
## 47          5  1.000   9.000
## 48         60 33.000  83.000
## 49         58 28.000  83.000
## 50         38 26.000  78.000
## 51         29 15.000  63.000
## 52         45 20.000  67.000
## 53         32 22.000  65.000
## 54          6  1.000  25.000
## 55         62 34.000  84.000
## 56        100 37.000 100.000
## 57         22 14.000  53.000
## 58         70 54.000  92.000
## 59          4  1.000  21.000
## 60         44 25.000  70.000
## 61         27 22.000  60.000
## 62         17 11.000  48.000
## 63         71 47.000  90.000
## 64         72 59.000  92.000
## 65         68 31.000  79.000
## 66         94 90.000 100.000
## 67         96 41.000  98.000
## 68         37 19.000  57.000
## 69         33 21.000  53.000
## 70         28 17.000  42.000
## 71         15  5.000  45.000
## 72         50 22.000  83.000
## 73         67 59.000  95.000
## 74         16  6.000  25.000
## 75         77 50.000  98.000
## 76         14  4.000  21.000
## 77         25 17.000  51.000
```

```
## 78          8  1.000  26.000
## 79         31 23.000  64.000
## 80         92 75.000  99.000
## 81         56 43.000  82.000
## 82         66 34.000  83.000
## 83         41 28.000  81.000
## 84         82 53.000  93.000
## 85         57 32.000  78.000
## 86         51 31.000  74.000
## 87         76 54.475  97.000
## 88         49 32.000  70.000
## 89         59 48.000  91.000
## 90         75 63.000  95.000
## 91         69 38.000  86.000
## 92         74 49.000  97.000
## 93         61 50.000  86.000
## 94         39 22.000  64.000
## 95         91 76.000 100.000
## 96         18  5.000  23.000
## 97         98 70.000 100.000
## 98         97 72.000 100.000
## 99          9  2.000  13.000
## 100        83 56.000  92.000
```

```
sum(All_Rank$True_Rank > All_Rank$Lower &
    All_Rank$True_Rank < All_Rank$Upper)/ (length(tst$Time) - tti)
```

```
## [1] 0.97
```

3. Hyper Parameter Learning