BIOS812 HW1

Xiran Wang

2020-02-12

2. 2.4 in text Book The time to death (in days) following a kidney transplant follows a log logistic distribution with $\alpha = 1.5$ and $\lambda = 0.01$.

(a) Find the 50, 100, and 150 day survival probabilities for kidney transplantation in patients.
(b) Find the median time to death following a kidney transplant.
(c) Show that the hazard rate is initially increasing and, then, decreasing over time. Find the time at which the hazard rate changes from increasing to decreasing.
(d) Find the mean time to death

2(a) Let T be the time to death (in days) following a kidney transplant The density function of T:

$$S(t) = P(T > t) = \frac{1}{1+\lambda t^{\alpha}} = \frac{1}{1+0.01t^{1.5}}, \quad t \geq 0$$

Plug 50,100 and 150 in to the function, the survival probabilities are 0.3906, 0.3890 and .3880 respectively.

```
St=function(t, alpha=1.5,lambda=0.01){

  S=1/(1+lambda* t^(alpha) )
  return(S)
}

table=rbind( St(t=50), St(t=100), St(t=150))
rownames(table)=c("t=50","t=100","t=150")
colnames(table)=c("Survival Probability")
kable(table,booktabs=T)
```

	Survival Probability
t=50	0.2204812
t=100	0.0909091
t=150	0.0516231

2(b) Mean time is the time when survival probablity is 0.5. The mean time to death following a kidney transplant is about 21.5 days.

$$S(t) = \frac{1}{1+0.01t^{1.5}} = 0.5, \quad t \geq 0$$
$$\Rightarrow t = (100)^{1/1.5} = 21.5444 \quad days$$

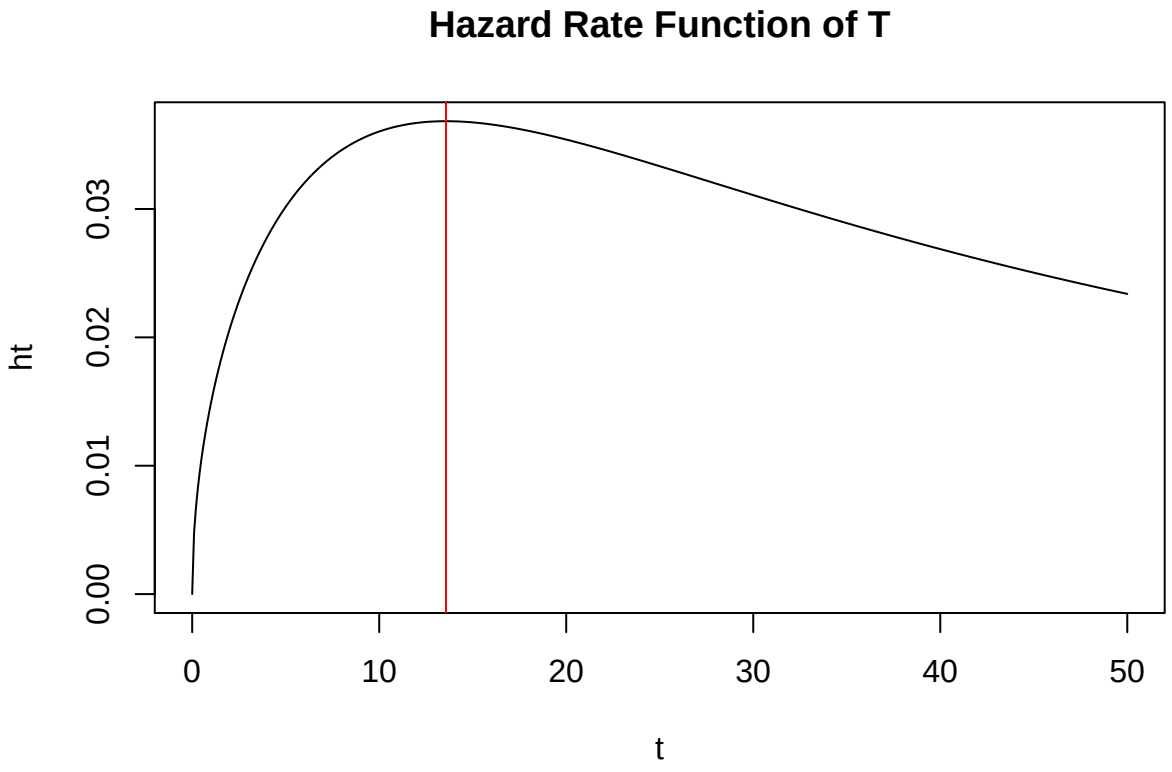
2(c) Hazard rate function for t:

$$h(t) = \frac{\alpha t^{\alpha-1}\lambda}{1+\lambda t^{\alpha}}$$
$$= \frac{0.015t^{0.5}}{1+0.01t^{1.5}}, \quad t \geq 0$$

$$h'(t) = \frac{0.015t^{-0.5}(0.5-0.01t^{1.5})}{(1+0.01t^{1.5})^2} = 0 \Rightarrow t = 13.5720$$

From t=10 to 13.5720, h(t) increases, after that h(t) decreases.

```
t=seq(0,50,.1)
ht = (0.015*t^0.5)/(1+0.01*t^1.5)
plot(t,ht,type="l",main="Hazard Rate Function of T")
abline(v=13.5720,col=2)
```



2(d) Density function of T:

$$
\begin{aligned}
f(t) &= \frac{0.015t^{0.5}}{(1+0.01t^{1.5})^2} \\
E(T) &= \int_{t=0}^{\inf} t * \frac{0.015t^{0.5}}{(1+0.01t^{1.5})^2} dt \\
Let \;\; u &= \frac{1}{(1+0.01t^{1.5})} \\
\Rightarrow x &= (\frac{100}{u} - 100)^{\frac{1}{1.5}}, \quad where \; 0 < u < 1. \\
E(T) &= \int_{u=0}^{1} (\frac{100}{u} - 100)^{\frac{1}{1.5}} du \\
&= 100^{\frac{1}{1.5}} \int_{u=0}^{1} u^{1.5}(1-u)^{\frac{1}{1.5}} du, \quad betakernel \\
&= 52.0934
\end{aligned}
$$

3

```
library(survival)

par(mfrow=c(2,3))
data.generateion <- function(fixed,Lambda,Alpha, c_rate, nn=200){
  lambda <-Lambda
  alpha <- Alpha

    C <- rep(fixed, nn)
    u <- runif(nn, 0,1)
    X <- (-log(u)/lambda)^(1/alpha)
    delta<-as.numeric(X<=C)
    c.rate <- sum(1-delta)/nn


  return(list(delta,C,X,c.rate,fixed)) #return(list(delta,c.rate))
}


# a= 0.1
#data.generateion(fixed=20,Alpha = 0.5,Lambda=0.5)
#data.generateion(fixed=5,Alpha = 1,Lambda=0.5)
#data.generateion(fixed=1,Alpha = 2,Lambda=0.5)

# a= 0.3
#data.generateion(fixed=6,Alpha = 0.5,Lambda=0.5)
#data.generateion(fixed=3,Alpha = 1,Lambda=0.5)
#data.generateion(fixed=2,Alpha = 2,Lambda=0.5)

Alphas <- rep(c(0.5,1,2),2 )
Lambdas <- rep(c(0.5,0.5,0.5),2 )
fixed_cens <- c(20,5,1,6,3,2)
censor_rates <-rep(c(0.1,0.3),c(3,3))
setting <- rbind(Alphas,Lambdas,fixed_cens,censor_rates)
```

```r
rownames(setting) <- c('alpha','lambda','censoring time','censoring rate')
colnames(setting)<-c(paste("Setting", 1:6,sep = ' '))

kable(setting,caption = "Simulation Setting")%>%
  kable_styling(latex_options = "hold_position")
```

Table 1: Simulation Setting

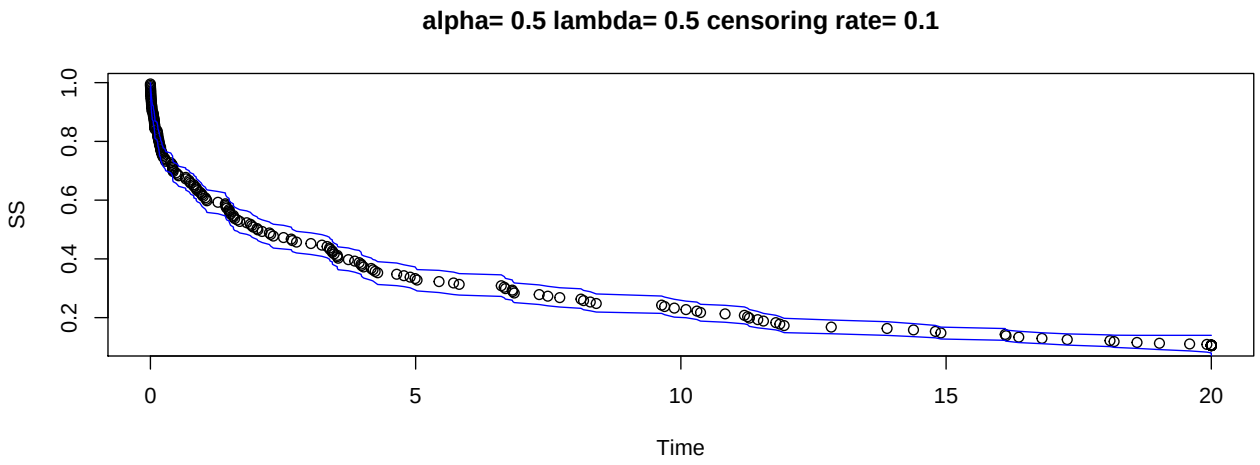
	Setting 1	Setting 2	Setting 3	Setting 4	Setting 5	Setting 6
alpha	0.5	1.0	2.0	0.5	1.0	2.0
lambda	0.5	0.5	0.5	0.5	0.5	0.5
censoring time	20.0	5.0	1.0	6.0	3.0	2.0
censoring rate	0.1	0.1	0.1	0.3	0.3	0.3

```r
get.ests <- function(Fixed,L,A,rep=10,c_rate,nn=200){

  #Fixed=20;L=0.5;A=0.5;rep=10;nn=200;

  estpar<-matrix(NA,nrow=rep,2)
  varla<-matrix(NA,nrow=rep,2)
  bias_alpha <- bias_lambda <- vector()

  ci_alpha <- matrix(NA, nrow=rep,2)
  ci_lambda<- matrix(NA, nrow=rep,2)
  delta <- vector()

  TIME <- rep(list(rep(NA, nn)), rep)

  SS <- matrix(NA,nrow = rep,ncol=nn)

  for (j in 1:rep) {
    out <- data.generateion(fixed=Fixed ,Lambda=L,Alpha =A,nn=200)

    delta <- unlist(out[1])
    C <-  unlist(out[2])
    X <- unlist(out[3])

    time <- delta*X+(1-delta)*C
    TIME[[j]] <- time


    LL <- function(theta){
      X=X; C=C; delta=delta
      time <- delta*X+(1-delta)*C

      alpha <- exp(theta[1])
      lambda <- exp(theta[2])
      -sum(delta*log(alpha)+delta*log(lambda)+delta*(alpha-1)*log(time)-lambda*time^alpha)
    }


    mll<-optim(c(0,0), LL, hessian = TRUE)
```

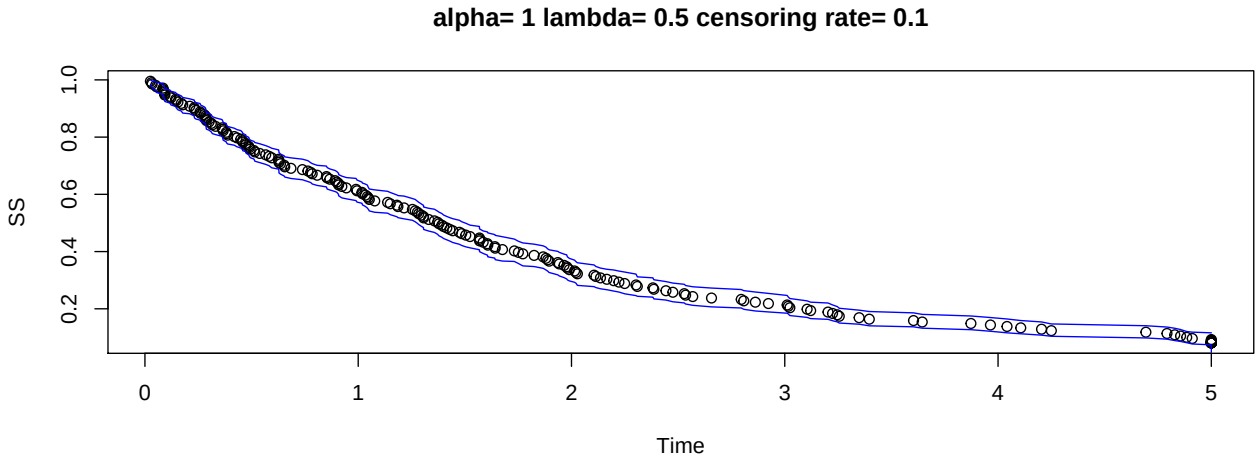
```
  estpar[j,]<-exp(mll$par)
  estvar<-solve(mll$hessian)
  dd<-diag( c(estpar[j,]) )
  varla[j,]<-c(diag( dd%*%estvar%*%t(dd) ))

  ci_alpha[j,] <- c(estpar[j,1]- 1.96 * sqrt( varla[j,1]), estpar[j,1]+ 1.96 * sqrt( varla[j,1]) )
  ci_lambda[j,] <- c(estpar[j,2]- 1.96 * sqrt( varla[j,2]), estpar[j,2]+ 1.96 * sqrt( varla[j,2]) )

  SS[j,] <- exp(- estpar[j,2]*time^(estpar[j,1])) # 500*200 matrix

}

bias_alpha <-  sum(estpar[,1]- A ) /rep
bias_lamba <- sum( estpar[,2]- L ) /rep

# Average standard deviation
StdErr_alpha <- sqrt(  sum( (estpar[,1]-A)^2 ) /rep )
StdErr_lambda <-  sqrt( sum( (estpar[,2]-L)^2 ) /rep  )
StdErr <-c(StdErr_alpha,StdErr_lambda)

# Empirical standard deviation
StDev_alpha <-sd( estpar[,1] )
StDev_lambda <-sd( estpar[,2] )
StDev <- c(StDev_alpha,StDev_lambda)

in_alpha <- ifelse( ci_alpha[,1]<=A & A<=ci_alpha[,2],1,0)
coverage_alpha <- sum( in_alpha)/ length(in_alpha)

in_lam <- ifelse( ci_lambda[,1]<L & L<ci_lambda[,2],1,0)
coverage_lambda <- sum( in_alpha)/ length(in_alpha)


sort_TIME <- matrix(NA, ncol = nn, nrow = rep)
sort_SS <- matrix(NA, ncol = nn, nrow = rep)


for ( k in 1:rep) {
  sort_TIME[k,] <- sort( TIME[[1]] )
  sort_SS[k,]<-  SS[k,][ order( TIME[[k]] )]
}

qtl_low <- (apply(sort_SS, 2, quantile, probs = c(.025),na.rm=T))
qtl_up <- (apply(sort_SS, 2, quantile, probs = c(.975),na.rm=T))
mean_SS <- apply(sort_SS,2,mean)

plt <- plot( sort_TIME[1,],mean_SS,
             main =paste('alpha=',A,'lambda=',L,'censoring rate=',c_rate),xlab = "Time",ylab = "SS")
points(sort_TIME[1,], qtl_low,col="blue", type="l")
points(sort_TIME[1,], qtl_up,col="blue", type="l")

print(plt)

return(list(bias_alpha,bias_lamba, StdErr,StDev,coverage_alpha,coverage_lambda))
```

```
  }
```

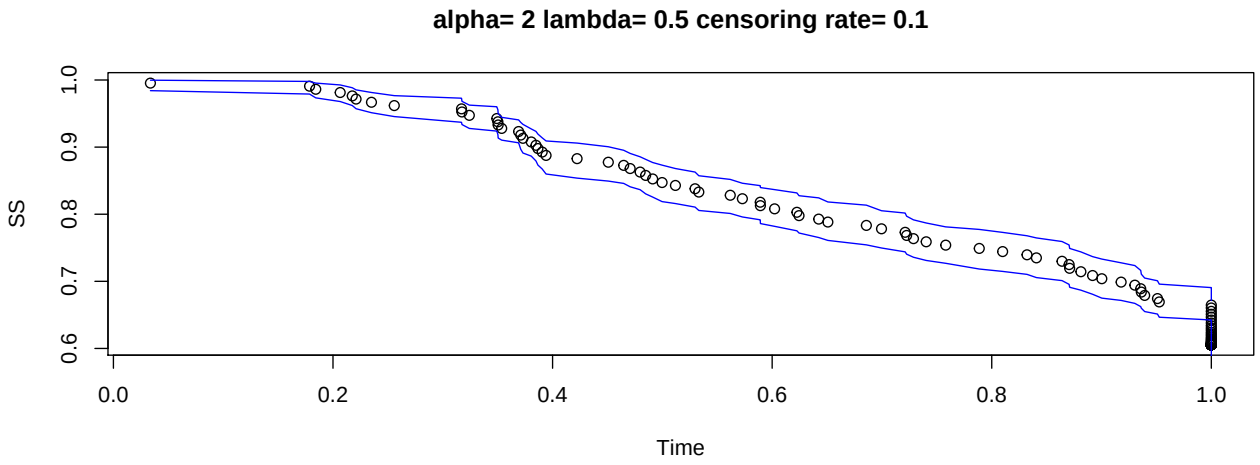
```
#bias_alpha,bias_lamba, StdErr,StdDev,coverage_alpha,coverage_lambda


par(mfrow=c(1,1))
r<- rep(list(rep(NA, 8)), 6)
for (s in 1:6) {
a <- setting["alpha", s]
l <- setting["lambda", s]
fixed <- setting["censoring time", s]
results<- get.ests( Fixed=fixed,L=l,A=a,rep=500,c_rate=0.1,nn=200)
r[[s]] <- unlist(results)

}
```



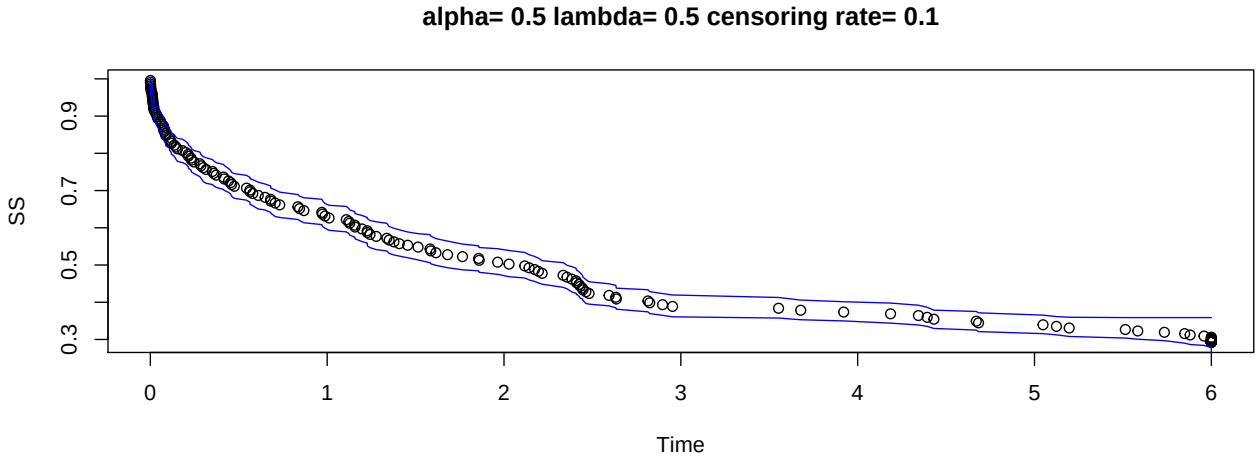

```
## NULL
```



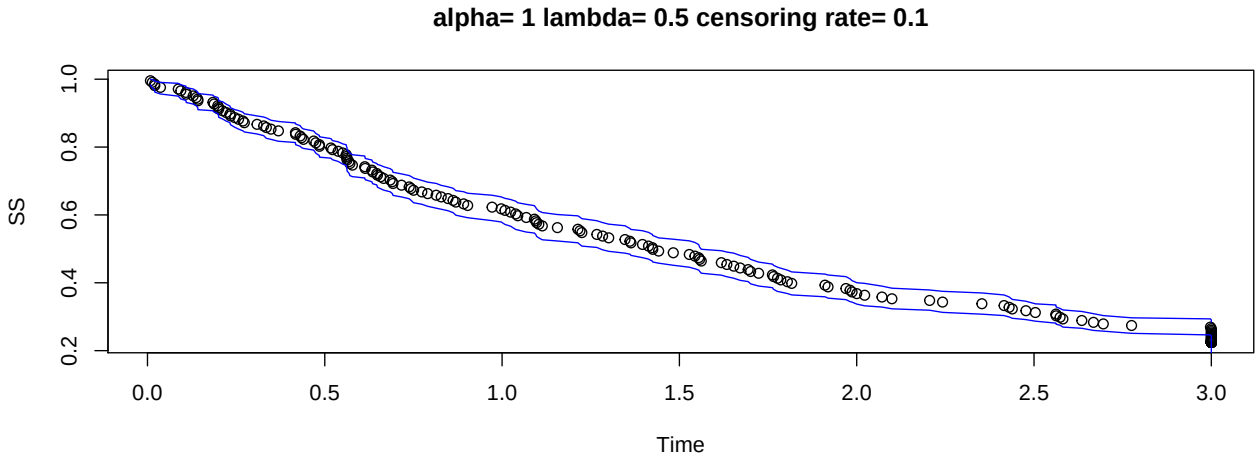

```
## NULL
```



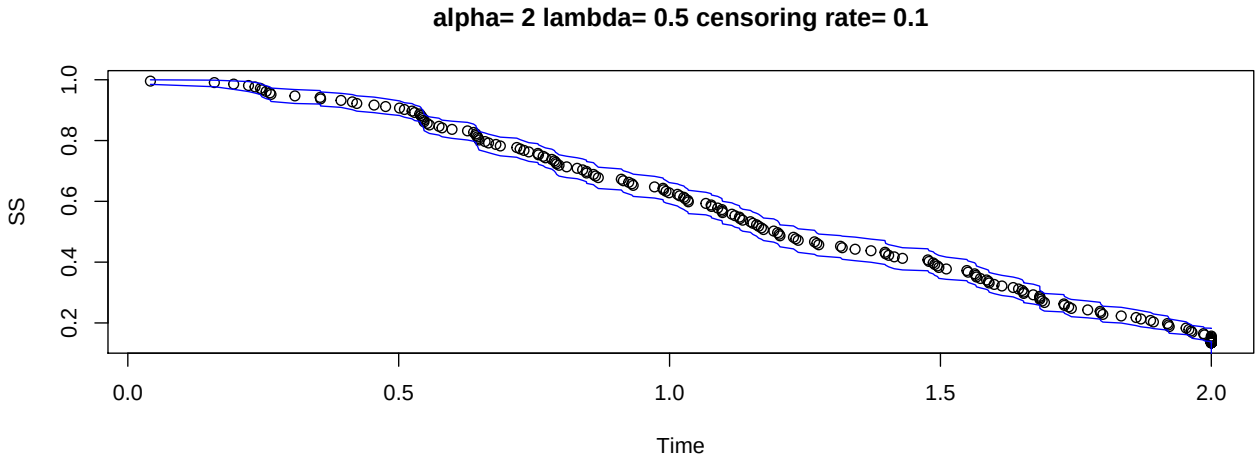

```
## NULL
```




```
## NULL
```




```
## NULL
```




```
## NULL
```

```
result_table <- cbind(r[[1]],r[[2]],r[[3]],r[[4]],r[[5]],r[[6]])
rownames(result_table)<-c('bais alpha','bias lambda','StdErr alpha','StdErr lambda','StDev alpha','StDe

TABLE<-round(rbind( setting, result_table),3)

kable(TABLE,caption = "Simulation Results")%>%
  kable_styling(latex_options = "hold_position")
```

Table 2: Simulation Results

	Setting 1	Setting 2	Setting 3	Setting 4	Setting 5	Setting 6
alpha	0.500	1.000	2.000	0.500	1.000	2.000
lambda	0.500	0.500	0.500	0.500	0.500	0.500
censoring time	20.000	5.000	1.000	6.000	3.000	2.000
censoring rate	0.100	0.100	0.100	0.300	0.300	0.300
bais alpha	0.004	0.007	0.020	0.004	0.004	0.000
bias lambda	0.003	0.002	0.003	0.001	-0.001	0.005
StdErr alpha	0.030	0.063	0.230	0.038	0.064	0.128
StdErr lambda	0.045	0.048	0.059	0.048	0.045	0.048
StDev alpha	0.030	0.063	0.229	0.038	0.064	0.128
StDev lambda	0.045	0.048	0.059	0.048	0.045	0.048
coverage rate alpha	0.970	0.954	0.934	0.946	0.968	0.942
coverage lambda	0.970	0.954	0.934	0.946	0.968	0.942