Final Project

Tori Wang

2023-06-13

```
#Weibell Distribution: It is characterized by its shape parameter k > 0 and its scale parameter .
#Its probability density function is given by
setwd("~/Documents/Stats 102B")
source("bootsample.R")
source("bootstats.R")
k=1.5
lambda = 1
n = c(100,250)
#and 250,

s_size100 = rweibull(n[1],shape=k,scale=lambda)
s_size250 = rweibull(n[2],shape=k,scale=lambda)

# Your results should be based on M = 100, 500 random samples and B = 5000
# bootstrap replicates.

M = c(100,500)
B = 5000
alpha=.05
zval=qnorm(p=alpha/2, lower.tail=FALSE)

xbar = function(w){median(w)}
```

```
#1. Normal bootstrap confidence interval.
set.seed(0)
size_100_CIs = data.frame(matrix(ncol = 2, nrow = 0))
size_250_CIs = data.frame(matrix(ncol = 2, nrow = 0))


set.seed(0)
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size100, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_100_CIs=rbind(size_100_CIs,cbind(mean(s_size100)-zval*boot.x.bar$se,
                                        mean(s_size100)+zval*boot.x.bar$se))
}
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size250, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_250_CIs=rbind(size_250_CIs,cbind(mean(s_size250)-zval*boot.x.bar$se,
                                        mean(s_size250)+zval*boot.x.bar$se))
}
```

```r
#2. Basic bootstrap confidence interval.
set.seed(0)
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size100, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_100_CIs=rbind(size_100_CIs,cbind(2*mean(s_size100)-quantile(boot.x.bar$theta,probs=(1-alpha/2)),
                              2*mean(s_size100)-quantile(boot.x.bar$theta,probs=(alpha/2))))
}
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size250, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_250_CIs=rbind(size_250_CIs,cbind(2*mean(s_size250)-quantile(boot.x.bar$theta,probs=(1-alpha/2)),
                              2*mean(s_size250)-quantile(boot.x.bar$theta,probs=(alpha/2))))
}
```

```r
#3. Percentile bootstrap confidence interval.
set.seed(0)
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size100, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_100_CIs=rbind(size_100_CIs,cbind(quantile(boot.x.bar$theta,probs=(alpha/2)),
                        quantile(boot.x.bar$theta,probs=(1-alpha/2))))
}
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size250, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_250_CIs=rbind(size_250_CIs,cbind(quantile(boot.x.bar$theta,probs=(alpha/2)),
                        quantile(boot.x.bar$theta,probs=(1-alpha/2))))
}
```

```r
#4. Bootstrap-t (studentized) confidence interval.
# I am using a smaller B number of replicates because my computer is not powerful enough
B=1000
set.seed(0)
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size100, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  boot.samples.SE = rep(0,ncol(boot.samples))
  #ncol(boot.samples)
  for (l in 1:ncol(boot.samples)) {
    # obtain bootstrap samples from original bootstrap sample l
    iter.boot.samples = bootsampling(boot.samples[,l],B)
    # get the std dev of the bootstrap sampling distribution based
    # on the l-th bootstrap sampe
    boot.samples.SE[l] = boot.stats(iter.boot.samples,xbar)$se
  }

  T = (boot.x.bar$theta-median(s_size100)) / boot.samples.SE
  qval = quantile(T,probs=c(alpha/2,1-alpha/2))
  size_100_CIs=rbind(size_100_CIs,
                  cbind(median(s_size100)-qval[2]*boot.x.bar$se,
                        median(s_size100)-qval[1]*boot.x.bar$se))
}
```

```
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size250, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  boot.samples.SE = rep(0,ncol(boot.samples))
  #ncol(boot.samples)
  for (l in 1:ncol(boot.samples)) {
    # obtain bootstrap samples from original bootstrap sample l
    iter.boot.samples = bootsampling(boot.samples[,l],B)
    # get the std dev of the bootstrap sampling distribution based
    # on the l-th bootstrap sampe
    boot.samples.SE[l] = boot.stats(iter.boot.samples,xbar)$se
  }

  T = (boot.x.bar$theta-median(s_size250)) / boot.samples.SE
  qval = quantile(T,probs=c(alpha/2,1-alpha/2))
  size_250_CIs=rbind(size_250_CIs,
                  cbind(median(s_size250)-qval[2]*boot.x.bar$se,
                        median(s_size250)-qval[1]*boot.x.bar$se))
}
print(size_100_CIs)
```

```
##                V1        V2
## 1       0.8054017 1.0150623
## 2       0.8327999 0.9876640
## 97.5%   0.8678627 1.0273614
## 97.5%1  0.8724622 1.0348856
## 2.5%    0.7931026 0.9526013
## 2.5%1   0.7855784 0.9480017
## 97.5%2  0.5464201 0.8578118
## 97.5%3  0.7663647 0.8827607
```

```
print(size_250_CIs)
```

```
##                V1        V2
## 1       0.7011552 1.0223163
## 2       0.7997993 0.9236721
## 97.5%   0.8895917 1.1963081
## 97.5%1  0.9448859 1.0713772
## 2.5%    0.5271634 0.8338798
## 2.5%1   0.6520943 0.7785855
## 97.5%2  0.6964447 0.9241893
## 97.5%3  0.6384580 0.8119315
```

Part b)

```
k=1.5
lambda = 1
n = c(100,250)
#and 250,

s_size100 = rweibull(n[1],shape=k,scale=lambda)
s_size250 = rweibull(n[2],shape=k,scale=lambda)

# Your results should be based on M = 100, 500 random samples and B = 5000
# bootstrap replicates.
```

```r
M = c(100,500)
B = 5000
alpha=.05
zval=qnorm(p=alpha/2, lower.tail=FALSE)

xbar = function(w){sd(w)}
```

```r
#1. Normal bootstrap confidence interval.
set.seed(0)
size_100_CIs = data.frame(matrix(ncol = 2, nrow = 0))
size_250_CIs = data.frame(matrix(ncol = 2, nrow = 0))


set.seed(0)
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size100, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_100_CIs=rbind(size_100_CIs,cbind(mean(s_size100)-zval*boot.x.bar$se,
                                        mean(s_size100)+zval*boot.x.bar$se))
}
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size250, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_250_CIs=rbind(size_250_CIs,cbind(mean(s_size250)-zval*boot.x.bar$se,
                                        mean(s_size250)+zval*boot.x.bar$se))
}
```

```r
#2. Basic bootstrap confidence interval.
set.seed(0)
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size100, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_100_CIs=rbind(size_100_CIs,cbind(2*mean(s_size100)-quantile(boot.x.bar$theta,probs=(1-alpha/2)),
                                2*mean(s_size100)-quantile(boot.x.bar$theta,probs=(alpha/2))))
}
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size250, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_250_CIs=rbind(size_250_CIs,cbind(2*mean(s_size250)-quantile(boot.x.bar$theta,probs=(1-alpha/2)),
                                2*mean(s_size250)-quantile(boot.x.bar$theta,probs=(alpha/2))))
}
```

```r
#3. Percentile bootstrap confidence interval.
set.seed(0)
set.seed(0)
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size100, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  size_100_CIs=rbind(size_100_CIs,cbind(quantile(boot.x.bar$theta,probs=(alpha/2)),
                          quantile(boot.x.bar$theta,probs=(1-alpha/2))))
}
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size250, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
```

```
  size_250_CIs=rbind(size_250_CIs,cbind(quantile(boot.x.bar$theta,probs=(alpha/2)),
                        quantile(boot.x.bar$theta,probs=(1-alpha/2))))
}
```

```
#4. Bootstrap-t (studentized) confidence interval.
# I am using a smaller B number of replicates because my computer is not powerful enough
B=1000
set.seed(0)
for(m in 1:2){
  boot.samples = bootsampling(sample(s_size100, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  boot.samples.SE = rep(0,ncol(boot.samples))
  #ncol(boot.samples)
  for (l in 1:ncol(boot.samples)) {
    # obtain bootstrap samples from original bootstrap sample l
    iter.boot.samples = bootsampling(boot.samples[,l],B)
    # get the std dev of the bootstrap sampling distribution based
    # on the l-th bootstrap sampe
    boot.samples.SE[l] = boot.stats(iter.boot.samples,xbar)$se
  }

  T = (boot.x.bar$theta-median(s_size100)) / boot.samples.SE
  qval = quantile(T,probs=c(alpha/2,1-alpha/2))
  size_100_CIs=rbind(size_100_CIs,
                  cbind(median(s_size100)-qval[2]*boot.x.bar$se,
                        median(s_size100)-qval[1]*boot.x.bar$se))
}

for(m in 1:2){
  boot.samples = bootsampling(sample(s_size250, M[m], replace = TRUE),B)
  boot.x.bar = boot.stats(boot.samples,xbar)
  boot.samples.SE = rep(0,ncol(boot.samples))
  #ncol(boot.samples)
  for (l in 1:ncol(boot.samples)) {
    # obtain bootstrap samples from original bootstrap sample l
    iter.boot.samples = bootsampling(boot.samples[,l],B)
    # get the std dev of the bootstrap sampling distribution based
    # on the l-th bootstrap sampe
    boot.samples.SE[l] = boot.stats(iter.boot.samples,xbar)$se
  }

  T = (boot.x.bar$theta-median(s_size250)) / boot.samples.SE
  qval = quantile(T,probs=c(alpha/2,1-alpha/2))
  size_250_CIs=rbind(size_250_CIs,
                  cbind(median(s_size250)-qval[2]*boot.x.bar$se,
                        median(s_size250)-qval[1]*boot.x.bar$se))
}
print(size_100_CIs)
```

```
##               V1        V2
## 1      0.7854656 1.1102895
## 2      0.8837174 1.0120377
## 97.5%  1.0589112 1.3746853
## 97.5%1 1.1761794 1.3061626
```

```
## 2.5%   0.5210698 0.8368439
## 2.5%1  0.5895925 0.7195757
## 97.5%2 0.6719327 1.0756572
## 97.5%3 0.7704209 0.8941435
```

```
print(size_250_CIs)
```

```
##               V1        V2
## 1      0.7956586 0.9943909
## 2      0.8371067 0.9529429
## 97.5%  1.0507879 1.2445047
## 97.5%1 1.0735395 1.1901595
## 2.5%   0.5455449 0.7392617
## 2.5%1  0.5998900 0.7165100
## 97.5%2 0.8644816 1.4028386
## 97.5%3 0.8876866 1.0232547
```

Problem 2