STP598-Assignment 2

Hao Wang
hwang306

September 28, 2017

1 Question 1

```
library(boot)
mydata <- aircondit
```

let $t = hours$, then t is geq 0. the pdf of t is

$$f(t;\lambda) = \lambda e^{-\lambda t}$$

Assume independent, the joint pdf for t_i from 1 to n is

$$f(\mathbf{t};\lambda) = \lambda^n e^{-\lambda \sum t_i}$$

Take log transformation.

$$ln(f(\mathbf{t})) = nln(\lambda) - \lambda \sum t_i$$

take derivative with respect to λ

$$\frac{\partial\, ln}{\partial \lambda} = \frac{n}{\lambda} - \sum t_i$$

the MLE of lambda is the value of lambda when this equation is 0. Thus

$$\lambda = \frac{n}{\sum t_i}$$

To get this estimate in r

```
lambda <- nrow(mydata) / sum(mydata$hours)
lambda
```

```
## [1] 0.00925212
```

1.1 use loop for se and bias

```
# set up the bootstrap
B <- 10000        #number of replicates
n <- nrow(mydata) #sample size
R <- numeric(B)   #storage for replicates

#bootstrap method using loop
for (b in 1:B) {
  #randomly select the indices
  i <- sample(1:n, size = n, replace = TRUE)
  HOUR <- mydata$hours[i]
  R[b] <- n / sum(HOUR) #this is the lambda scores from loop
}

#output
se.R <- sd(R)
se.R
```

```
## [1] 0.004307683
```

```
bias.R <- mean(R) - lambda
bias.R
```

```
## [1] 0.001296279
```

1.2 use boot funtion

```
# write a funtion for estimate lambda in bootstrap
bs <- function(data, indices) {
  d <- data[indices,] # allows boot to select sample
  L <- nrow(data)/sum(d)
  return(L)
}

set.seed(99)
results <- boot(data = mydata, statistic = bs, R = 10000)
results
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
```

```
## boot(data = mydata, statistic = bs, R = 10000)
##
##
## Bootstrap Statistics :
##       original      bias    std. error
## t1* 0.00925212 0.001415736  0.00438065
```

1.3 use replicates function

```
n <- nrow(mydata) #sample size
results <- replicate(10000,
                     expr = {
                       y <- sample(1:n, size = n, replace = TRUE)
                     nrow(mydata)/sum(y)})
bias.R <- mean(results) - lambda
bias.R
```

```
## [1] 0.148496
```

```
se.R <- sd(results)
se.R
```

```
## [1] 0.02614721
```

2 Question 2

2.1 Part I question

This question requires a comparison between regular intervals and bootstrap intervals.

I examine the four models in part one first

```
mydata <- read.csv("https://raw.githubusercontent.com/haowang666/Computational-Stats/mas

#model 1 lm
lm1 <- lm(HtVol ~ Male + Age + Ht + Wt, data = mydata)
confint(lm1, level = 0.9)
```

```
##                       5 %         95 %
## (Intercept) -414.8012233 -112.5317827
## Male           2.5779840   80.1009282
## Age           -0.9730374    0.2983624
## Ht             1.7568628    5.0089676
## Wt             4.9516099    7.2266185
```

```r
#model 2 lad
library("quantreg")
lad1  <- quantreg::rq(HtVol ~ Male + Age + Ht + Wt, data = mydata, alpha = 0.05, ci = TR
summary(lad1)
```

```
##
## Call: quantreg::rq(formula = HtVol ~ Male + Age + Ht + Wt, data = mydata,
##     alpha = 0.05, ci = TRUE)
##
## tau: [1] 0.5
##
## Coefficients:
##             coefficients lower bd   upper bd
## (Intercept) -213.76087   -407.11529 -188.29687
## Male          48.16617     21.33280   71.79785
## Age            0.18341     -1.85850    0.35471
## Ht             2.99404      2.76416    5.10450
## Wt             4.41326      2.85604    8.74488
```

```r
# model 3 lm
lm2 <- lm(HtVol ~ Male + Age + BMI + BSA, data = mydata)
confint(lm2, level = 0.9)
```

```
##                      5 %          95 %
## (Intercept) -191.913547 -51.977661214
## Male          -3.238045  77.253468560
## Age           -1.337623  -0.009833138
## BMI          -10.254832  -0.343960509
## BSA          467.483166 713.717291493
```

```r
# model 4 lad
lad <- quantreg::rq(HtVol ~ Male + Age + BMI + BSA, data = mydata, alpha = 0.05, ci = TR
summary(lad)
```

```
##
## Call: quantreg::rq(formula = HtVol ~ Male + Age + BMI + BSA, data = mydata,
##     alpha = 0.05, ci = TRUE)
##
## tau: [1] 0.5
##
## Coefficients:
##             coefficients lower bd   upper bd
## (Intercept)  -40.98998   -104.92448   -6.20440
## Male          37.87068     -4.40723   59.38167
## Age           -0.22338     -1.66704    0.29381
## BMI           -6.65581     -8.43418   -2.10408
```

```
## BSA          501.57188    418.43906  611.75467
```

We can use bootstrap to compute confidence intervals for the four models.

- **for model 1**

```
library(boot)
# Bootstrap 90% CI for regression coefficients

# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data = d)
  return(coef(fit))
}

# bootstrapping with 1000 replications
set.seed(99)
results <- boot(data = mydata, statistic = bs,
    R = 1000, formula = HtVol ~ Male + Age + Ht + Wt)


# get 95% confidence intervals
boot.ci(results, conf = 0.9, type = "bca", index = 1) # intercept
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 90%   (-384, -107 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 2) # Male
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 2)
##
## Intervals :
## Level       BCa
## 90%   ( 8.83, 78.96 )
```

```
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 3) # Age
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 3)
##
## Intervals :
## Level       BCa
## 90%   (-1.5264,  0.2792 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 4) # Ht
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 4)
##
## Intervals :
## Level       BCa
## 90%   ( 1.752,  4.683 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 5) # Wt
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 5)
##
## Intervals :
## Level       BCa
## 90%   ( 3.951,  7.978 )
## Calculations and Intervals on Original Scale
```

The steps for the other three models are very similar

- **for model 2:**

```
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
```

```
  fit <- quantreg::rq(formula, data = d)
  return(coef(fit))
}

# bootstrapping with 1000 replications
set.seed(99)
results <- boot(data = mydata, statistic = bs,
    R = 1000, formula = HtVol ~ Male + Age + Ht + Wt)


# get 95% confidence intervals
boot.ci(results, conf = 0.9, type = "bca", index = 1) # intercept
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 90%   (-312.4,  136.7 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 2) # Male
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 2)
##
## Intervals :
## Level       BCa
## 90%   (  4.69, 104.41 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 3) # Age
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 3)
##
```

```
## Intervals :
## Level       BCa
## 90%   (-0.5163,  1.2208 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 4) # Ht
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 4)
##
## Intervals :
## Level       BCa
## 90%   (-2.376,  4.219 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 5) # Wt
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 5)
##
## Intervals :
## Level       BCa
## 90%   ( 2.895,  8.755 )
## Calculations and Intervals on Original Scale
```

- **for model 3:**

```
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data = d)
  return(coef(fit))
}

# bootstrapping with 1000 replications
set.seed(99)
results <- boot(data = mydata, statistic = bs,
    R = 1000, formula = HtVol ~ Male + Age + BMI + BSA)
```

```r
# get 95% confidence intervals
boot.ci(results, conf = 0.9, type = "bca", index = 1) # intercept
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 90%   (-237.3,  -48.5 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```r
boot.ci(results, conf = 0.9, type = "bca", index = 2) # Male
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 2)
##
## Intervals :
## Level       BCa
## 90%   ( 2.66, 78.58 )
## Calculations and Intervals on Original Scale
```

```r
boot.ci(results, conf = 0.9, type = "bca", index = 3) # Age
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 3)
##
## Intervals :
## Level       BCa
## 90%   (-2.1974,  0.0238 )
## Calculations and Intervals on Original Scale
```

```r
boot.ci(results, conf = 0.9, type = "bca", index = 4) # BMI
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 4)
##
## Intervals :
## Level       BCa
## 90%   (-9.317,  3.149 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 5) # BSA
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 5)
##
## Intervals :
## Level       BCa
## 90%   (468.2, 808.9 )
## Calculations and Intervals on Original Scale
```

- **for model 4**

```
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- quantreg::rq(formula, data = d)
  return(coef(fit))
}

# bootstrapping with 1000 replications
set.seed(99)
results <- boot(data = mydata, statistic = bs,
    R = 1000, formula = HtVol ~ Male + Age + BMI + BSA)


# get 95% confidence intervals
boot.ci(results, conf = 0.9, type = "bca", index = 1) # intercept
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 1)
##
## Intervals :
```

```
## Level       BCa
## 90%   (-123.11,   15.74 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 2) # Male
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 2)
##
## Intervals :
## Level       BCa
## 90%   (-4.29, 91.29 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 3) # Age
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 3)
##
## Intervals :
## Level       BCa
## 90%   (-2.8375,  0.2543 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 4) # BMI
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 4)
##
## Intervals :
## Level       BCa
## 90%   (-11.999,  -1.306 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 5) # BSA
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
```

```
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 5)
##
## Intervals :
## Level       BCa
## 90%   (391.2, 874.7 )
## Calculations and Intervals on Original Scale
```

Compare with the confidence intervals of `lm` and `rq`, they are slifghtly different as `boot.ci` function use the bootstrap resampling method.

2.2 Part 2 question

I pick `Smoking`, `Age`, `Gender`, `Diabetes` as my predictors.

```
mydata <- read.csv("https://raw.githubusercontent.com/haowang666/Computational-Stats/mas
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
mydata <- mydata %>%
  select(Smoking, Age, Gender, Diabetes, Death)

# Scale Age variable to make it in the similar range
mydata2 <- mydata %>%
  mutate_each_(funs(scale(.) %>% as.vector),
                            vars = c("Age"))

glm.fit <- glm(Death ~ Smoking + Age + Gender + Diabetes, data = mydata2, family = "bino
confint(glm.fit, level = 0.9)
```

```
## Waiting for profiling to be done...
```

```
##                    5 %       95 %
## (Intercept) -2.6141260 -1.1934587
## Smoking     -1.4021651  0.4429234
## Age         -0.1242686  0.7034916
```

```
## Gender      -1.2148137  0.4653185
## Diabetes    -1.9503506  0.7141419
```

The bootstrap confidence intervals can be done in a similar way

```
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- glm(formula, data = d, family = "binomial")
  return(coef(fit))
}

# bootstrapping with 1000 replications
set.seed(99)
results <- boot(data = mydata2, statistic = bs,
    R = 1000, formula = Death ~ Smoking + Age + Gender + Diabetes)


# get 95% confidence intervals
boot.ci(results, conf = 0.9, type = "bca", index = 1) # intercept
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 90%   (-2.697, -1.142 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 2) # Smoking
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 2)
##
## Intervals :
## Level       BCa
## 90%   (-1.5797,  0.3523 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 3) # Age
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 3)
##
## Intervals :
## Level       BCa
## 90%   (-0.0361,  0.6718 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 4) # Gender
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 4)
##
## Intervals :
## Level       BCa
## 90%   (-1.2498,  0.5724 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, conf = 0.9, type = "bca", index = 5) # Diabetes
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.9, type = "bca", index = 5)
##
## Intervals :
## Level       BCa
## 90%   (-16.2837,   0.8063 )
## Calculations and Intervals on Original Scale
```

3 Question 3

Permuation test

```
#generate a random dataset including x and y variables

set.seed(1)
x <- rnorm(100) + 100
```

```r
y <- rchisq(100, 7)

#spearman correlation
cor.0 <- cor(x, y, method = "spearman")
cor.test(x, y, method = "spearman")
```

```
##
##  Spearman's rank correlation rho
##
## data:  x and y
## S = 153240, p-value = 0.4255
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.08046805
```

```r
#run a permutation test for 10,000 times
#number of permutations
R <- 10000
reps <- numeric(R)
#create a long vector first
z <- c(x,y) #pooled sample
K <- length(z)
for (i in 1:R) {
#generate indicies k for the first sample
k <- sample(K, size = 100, replace = FALSE)
x1 <- z[k]
y1 <- z[-k] #the rest
reps[i] <- cor(x1, y1, method = "spearman") #spearman test statistics
}

#get empirical p vale
p <- mean(c(cor.0, reps) >= cor.0)
p
```

```
## [1] 0.2122788
```

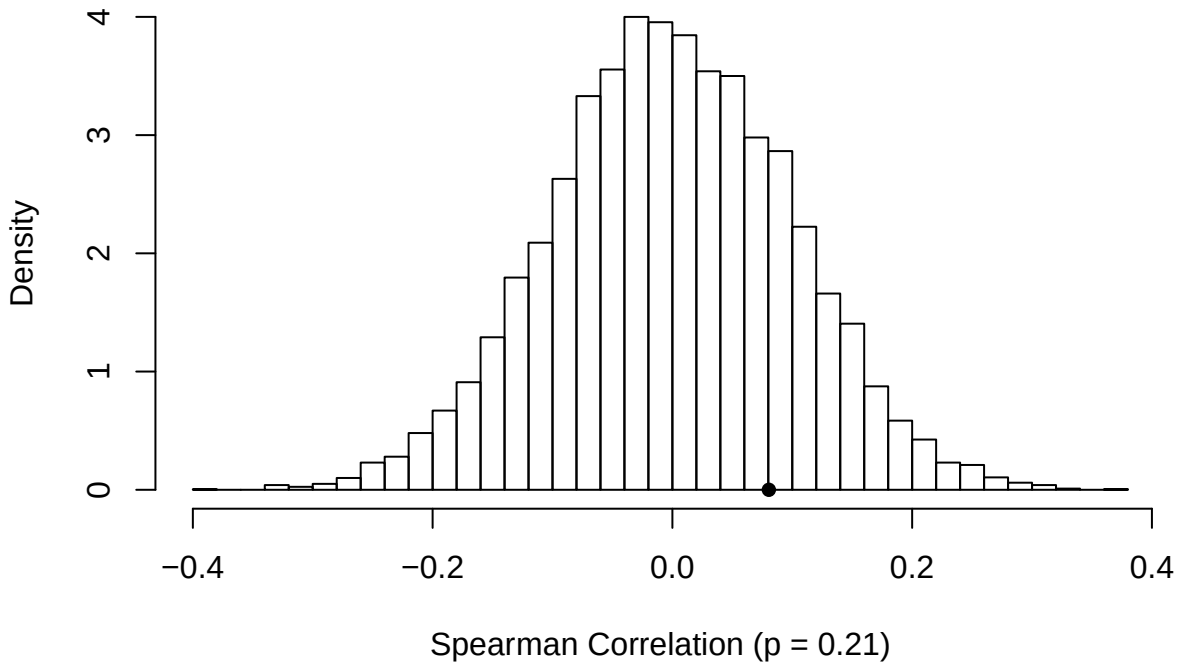
The empirical test value $\hat{p}$ can be obtained by

$$\hat{p} = \frac{\{1 + \sum_{b=1}^{B} I(\hat{\theta}^{(b)}) \geq \hat{\theta}\}}{B+1}$$

In this equation, B is the number of permutations, $\hat{theta}$ is the test statistics. In my case the test statistics is the spearman correlation test value. The p value I got is 0.2122788.

And we can get a histogram of the spearman statistics

```r
hist(reps, main = "", freq = FALSE, xlab = "Spearman Correlation (p = 0.21)", breaks = "
points(cor.0, 0, cex = 1, pch = 16) #observed T
```



Thus we cannot reject the null: true spearman correlation is 0. In the orginal spearman test, the p value is 0.422.