STAT 2450 Assignment 4 (30 points)

Alice Liu

Banner: B00783546

Hints

Please review the lectures on t-confidence intervals and on the bootstrap.

I recall that to simulate 20 observations from a normal distribution with mean 4, standard deviation .75, you can use:

```
set.seed(87654321)
data=rnorm(20,mean=4,sd=.75)
```

and to calculate a 95% t-confidence interval for the mean of 'data', you can use:

```
t.test(data,conf.level=.95)
```

```
##
##  One Sample t-test
##
## data:  data
## t = 27.895, df = 19, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  3.877812 4.506945
## sample estimates:
## mean of x
##  4.192378
```

1.

Simulate $n = 50$ observations from a normal distribution with mean 3, standard deviation .80, using:

ANSWER:

```
set.seed(87612345)
mydata=rnorm(50,mean=3,sd=.80)
# add your own R code below this line
```

Calculate a 98% t-confidence interval for the mean of 'data'

ANSWER:

```r
# add your own R code below this line
t.test(mydata,conf.level=.98)
```

```
##
##  One Sample t-test
##
## data:  mydata
## t = 27.343, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 98 percent confidence interval:
##  2.682230 3.199539
## sample estimates:
## mean of x
##  2.940885
```

Carry out the following bootstrap sampling scheme. Draw $B = 2000$ bootstrap samples. For each sample, calculate the sample average, using the R built-in function "mean".

You can use a for loop to do this, where the loop index i runs from 1 to B.
Inside the loop you can do the bootstrap sampling with "bdata=sample(data,n,replace=T)", where n is the number of sample in your dataset. and then you need something similar to:

"bootstrapmeans=c(bootstrapmeans,mean(bdata))".

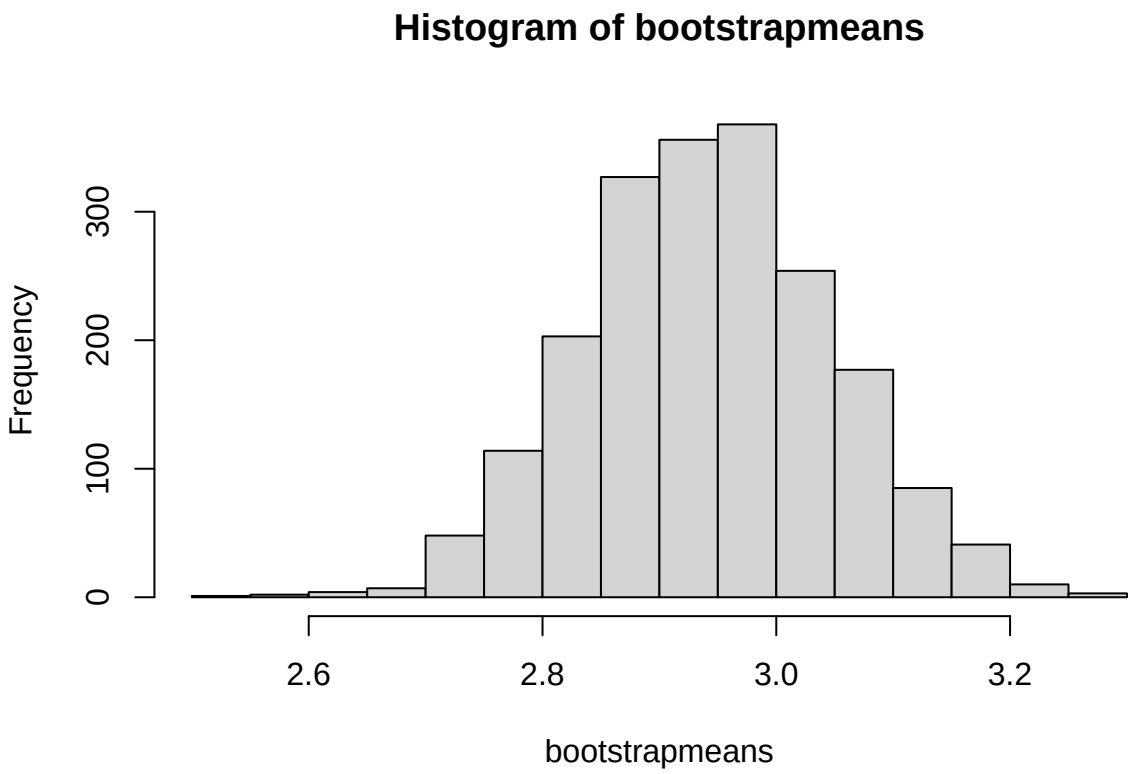
ANSWER:

```r
set.seed(87612345)
# add your own R code below this line
bootstrapmeans=NULL
# use a variable called B for the number of bootstrap sample (2000)
B=2000
# use a variable called n for the number of sample
n=length(mydata)
for(i in 1:B){
   bdata=sample(mydata,n,replace=T)
   bootstrapmeans=c(bootstrapmeans,mean(bdata))
}
```

- (1a) make a histrogram of the bootstrapmeans with 20 bars, using something like:

"hist(bootstrapmeans,nclass=20)". ANSWER:

```r
# add your own R code below this line
hist(bootstrapmeans,nclass=20)
```



Does the histogram look like it has a normal shape?

ANSWER: Yes, this histogram has a normal shape.

- (1b) calculate the 1'th and 99'th percentiles of the bootstrapped means using

"quantile(bootstrapmeans,probs=c(.01,.99))".

ANSWER:

```
# add your own R code below this line
quantile(bootstrapmeans,probs=c(.01,.99))
```

```
##       1%      99%
## 2.714163 3.181877
```

This is a valid 98% bootstrap confidence interval for the unknown population mean, which does not depend on the assumption of normality.
This interval uses the so-called "bootstrap percentile method".

(7 points: 5 points for histogram, 2 points for percentiles)

2.

Repeat problem 1, but using a sample of size $n = 60$ from a chi-square distribution with 0.2 degree of freedom (note that this does not need to be an integer). Use $B = 1500$ bootstrap samples.

You will need first to generate the data, and then to write a loop for producing bootstrap samples from this data.

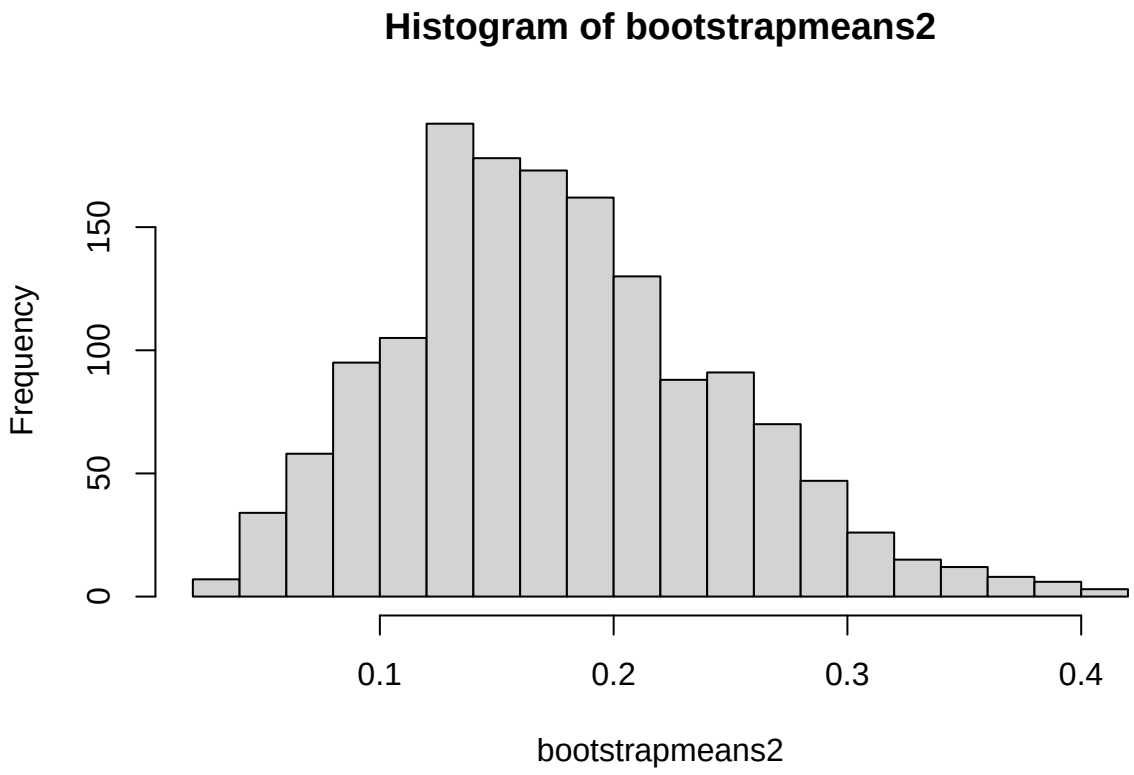
ANSWER:

```
set.seed(117426)
# add your own R code below this line
mydata2=rchisq(60,0.2)
```

(2a) make the histogram of the bootstrap means with 25 bars.

ANSWER:

```
# add your own R code below this line
bootstrapmeans2=NULL
B2=1500
n2=length(mydata2)
for(i in 1:B2){
   bdata2=sample(mydata2,n2,replace=T)
   bootstrapmeans2=c(bootstrapmeans2,mean(bdata2))
}
hist(bootstrapmeans2,nclass=25)
```



Does the histogram look a close to normal as it did when you sampled from a normal distribution?

ANSWER: No, this histogram doesn't look as normal shape. Obviously, this hitogram is right-skewed.

(5 points for histogram)

(2b) make a 96% confidence interval for the population mean using the percentile method

ANSWER:

```
# add your own R code below this line
quantile(bootstrapmeans2,prob=c(0.02,0.98))
```

```
##         2%        98%
## 0.05588592 0.33502848
```

(2 points for the interval)

(2c) calculate the usual 96% t-interval assuming normality, using the t-test procedure.

ANSWER:

```
# add your own R code below this line
t.test(mydata2,conf.level = .96)
```

```
##
##  One Sample t-test
##
## data:  mydata2
## t = 2.4841, df = 59, p-value = 0.01585
## alternative hypothesis: true mean is not equal to 0
## 96 percent confidence interval:
##  0.02695023 0.32177796
## sample estimates:
## mean of x
## 0.1743641
```

Compare the percentile (bootstrap method) confidence interval obtained in 2b and and t-confidence intervals obtained in 2c. Do you expect the two methods to give similar results or not for this distribution (which is not normal)?

ANSWER:No, I do not expect to get similar results.

(2 points for extracting the confidence interval from the t.test output, or for calculating using first principles.)

3.

The following code draws a sample of size $n = 30$ from a chi-square distribution with 15 degrees of freedom, and then puts $B = 200$ bootstrap samples into a matrix M.

After that, the 'apply' function is used to calculate the median of each bootstrap sample, giving a bootstrap sample of 200 medians.

```
set.seed(117888)
data=rchisq(30,15)
M=matrix(rep(0,30*200),byrow=T,ncol=30)
for (i in 1:200)M[i,]=sample(data,30,replace=T)
bootstrapmedians=apply(M,1,median)
```

(3a) Use the 'var' command to calculate the variance of the bootstrapped medians. (This provides a valid estimate of the variance of the median of a random variable.)

ANSWER:

```
# add your own R code below this line
var(bootstrapmedians)
```

```
## [1] 1.138574
```

(1 point for variance)

(3b) Use the apply command to calculate bootstrapped means rather than bootstrapped medians, and then use 'var' to calculate an estimate of the variance of the mean (hint: use 'apply' on the 'mean') .

ANSWER:

```
# add your own R code below this line
bootstrapmeans=apply(M,1,mean)
var(bootstrapmeans)
```

```
## [1] 1.029961
```

(3 points for the variance of the bootstrapped means. Note: if you run the sampling again, yournumerical result may change. This will not affect your mark if your code is correct).

(3c) The theoretical variance of a chi-square random variable with 15 degrees of freedom is 2(15)=30, so the theoretical variance of the mean is 30/30=1

To show this we have applied the formula:

$$Var(\bar{x}) = Var(X)/n$$

where n is the number of items that are averaged. So we know that the true variance of the mean is 1. If we had an infinite number of samples, the estimated variance of the mean would be 1.

But we only have $n = 30$ samples. Did the bootstrap do a reasonable job of estimating the variance of the mean?

ANSWER: Yes, the results are close. Since we know that the estimated variance of the mean is 1 from the given formula and the variance of the bootstrapped we calculated means in sample is 1.258912 which near 1.

4.

Following is a recursive function which takes as input (argument) a vector x and returns a scalar. Either by looking carefully at the function, or by running the function with a few simple input vectors and observing the returned values in each case, determine what the function is doing.

```
myfun=function(x){
if(length(x)==1){return(x)
} else
return(x[1]+myfun(x[-1]))
}
myfun(c(1:1000)) #here's what happens when it is applied to 1:1000
```

```
## [1] 500500
```

ANSWER (4 points) (Describe in precise words what the function does) This function is to calculate the sum of all elements inside x.

Modify the function slightly so that it returns the value:

$\sum_{i=1}^{n} e^{x_i}$

In other words, we want to write a function that calculates the sum of the exponential of each component of the input vector x.

Test your function with the argument vector: x=c(2,1,3,0.4). You should get an answer approximately equal to 31.6847

ANSWER (6 points):

```r
# add your own R code below this line
myfun2=function(x){
   if(length(x)==1){
      return(exp(x))
   }
   else{
      return(exp(x[1])+myfun2(x[-1]))
   }
}
myfun2(c(2,1,3,0.4))
```

```
## [1] 31.6847
```