Mark S
Working Alone (assignment 4)
2/12/2022
STAT486

Summary/Analysis of Assignment 4

For assignment 4 we were expected to create two functions. The first function's is the bootstrap function which computes a bootstrap on any given data set. This was like our creation as the last assignment. The principle of the Empirical Bootstrap is to perform computations on data itself to estimate variance of the generated statistics computed from the data. The bootstrap technique is incredibly insightful as it gives us an estimate on the extent to which our point estimates may vary. Bootstrap has a critical benefit which is that it allows statisticians and data scientists to set confidence intervals on parameters without making unreasonable assumptions. In this assignment the bootstrap function takes in the data, a nboot value, and an alpha level. We use a nboot value of 10,000 which determines the amount of time the bootstrap will run. For the alpha level we use two values 0.1 and 0.05. These values have significant impact that we will discuss after mentioning what the second function does.

Our second function was simulation. This function takes in population mean, the sample size, and number of simulations to run on the bootstrap function. For this assignment we set the population mean to a set value of 3 and the number of times the simulation will run to 1,000. The purpose of our second function is to perform simulations on the bootstrap for our data, this will allow us to determine the confidence intervals for different sample sizes. More specifically this will allow us to compare the coverage of the bootstrap confidence interval over the mean to the coverage of the central limit theorems confidence interval. One thing I noticed was that for small sample sizes of N = 3 for example we got extraordinary results that was unexpected although a bit expected. One solution we could do to fix this problem is by getting rid of the entirety of all the points at which NAN and all zeros appeared. Not something I am too familiar with. However, this is also a general issue with data sample of such low value. And bootstrap typically would not work with such small data sample.

We did the first analysis by testing out a few different things. The first thing we want to find out is how the coverage is for a confidence interval of 95% which can be obtained by setting the alpha value to 0.05. We then run the code multiple times, with a different sample size each time (n=3,10,30,100). This will tell us how ideal the coverage of the bootstrap confidence interval is compared to the central limit theorem confidence interval. If the confidence Interval is closer to .95 it will be better.

Type	**Alpha Level**	**N = 3**	**N = 10**	**N = 30**	**N = 100**
Boot Confidence Interval	**0.05**	**1**	**0.913**	**0.917**	**0.934**
Central Limit Interval	**0.05**	**0.696**	**0.813**	**0.854**	**0.909**

As we can see in this case the Bootstrap Confidence Interval provides much better coverage, but the Central Limit Interval gets closer as the sample size increase

Mark S
Working Alone (assignment 4)
2/12/2022
STAT486

The same testing will be applied when set the alpha level to 0.1. We will run the code multiple times (8 times) with a different sample size each time (n=3,10,30,100). This will tell us how ideal the coverage of the bootstrap confidence interval is compared to the central limit theorem confidence interval. If the confidence Interval is closer to .90 it will be better.

Type	**Alpha Level**	**N = 3**	**N = 10**	**N = 30**	**N = 100**
Boot Confidence Interval	**0.1**	**0.836**	**0.851**	**0.876**	**0.91**
Central Limit Interval	**0.1**	**0.622**	**0.761**	**0.817**	**0.886**

If we look at the following table as seen above. Apart from some numerical errors when we set n= 3 since we didn't remove all the NAN & zeros. We can clearly see that in most cases as well the Bootstrap Confidence Interval provides much better coverage, but the Central Limit Interval gets closer as the sample size increases.

Below contains our code used to receive the results in the tables above.

```r
#ASSIGNMENT 4
vectorizedBootstrap1<-
  function(vec0,nboot=10000,alpha=0.1)
  {
    #vec0 = c(1:10)
    #pretend this is our theoretical sample data
    n<-length(vec0)
    mu0<-mean(vec0)
    sd0<-sd(vec0)

    #generate a vector of length 'nboot' that contains iterative means
    #vectorized implementation
    #Im changing 2 things, creating function that can be applied directly with
apply
    # and changing the input matrix to be full length
    my.sample<-function(x){sample(x,replace=T)}
    bootdist0<-matrix(rep(vec0,nboot),byrow=F)
    #  bootdist<-apply(bootdist0,2,sample,x=vec0,size=nboot*n,replace=T)
    bootdist<-apply(bootdist0,2,my.sample)

    bootdist<-matrix(bootdist,nrow = n,ncol = nboot)
#    print(bootdist[,1:10])
    #convert the sampled data into a studentized vector
    helper <- function(x) ((mean(x)-mu0)/(sd(x)/sqrt(n)))
```

Mark S
Working Alone (assignment 4)
2/12/2022
STAT486

```
    bootvec<-apply(bootdist,2,helper) # Creates bootstrap vector
#    print(bootvec[1:100])
  #  print(mean(bootvec))

    #create bootstrapped CI
    lowerq<-quantile(bootvec,alpha/2,na.rm=T) # Creates the lower quantile
    upperq<-quantile(bootvec,1-alpha/2,na.rm=T) # Creates the upper quantile
#    print(lowerq)
#    print(upperq)

    LB<-mu0-(sd0/sqrt(n))*upperq #switch just like last project  #
    UB<-mu0-(sd0/sqrt(n))*lowerq
#    print(LB)
#    print(UB)

    #create normal CI
    NLB<-mu0-(sd0/sqrt(n))*qnorm(1-alpha/2) # Normal CI Lowerbound
    NUB<-mu0+(sd0/sqrt(n))*qnorm(1-alpha/2) # Normal CI Upperbound
#    print(NLB)
#    print(NUB)

    list(bootstrap.CI=c(LB,UB),normal.CI=c(NLB,NUB)) # Creates the boostrap CI
  }

vectorizedBootstrap <- function(vec0,nboot=10000,alpha=0.1)
{
  #vec0 = c(1:10)
  #pretend this is our theoretical sample data
  n<-length(vec0)
  mu0<-mean(vec0)
  sd0<-sd(vec0)

  #generate a vector of length 'nboot' that contains iterative means
  #vectorized implementation
  bootdist<-matrix(vec0)
  bootdist<-apply(bootdist,2,sample,x=vec0,size=nboot*n,replace=T)
  bootdist<-matrix(bootdist,nrow = n,ncol = nboot)
#  print(bootdist[,1:10])
  #convert the sampled data into a studentized vector
  helper <- function(x) ((mean(x)-mu0)/(sd(x)/sqrt(n)))
  bootvec<-apply(bootdist,2,helper)
#  print(bootvec[1:100])

  #create bootstrapped CI
```

Mark S
Working Alone (assignment 4)
2/12/2022
STAT486

```
  lowerq<-quantile(bootvec,alpha/2,na.rm=T)  # Creates the lower quantile
  upperq<-quantile(bootvec,1-alpha/2,na.rm=T) # Creates the upper quantile
#  print(lowerq)
#  print(upperq)

  LB<-mu0-(sd0/sqrt(n))*upperq #switch just like last project
  UB<-mu0-(sd0/sqrt(n))*lowerq
#  print(LB)
#  print(UB)

  #create normal CI
  NLB<-mu0-(sd0/sqrt(n))*qnorm(1-alpha/2)
  NUB<-mu0+(sd0/sqrt(n))*qnorm(1-alpha/2)
#  print(NLB)
#  print(NUB)
  list(bootstrap.CI=c(LB,UB),normal.CI=c(NLB,NUB))
}
simulate <- function(mu.theoretical=3,n=30,nsim=1000,alpha=0.1)
{
  #create coverage indicator vectors for both bootstrapped and normal dists.
  confidencevec.boot<-NULL
  confidencevec.norm<-NULL
  #real population mean
  mu<-(exp(mu.theoretical+0.5))

  #simulate multiple bootstraps
  for (i in 1:nsim)
  {
    mu0vec<-rlnorm(n,mu.theoretical) #vector of sample means
    tempobj<-vectorizedBootstrap1(mu0vec,alpha=alpha)  # temporary object of
vectorized Bootstrap1
    boot.CI<-tempobj$bootstrap.CI # Creates the bootstrap confidence and tempobj
accesses the bootstrap.CI which is linked to vectorizedBootstrap1
    norm.CI<-tempobj$normal.CI # Creates the bootstrap confidence and tempobj
accesses the normal.CI which is linked to vectorizedBootstrap1
    confidencevec.boot<-c(confidencevec.boot,(boot.CI[1]<mu)*(boot.CI[2]>mu))
    confidencevec.norm<-c(confidencevec.norm,(norm.CI[1]<mu)*(norm.CI[2]>mu))
  }
  #percentage results
  list(boot.coverage=(sum(confidencevec.boot)/nsim),norm.coverage=(sum(confidence
vec.norm)/nsim))
}

print(simulate(mu.theoretical = 3, n = 3, nsim = 1000, alpha = 0.1))
```

Mark S
Working Alone (assignment 4)
2/12/2022
STAT486

```
> print(simulate(mu.theoretical = 3, n = 3, nsim = 1000, alpha = 0.1))
$boot.coverage
[1] 0.836

$norm.coverage
[1] 0.622

> print(simulate(mu.theoretical = 3, n = 10, nsim = 1000, alpha = 0.1))
$boot.coverage
[1] 0.851

$norm.coverage
[1] 0.761

> print(simulate(mu.theoretical = 3, n = 30, nsim = 1000, alpha = 0.1))
$boot.coverage
[1] 0.876

$norm.coverage
[1] 0.817

> print(simulate(mu.theoretical = 3, n = 100, nsim = 1000, alpha = 0.1))
$boot.coverage
[1] 0.901

$norm.coverage
[1] 0.886
```

```
(The above output = Alpha at 0.1)
(The lower output = Alpha at 0.05)
```

```
> print(simulate(mu.theoretical = 3, n = 3, nsim = 1000, alpha = 0.05))
$boot.coverage
[1] 1

$norm.coverage
[1] 0.696

> print(simulate(mu.theoretical = 3, n = 10, nsim = 1000, alpha = 0.05))
$boot.coverage
[1] 0.913

$norm.coverage
[1] 0.813

> print(simulate(mu.theoretical = 3, n = 30, nsim = 1000, alpha = 0.05))
$boot.coverage
[1] 0.917

$norm.coverage
[1] 0.854
> print(simulate(mu.theoretical = 3, n = 100, nsim = 1000, alpha = 0.05))
$boot.coverage
[1] 0.934

$norm.coverage
[1] 0.909
```