Bootstrap and Jack Knife Estimator

Josh Levine	Harsh Patel	Jaini Patel	Yifan Liao	Aayush Shah
jl2108	hkp49	jp1891	yl1463	avs93

Part 1: Build a bias, standard deviation, and confidence interval estimator for an arbitrary statistic (named statfunc) based on the bootstrap (use 10000 =nboot) and the jackknife

What is Bootstrapping?

Bootstrapping is a statistical procedure that resamples a single data set to create many simulated samples. This process allows you to calculate standard errors, construct confidence intervals, and perform hypothesis testing for numerous types of sample statistics. Statistical inference generally relies on the sampling distribution and the standard error of the feature of interest.

What is Jack Knife?

The jackknife is a method used to estimate the variance and bias of a large population. This was the earliest form of sampling methods created. It involves a leave-one-out strategy of the estimation of a parameter (e.g., the mean) in a data set of N observations (or records). Ideally, N − 1 models are built on the data set with different factors left out of each model. The estimates of all models are then aggregated into a single estimate of the parameter. The jackknife gets computationally intractable as $N \to \infty$.

Function Used for Boot Strap:

```
my.bootstrapci.ml<-function(vec0,statfunc=mean,nboot=10000,alpha=0.05)#function my.bootstrapci.ml takes in
 a vector
{
  #extract sample size, mean and standard deviation from the original data
  n0<-length(vec0)
  mean0<-statfunc(vec0)
  sd0<-sqrt(var(vec0))
  sdboot<-NULL
  # create a vector to store the location of the bootstrap studentizeddeviation vector
  bootpivotalvec<-NULL
  bootbiasvec<-NULL
  bootpercentilevec<-NULL
  #jacckbiasvec<-NULL dont need these vectors...this is the bootstrap code not the jackknife code
  #jacksdvec<-NULL
  #create the bootstrap distribution using a for loop
  for(i in 1:nboot)
  {
    vecb<-sample(vec0,replace=T)
    #create mean and standard deviation to studentize
    meanb<-statfunc(vecb)
    sdb<-Jackknife(vecb,statfunc)$jacksd
    #sdb<-sqrt(var(vecb))
    #note since resamplingfull vector we can use n0 for sample size of vecb
    bootpivotalvec<-c(bootpivotalvec,(meanb-mean0)/(sdb/sqrt(n0)))
    #calculate the vector that stores the bias of each sample
    bootbiasvec<-c(bootbiasvec,meanb-mean0)
    bootpercentilevec<-c(bootpercentilevec, meanb)
  }
  #Calculate the mean of the bias and the standard deviation of the vector
  bootbias<-mean(bootbiasvec)
  bootsd <- sd(bootpivotalvec)

  #Calculate lower and upper quantile of the bootstrap distribution
  lq<-quantile(bootpivotalvec,alpha/2)
  uq<-quantile(bootpivotalvec,1-alpha/2)

  #Calculate lower and upper percenitle quantiles
  lqpercentile<-quantile(bootpercentilevec, alpha/2)
  uqpercentile<-quantile(bootpercentilevec, 1-alpha/2)
  #ADD the other two confidence intervals.
  #incorporate into the bootstrap confidence interval (what algebra supports this?) and output result
  LB<-mean0-(sd0/sqrt(n0))*uq
  UB<-mean0-(sd0/sqrt(n0))*lq

  #setting the lower and upper bound percentiles
  LBpercentile<-lqpercentile
  UBpercentile<-uqpercentile

  #since I have the mean and standard deviation calculate the normal confidence interval here as well
  NLB<-mean0-(sd0/sqrt(n0))*qt(1-alpha/2,n0-1)
  NUB<-mean0+(sd0/sqrt(n0))*qt(1-alpha/2,n0-1)

  #standard deviation of the percentile vector
  sdboot<-sd(bootpercentilevec)
  #boot normal confidence lower bound
  BNCLB<-mean0-sdboot*qt(1-alpha/2,n0-1)
  #boot normal confidence upper bound
  BNCUB<-mean0+sdboot*qt(1-alpha/2,n0-1)
```

```
  #returns the results of the bootstrap; including the confidence intervals
  list(normal.confidence.interval=c(NLB,NUB), bootstrap.pivotal.confidence.interval=c(LB,UB), boot.percenti
le=c(LBpercentile, UBpercentile), bootnormal=c(BNCLB,BNCUB),bootbias = bootbias, bootsd = bootsd)
}
```

Function Used for Jack Knife:

```
Jackknife<-function(v1, statfunc = sd, alpha = 0.05)#function jackknife takes in a vector
{
  n1<-length(v1)#set n1 equal to the length of the vector passed as a param
  jackvec<-NULL#creates a vector called jackvec
  mu0<-statfunc(v1)#sets mu0 to the sd of v1

  for(i in 1:n1)#loops over a range of 1 to n1 (which is the length of v1)
  {
    mua<-statfunc(v1[-i]) #sets mua to the sd of n-1 of the values in v1
    jackvec<-c(jackvec, n1*(mu0)-(n1-1)*mua) #combines the existing jackvec with the equation
  }

  jackbias<-mean(jackvec)-mu0#sets the bias to the average of jackvec minus the sd of v1
  jacksd<-sd(jackvec)#sets jacksd to the standard deviation of jackvec

  # Calculate the jacknife confidence intervals - lower bound and upper bound
  jackLowerBound <- mean(jackvec) - (jacksd/sqrt(n1))*qnorm(1-alpha/2)
  jackUpperBound <- mean(jackvec) + (jacksd/sqrt(n1))*qnorm(1-alpha/2)

  list(jackknife.confidence.interval = c(jackLowerBound, jackUpperBound), mu0=mu0,jackbias=jackbias,jacksd=
jacksd)
}
```

Calling the Bootstrap Function

```
my.bootstrapci.ml(1:1000)
```

```
## $normal.confidence.interval
## [1] 482.5774 518.4226
##
## $bootstrap.pivotal.confidence.interval
##    97.5%     2.5%
## 482.7503 518.1687
##
## $boot.percintile
##     2.5%    97.5%
## 482.8469 518.1835
##
## $bootnormal
## [1] 482.6947 518.3053
##
## $bootbias
## [1] -0.0132189
##
## $bootsd
## [1] 0.9951427
```

We see the results for the Confidence, Pivotal, Percentile and Normal intervals for a distribution from 1 to 1000 shown above. Also included in the data is the bias and standard deviation. We notice that all the intervals do produce similar results. This is expected on the data set that is passed. Other parameters could have been passed (and will be in later segments of this report) such as statfunc, nboot, and alpha. Statfunc represents what function we want to us in our bootstrap, it is currently defaulted to mean right now. Nboot determines how many times we want to run the loop inside the bootstrap. And alpha helps calculate the lower and upper bounds of the intervals.

Calling the Jack Knife Function

```
Jackknife(1:1000, mean)
```

```
## $jackknife.confidence.interval
## [1] 482.5991 518.4009
##
## $mu0
## [1] 500.5
##
## $jackbias
## [1] 0
##
## $jacksd
## [1] 288.8194
```

We see the results for the Jack knife confidence interval, mu(mean), bias, and standard deviation. The results for the confidence interval is also similar to the intervals for the bootstrap since we are passing the same exact distribution and are using the mean parameter for the jack knife. Mu is expected to be 500 since our distribution is from 1 to one thousand in this case. The bias is 0 since we calculate that by subtracting the mean of jack vec and the mean of the original vector (this will change when we use rlnorm). Lastly, the function will also return the standard deviation of the the vector.

Part 2: Build a simulator

Function Used for the Simulator: (**Note:** nboot is set to 1000 because when it was set to 10000 the function takes too much processing power)

```
sim.func <- function(mu.val = 3, statfunc = mean, n = 30, nsim = 1000){

  #create coverage indicator vectors for bootstrap and normal
  boot.confidence.intervals <- NULL
  jack.normal.confidence.intervals <- NULL
  boot.percentile.confidence.intervals <- NULL

  #calculate real mean
  mulnorm <- (exp(mu.val+1/2))
  #run simulation
  for(i in 1:nsim){
    #print(i)
    #if((i/100)==floor(i/100)){
    #  print(i)
    #}

    #sample the simulation vector
    vec.sample <- rlnorm(n,mu.val)

    #bootstrap it - change this to be more general cause the bootstrap now calls the jacknife for some reas
on
    boot.list <- my.bootstrapci.ml(vec.sample, statfunc = statfunc ,nboot = 1000, alpha = 0.05)
    #jackknife it
    jack.list <- Jackknife(vec.sample, statfunc = statfunc, alpha = 0.05)

    #fetch confidence intervals
    boot.conf <- boot.list$bootstrap.pivotal.confidence.interval
    jack.conf <- jack.list$jackknife.confidence.interval
    boot.perc <- boot.list$boot.percintile

    #count up the coverage by the bootstrap confidence interval
    boot.confidence.intervals <- c(boot.confidence.intervals,(boot.conf[1]<mulnorm)*(boot.conf[2]>mulnorm))

    #count up the coverage by the jackknife confidence interval
    jack.normal.confidence.intervals <- c(jack.normal.confidence.intervals,(jack.conf[1]<mulnorm)*(jack.con
f[2]>mulnorm))

    #count up the coverage by the bootstrap percentile confidence interval
    boot.percentile.confidence.intervals <- c(boot.percentile.confidence.intervals,(boot.perc[1]<mulnorm)*
(boot.perc[2]>mulnorm))

  }

  #calculate and output coverage probability estimates
  list(boot.confidence.intervals = (sum(boot.confidence.intervals)/nsim), jack.normal.confidence.intervals
 = (sum(jack.normal.confidence.intervals)/nsim), boot.percentile.confidence.intervals = (sum(boot.percentil
e.confidence.intervals)/nsim))
}
```

The function above will create a random distribution using rlnorm and run both the jack knife and bootstrap interval estimator functions. It call each function 1000 times (nsim is defaulted to 1000, you may change this parameter if you like) and then stores the results. After getting the results it calculates the confidence interval for the functions.

Part 3: Run the simulator for samples 10, 30, 100 while alpha = 0.05

Simulation with **n = 10** (Note: alpha is set to 0.05 in the simulation.)

```
n10 <- sim.func(mu.val = 3, statfunc = mean, n = 10, nsim = 1000)
n10
```

```
## $boot.confidence.intervals
## [1] 0.925
##
## $jack.normal.confidence.intervals
## [1] 0.819
##
## $boot.percentile.confidence.intervals
## [1] 0.812
```

Simulation with **n = 30** (Note: n is defaulted to 30 in the simulation, but we set it here for clarity.)

```
n30 <- sim.func(mu.val = 3, statfunc = mean, n = 30, nsim = 1000)
n30
```

```
## $boot.confidence.intervals
## [1] 0.922
##
## $jack.normal.confidence.intervals
## [1] 0.874
##
## $boot.percentile.confidence.intervals
## [1] 0.872
```

Simulation with **n = 100** (Note: nsim is also already defaulted to 100, but we include it here for clarity.)

```
n100 <- sim.func(mu.val = 3, statfunc = mean, n = 100, nsim = 1000)
n100
```

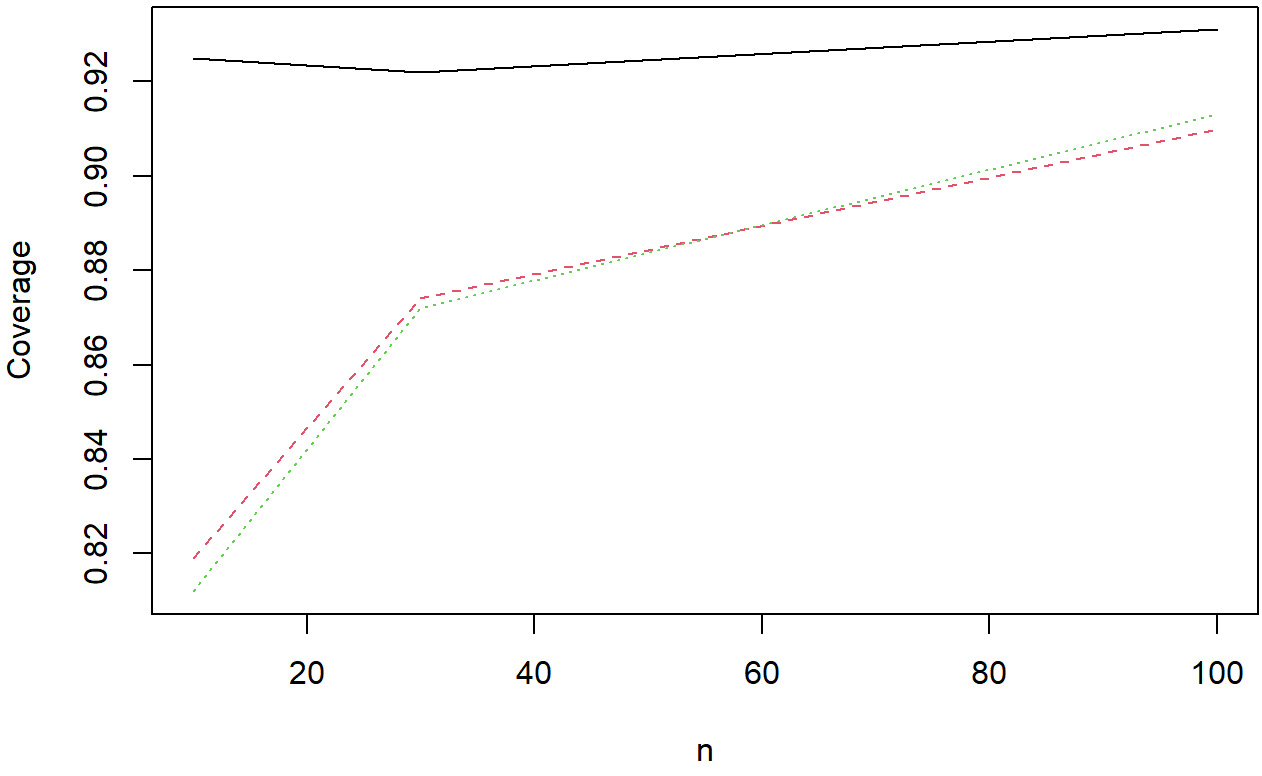
```
## $boot.confidence.intervals
## [1] 0.931
##
## $jack.normal.confidence.intervals
## [1] 0.91
##
## $boot.percentile.confidence.intervals
## [1] 0.913
```

Results:

```
data<-matrix(c(n10$boot.confidence.intervals, n10$jack.normal.confidence.intervals, n10$boot.percentile.confidence.intervals, n30$boot.confidence.intervals, n30$jack.normal.confidence.intervals, n30$boot.percentile.confidence.intervals, n100$boot.confidence.intervals, n100$jack.normal.confidence.intervals, n100$boot.percentile.confidence.intervals),ncol=3,byrow=TRUE)
colnames(data) <- c("Bootstrap Pivotal CI","Jack Knife Normal CI", "Bootstrap Percentile CI")
rownames(data) <- c("10","30","100")
data <- as.table(data)
data
```

```
##     Bootstrap Pivotal CI Jack Knife Normal CI Bootstrap Percentile CI
## 10                 0.925                0.819                   0.812
## 30                 0.922                0.874                   0.872
## 100                0.931                0.910                   0.913
```

```
matrixTest=matrix(c(n10$boot.confidence.intervals,n30$boot.confidence.intervals,n100$boot.confidence.intervals,n10$jack.normal.confidence.intervals,n30$jack.normal.confidence.intervals,n100$jack.normal.confidence.intervals,n10$boot.percentile.confidence.intervals,n30$boot.percentile.confidence.intervals,n100$boot.percentile.confidence.intervals),nrow=3,ncol=3)
matplot(x=c(10,30,100),y=matrixTest,type='l', xlab="n", ylab="Coverage")
```



From the results:

Note: The Top black line represents the Bootstrap Pivotal CI. The dotted red line represents the Jack Knife Normal Ci. And the dotted green line represents the Bootstrap Percentile CI

1. All coverage rates increase as sample size increases.
2. Coverage for the bootstrap pivotal CI is the highest among all three.
3. The Jack Knife normal CI and Bootstrap Percentile CI have very similar results as you see in both the table and graph above.

Part 4: For the standard deviation of the normal distribution, estimate the bias of the the sample standard deviation when dividing by n, compare the bootstrap and the jackknife (10000 simulations).

Jack Knife Standard Deviation:

```
Jackknife_sd <- function(v1){
  n1 <- length(v1)
  jackvec <- NULL
  mu0 <- sd(v1)/n1
  for(i in 1:n1){
    mua <- sd(v1[-i])/(n1-1)
    jackvec <- c(jackvec, n1*(mu0)-(n1-1)*mua)
  }
  jackbias <- mean(jackvec) - mu0
  return (jackbias)
}
```

Bootstrap Standard Deviation:

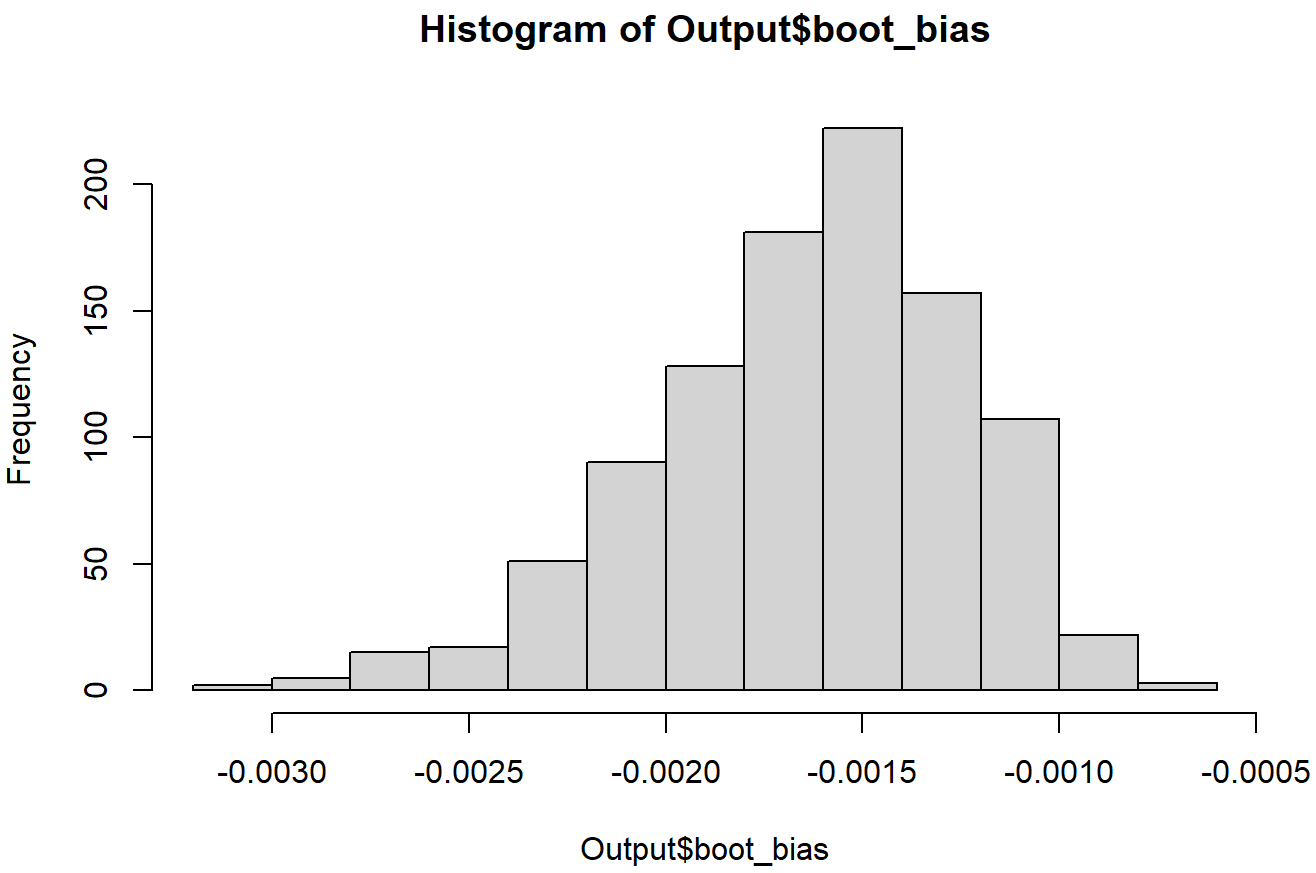
```
Bootstrap_sd <- function(vec0, nboot = 10000){
  #extract sample size, mean and standard deviation from the original data
  n <- length(vec0)
  mean0 <- sd(vec0)/n
  bootvec <- NULL
  bootbiasvec <- NULL
  #create the bootstrap distribution using a for loop
  for(i in 1:nboot){
    6
    vecb <- sample(vec0, replace = T)
    #create mean and standard deviation to studentize
    meanb <- sd(vecb)/n
    #note since resampling full vector we can use n0 for sample size of vecb
    bootvec <- c(bootvec, meanb)
    #Calculation the vector that stores the bias of each bootstap sample
    bootbiasvec <- c(bootbiasvec, meanb-mean0)
  }
  return(mean(bootbiasvec))
}
```

Simulator to compare both the Jack Knife and the Bootstrap Standard Deviations:

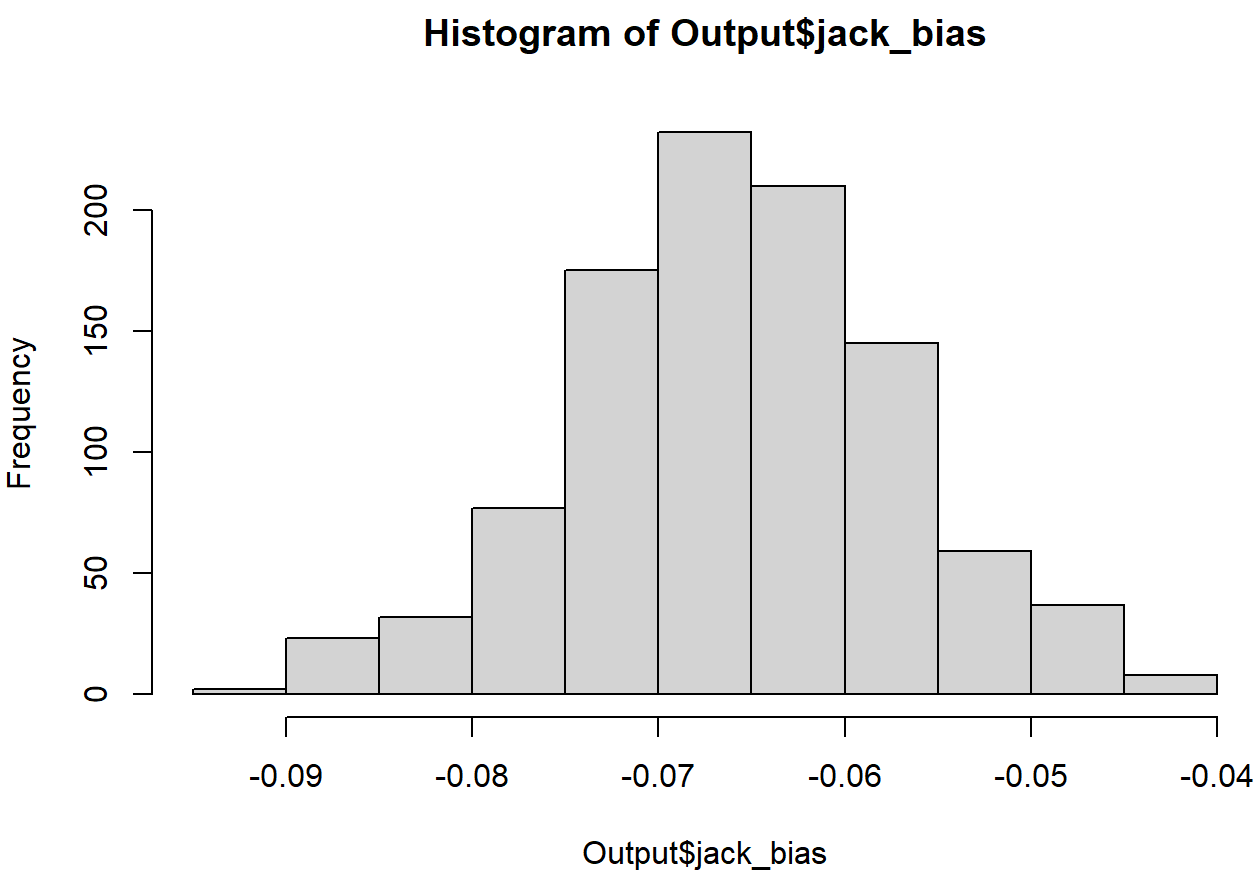
```
simulation_for_jackknife_and_bootstrap <- function(mu = 3, sd = 2, n = 30 , nsim = 4){
  #create coverage indicator vectors for bootstrap and normal
  bvec.boot <- NULL
  bvec.jack <- NULL
  #run simulation
  for(i in 1:nsim){
    #if((i/100)==floor(i/100)){
    #  print(i)
    #}
    #sample the simulation vector
    vec.sample <- rnorm(n, mean = mu, sd = sd)
    #bootstrap bias
    bvec.boot <- c(bvec.boot, Bootstrap_sd(vec.sample, nboot = 1000))
    #jackknife bias
    bvec.jack <- c(bvec.jack, Jackknife_sd(vec.sample))
  }
  list(boot_bias = bvec.boot, jack_bias = bvec.jack)
}

Output <- simulation_for_jackknife_and_bootstrap(mu = 3, sd = 2, n = 30 , nsim = 1000)
```

The function above will simulate both the jack knife and bootstrap and save the output into a variable called Output. We will use that output for both the jack knife and bootstrap to graph their values.

```
hist(Output$boot_bias)
```



```
hist(Output$jack_bias)
```



We do observe that the Jack Knife histogram distribution is normal. The bootstrap histogram is relatively normal but slightly skewed.

Extra Credit: Run the bootstrap and replace the for loop with the apply method

```
my.bootstrapciapply.ml<-function(vec0,nboot=10000,alpha=0.05)#function my.bootstrapci.ml takes in a vector
{
  #extract sample size, mean and standard deviation from the original data
  n0<-length(vec0)
  mean0<-mean(vec0)
  sd0<-sqrt(var(vec0))
  # create a vector to store the location of the bootstrap studentizeddeviation vector
  bootvec<-NULL
  bootbiasvec<-NULL

  #create the bootstrap distribution using a for loop
  x <- matrix(sample(vec0,nboot*n0,replace=T), nboot, n0)#Create a matrix for bootstrap run
  meanb<-apply(x, 1, mean)# mean of sample
  sdb<-apply(x, 1, sd)# standard deviation of sample

  #note since resampling full vector we can use n0 for sample size of vecb
  bootvec<-c(bootvec,(meanb-mean0)/(sdb/sqrt(n0))) #resampling and storing in the bootvec
  bootbiasvec<-c(bootbiasvec,meanb-mean0) # creating a bootbias vec and storing the difference of mean in e
ach sample

  #Calculate the mean of the bias and the standard deviation of the vector
  bootbias<-mean(bootbiasvec)
  bootsd <- sd(bootvec)

  #Calculate lower and upper quantile of the bootstrap distribution
  lq<-quantile(bootvec,alpha/2)
  uq<-quantile(bootvec,1-alpha/2)

  #ADD the other two confidence intervals.
  #incorporate into the bootstrap confidence interval (what algebra supports this?) and output result
  LB<-mean0-(sd0/sqrt(n0))*uq
  UB<-mean0-(sd0/sqrt(n0))*lq

  #since I have the mean and standard deviation calculate the normal confidence interval here as well
  NLB<-mean0-(sd0/sqrt(n0))*qnorm(1-alpha/2)
  NUB<-mean0+(sd0/sqrt(n0))*qnorm(1-alpha/2)

  list(bootstrap.confidence.interval=c(LB,UB), normal.confidence.interval=c(NLB,NUB), bootbias = bootbias,
 bootsd = bootsd)
}

my.bootstrapciapply.ml(rlnorm(10),10000,0.05)
```

```
## $bootstrap.confidence.interval
##     97.5%      2.5%
## 0.7004426 2.8741352
##
## $normal.confidence.interval
## [1] 0.6867017 2.2995476
##
## $bootbias
## [1] -0.001447877
##
## $bootsd
## [1] 1.401075
```

Above is our attempt at calculating the bootstrap without a loop using the apply method. You can see the normal and confidence CI calculated. As well as the bias and standard deviation. We call the function using a random log distribution using the rlnorm method.