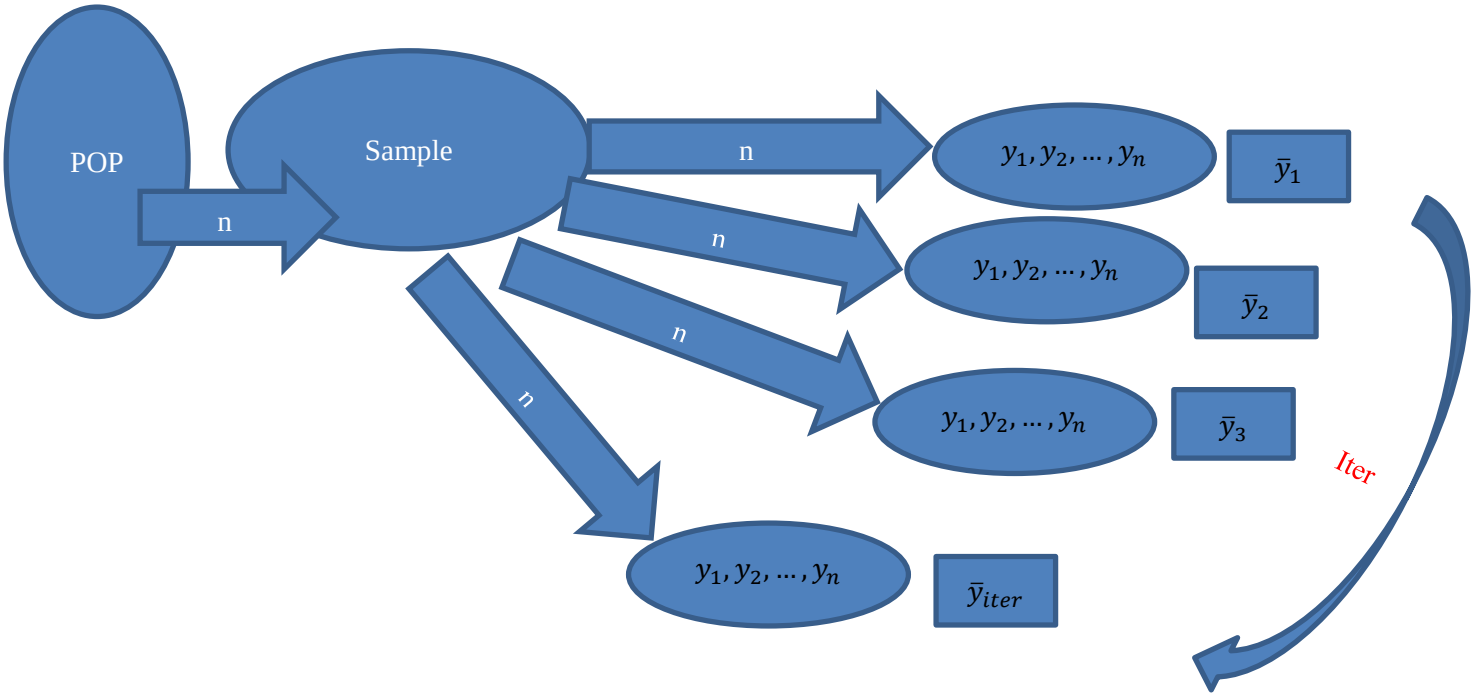
ISE 3293/5013 Laboratory 9
BOOTSTRAP and confidence intervals

In this lab we will learn about two types of estimate:

1. Point
2. Interval

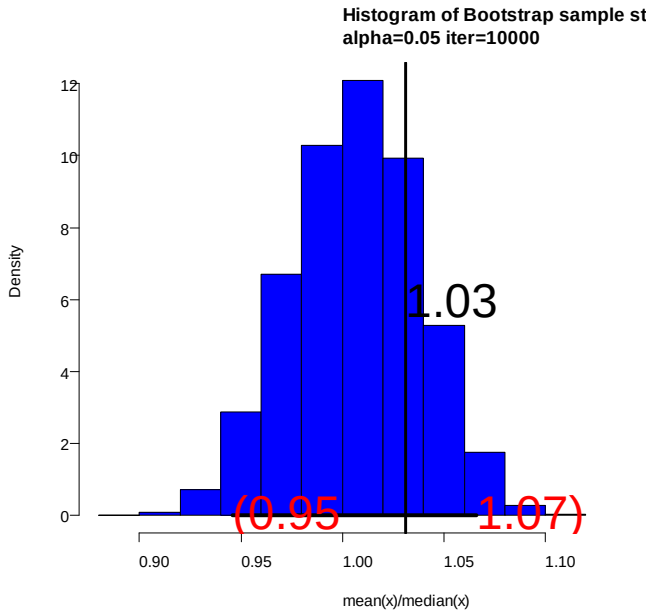
These estimates are important and will be studied in detail as we progress through chapter 7 of MS. We will use a simulation technique called the bootstrap. This gets its name from the idea that we derive everything from what we have at hand – that is we pull ourselves up from our bootstraps. Specifically this means that we take one sample from the population of interest and assume that the sample is representative, that is, it contains information that is summative of the population parameters. We will use the sample by *resampling* from it.



The R function `sample()` will be used to do the resampling. If the sample is of size n then the bootstrap re-sampling will create samples of size n *with replacement*. The following is a function that will facilitate the bootstrap procedure.

```
myboot<-function(iter=10000,x,fun="mean",alpha=0.05,...){

#Notice where the ... is repeated in the code
n=length(x)   #sample size

#Now sample with replacement
y=sample(x,n*iter,replace=TRUE) #A

# Make a matrix with all the resampled values
rs.mat=matrix(y,nr=n,nc=iter,byrow=TRUE)
xstat=apply(rs.mat,2,fun)
# xstat is a vector and will have iter values in it
ci=quantile(xstat,c(alpha/2,1-alpha/2)) #B
# Nice way to form a confidence interval
# A histogram follows
#  The object para will contain the parameters used to make the histogram
para=hist(xstat,freq=FALSE,las=1,main="Histogram of Bootstrap sample statistics",...)

#mat will be a matrix that contains the data, this is done so that I can use apply()
mat=matrix(x,nr=length(x),nc=1,byrow=TRUE)

#pte is the point estimate
#This uses whatever fun is
pte=apply(mat,2,fun)
abline(v=pte,lwd=3,col="Black")# Vertical line
segments(ci[1],0,ci[2],0,lwd=4)      #Make the segment for the ci
text(ci[1],0,paste("(",round(ci[1],2),sep=""),col="Red",cex=3)
text(ci[2],0,paste(round(ci[2],2),")",sep=""),col="Red",cex=3)

# plot the point estimate 1/2 way up the density
text(pte,max(para$density)/2,round(pte,2),cex=3)

return(list(ci=ci,fun=fun,x=x))# Some output to use if necessary
}
```

The above code will be used to make a number of plots, point and interval estimates and useful output in the form of a list.



Tasks

All output made please copy and paste into **this word file**. Save and place in the dropbox when completed. Anything you are asked to make should be recorded under the question in this document. There will be two files you need to upload:

- a pdf of this document (pdf) or the word file (docx)
- a text file of all the code you used to create answers (txt)

Note: All plots you are asked to make should be recorded in this document.

- Task 1
  - Make a folder LAB9
  - Download the file "lab9.r"
  - Place this file with the others in LAB9.
  - Start Rstudio
  - Open "lab9.r" from within Rstudio.
  - Go to the "session" menu within Rstudio and "set working directory" to where the source files are located.
  - Issue the function `getwd()` and copy the output here.

```
F:/Google Drive - Saied/Courses/02 OU/11 Fundamentals of Engine
ering Statistical Analysis/02 Labs/09 Lab 9
```

- Task 2
  - Create your own R file and record the R code you used to complete the lab.
  - In the above code for `myboot()`there are two lines marked A and B. Using any resources available explain what each line does
    - Line A

```
We have set of random variable as x which may be
created by a random generator function. It is an input
argument for this function. This set of numbers, x,
```

could belongs to any type of distributions. At this line, we make n*iter random samples from set of x with replacement.

- Line B

  On this line we get the quantile values for alpha/2 and 1-alpha/percent of given xstat data.

- The sample() function should be studied a little further. As used in the myboot() function, each datum in x will be selected with equal probability. Why is this necessary?

  Because the distribution of x, set of sampling population, is not defined. It could be any distribution. The chance of selecting for all elements are same. But the elements could be repeated in the given set. The repetition change the probability for that specific element.

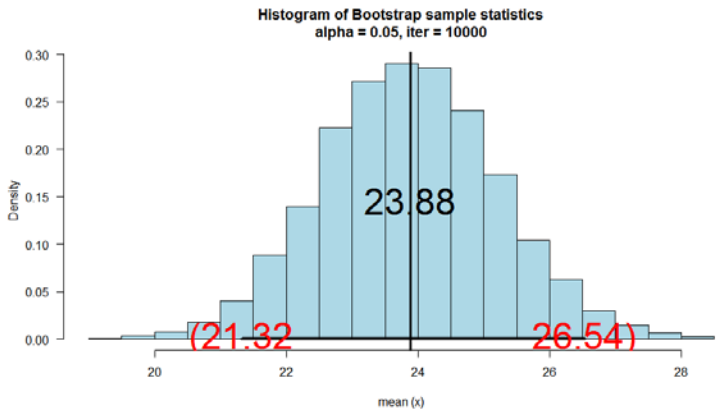
- Issue the following lines in R and record the unique sample you get
  - `set.seed(35) # This will give everyone the same sample`
  - `sam=round(rnorm(20,mean=10,sd=4),2)`
  - `unique(sample(sam,20,replace=TRUE) ) # repeat this line 5Xs`
  - Explain what you see.

    There was different amount of unique numbers. Because of the replacement we got some repeated numbers.
  - `unique(sample(sam,20,replace=FALSE) ) # repeat this line 5Xs`
  - Explain what you see.

    There were not repeated numbers available in the sample set. Because when we used a number it could not be reused. Replacement argument was false. Therefor in every run, we have same amount numbers as outpout.
- Issue `sample(sam,21,replace=FALSE)` what happens? Why?

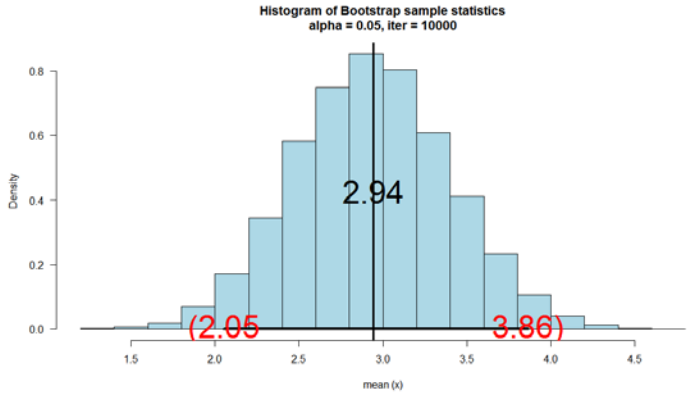
  Got error. Because it is not possible to draw 21 samples from a set with 20 elements. The maximum possible sample size without replacement is the number of sampling set.
  In other other words for this special case, in the bootstrap technique we take a sample from the population and take further resamples from the original sample. So, to have a constant sample for all our resampling it is important to use replacement and get a unique sample which is why each datum has equal probability.

- Task 3
  - Using myboot() make 95% (alpha=0.05) bootstrap intervals (iter=10000) for the population mean μ and record the plots when the following samples are used:
    - A) `set.seed(39); sam=rnorm(25,mean=25,sd=10)`



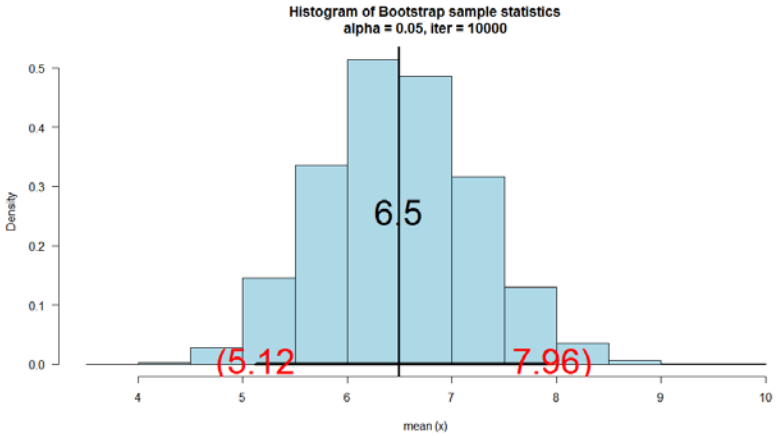

```
> mean(sam)
[1] 23.88103
```

- B) `set.seed(30); sam=rchisq(20,df=3)`



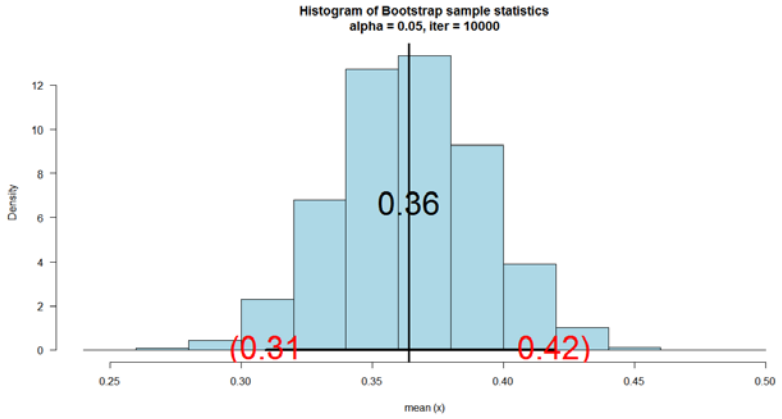

```
> mean(sam)
[1] 2.941449
```

- C) `set.seed(40); sam=rgamma(30,shape=2,scale=3)`



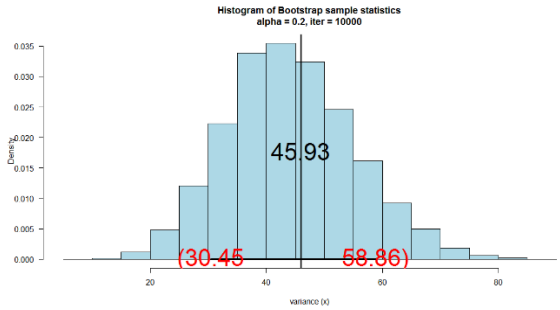

```
> mean(sam)
[1] 6.495501
```

- D) `set.seed(10); sam=rbeta(20,shape1=3,shape2=4)`



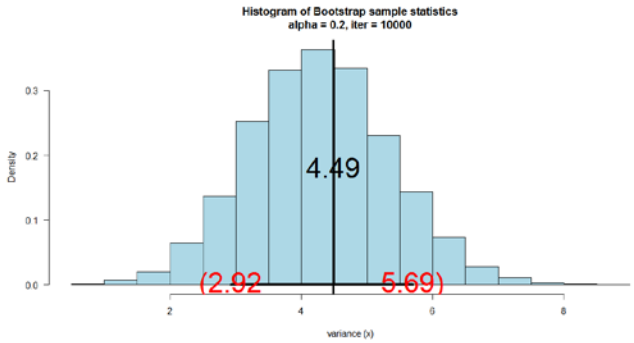

```
> mean(sam)
[1] 0.3640679
```

- In each of the above cases how close is the point estimate to the population value? HINT: You will need to calculate the population mean.
- In each of the above cases does the interval contain the population value? (HINT: You will need to calculate the population mean. See MS chapter 5 or use Wikipedia)
- Using myboot() make 80% (alpha=0.20) bootstrap intervals (iter=10000) for the population variance σ^2 and record the plots when the following samples are used:
  - A) `set.seed(39); sam=rnorm(25,mean=25,sd=10)`



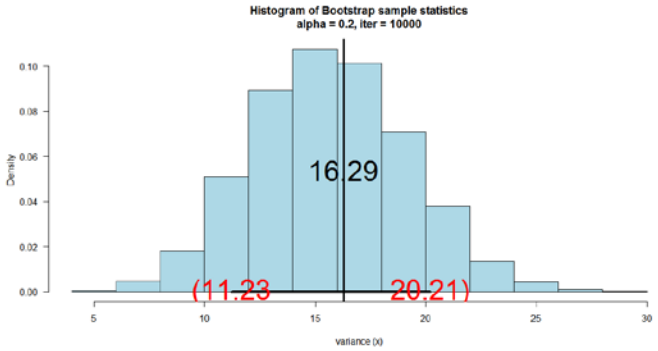

```
> var(sam)
[1] 45.92561
```

  - B) `set.seed(30); sam=rchisq(20,df=3)`



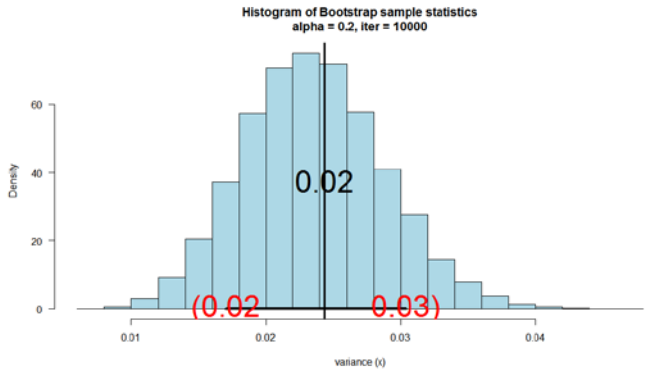

```
> var(sam)
[1] 4.489942
```

  - C) `set.seed(40); sam=rgamma(30,shape=2,scale=3)`



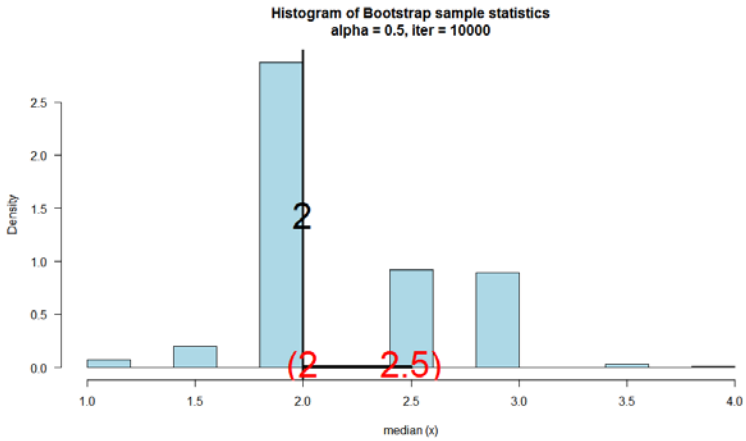

```
> var(sam)
[1] 16.2863
```

- D) `set.seed(10); sam=rbeta(20,shape1=3,shape2=4)`



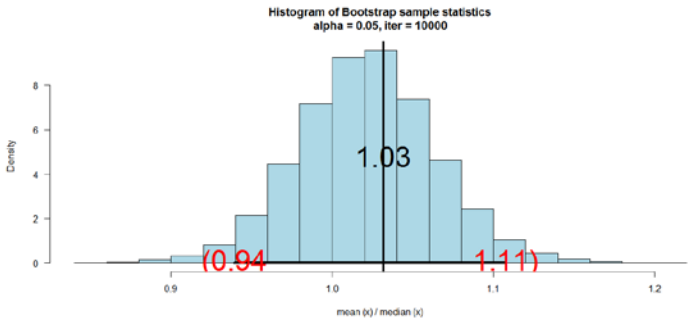

```
> var(sam)
[1] 0.02437883
```

- Task 4
  - Adjust `myboot()` so that it returns as a part of the list the vector containing the statistic (xstat) of interest along with all the other existing quantities.
  - Using the adjusted myboot() function call myboot(x=sam,fun=``median"), where sam=c(1,1,1,2,2,2,2,3,3,3,4,4) - make a barplot of xstat.
  - What is the bootstrap interval estimate for the median? (L,U)



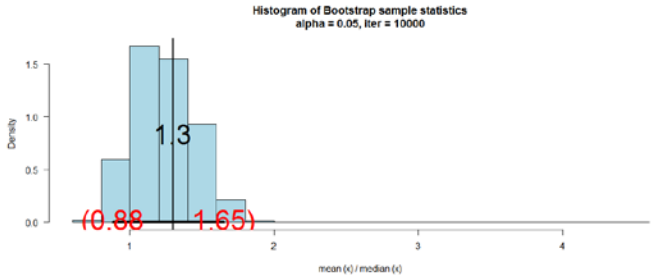

```
> median (sam)
[1] 2
```

- Task 5
  - Find a 95% interval estimate for the population mean/median (mean divided by median) using each of the following samples
    - A) `set.seed(39); sam=rnorm(25,mean=25,sd=10)`



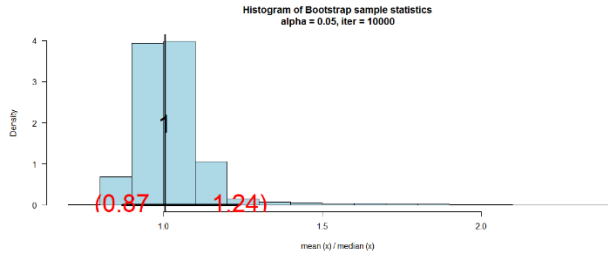

```
> mean(sam)/median(sam)
[1] 1.03183
```

    - B) `set.seed(30); sam=rchisq(20,df=3)`



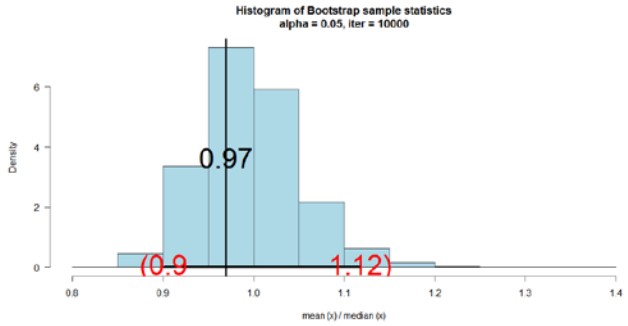

```
> mean(sam)/median(sam)
[1] 1.297476
```

    - C) `set.seed(40); sam=rgamma(30,shape=2,scale=3)`



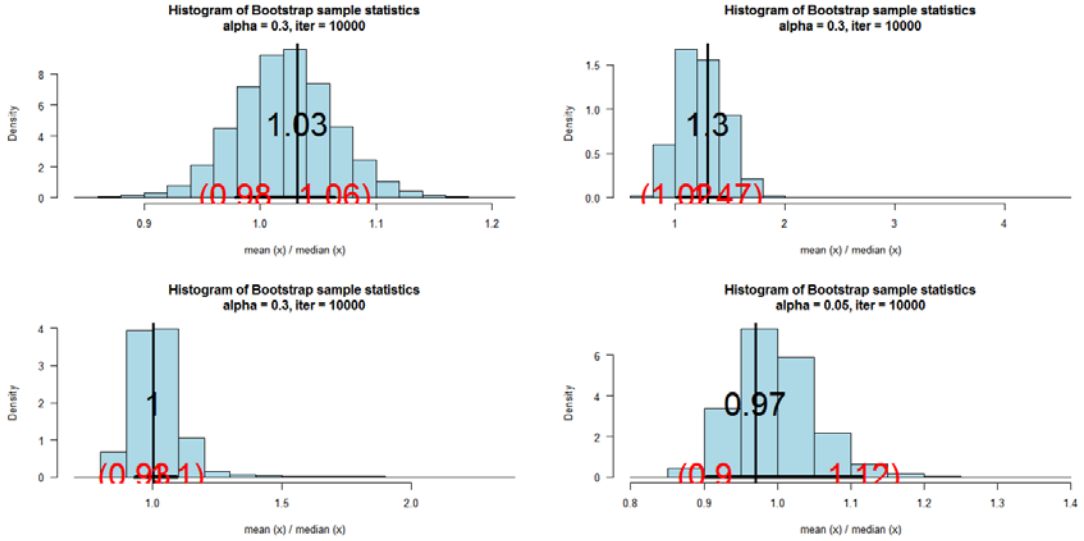

```
> mean(sam)/median(sam)
[1] 1.004388
```

    - D) `set.seed(10); sam=rbeta(20,shape1=3,shape2=4)`

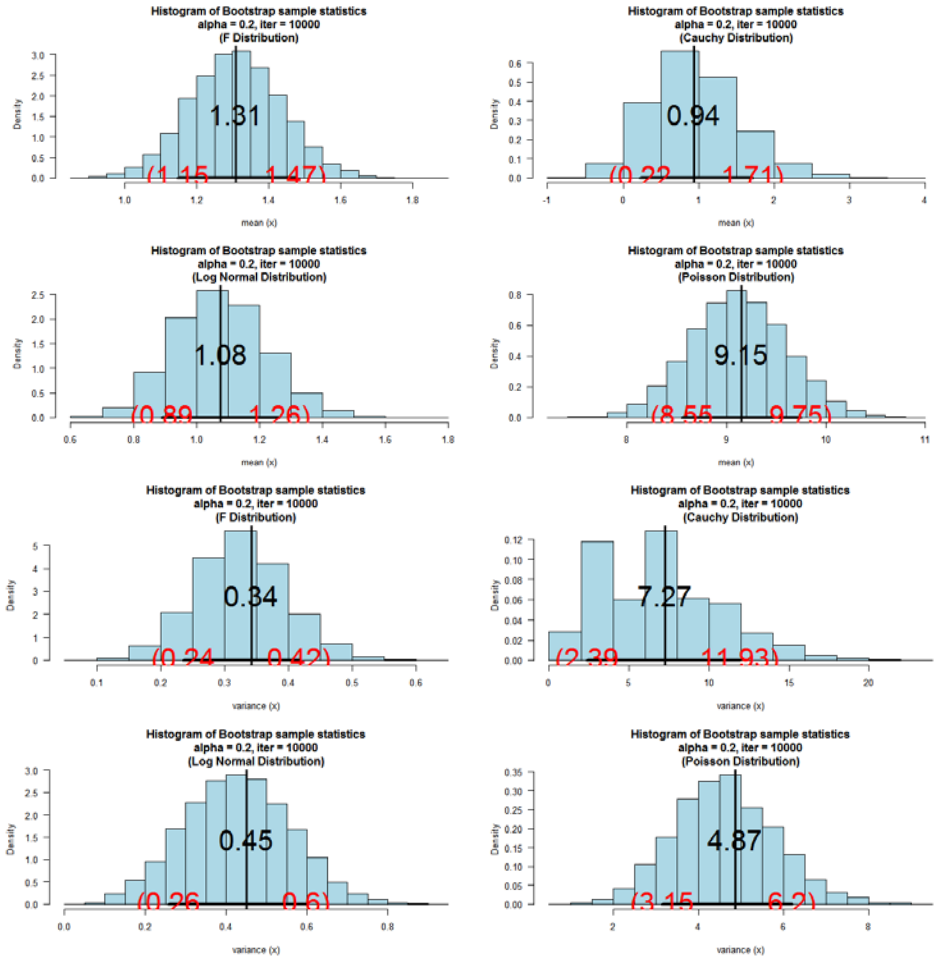



```
> mean(sam)/median(sam)
[1] 0.9698212
```

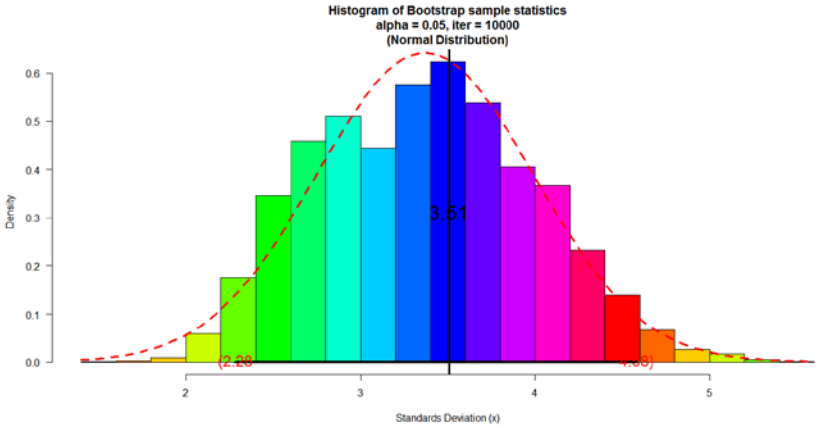
- Do the same except make 70% intervals.

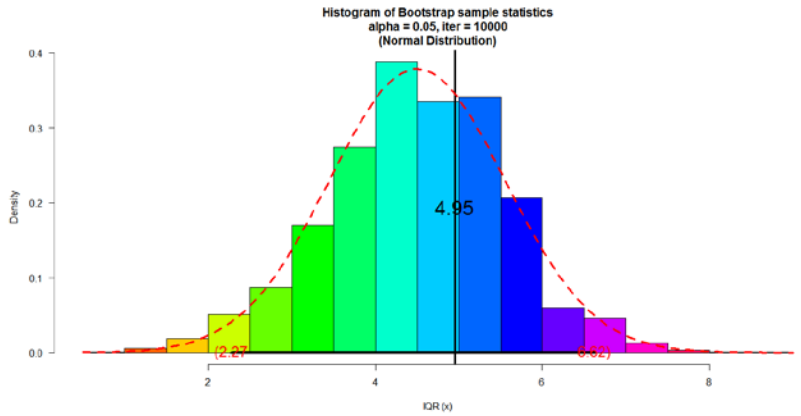


- Task 6
  - Issue the command `?distributions` in R
  - Choose four distributions that we haven't used this far and make one sample of size 20 from each.
  - Make 80% bootstrap intervals for the mean and variance using each sample. Use iter=10000.
  - Record the graphical output.



- Task 7
  - Use `myboot()`and create bootstrap intervals using two statistics that you find interesting. Use `set.seed(68); sam=rnorm(20,mean=10,sd=4)` as the sample. (You might try some functions like `quantile, IQR, sd, var, mean, median`)

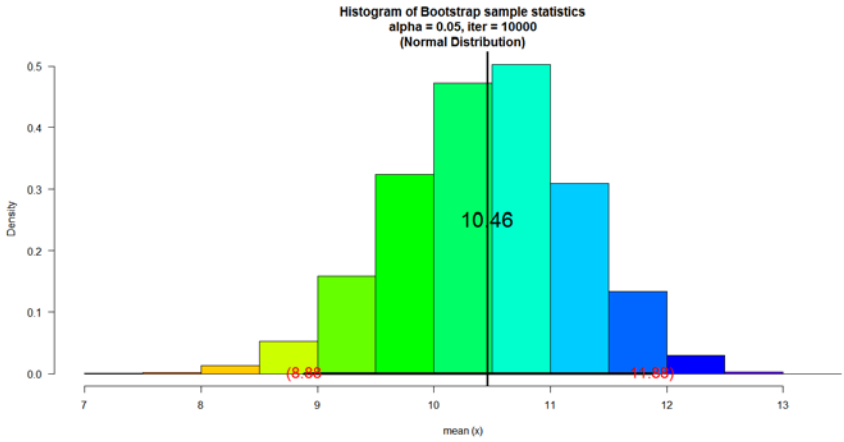




- Theory demands that if $Y \sim N(\mu, \sigma), then\ P\left(\bar{Y} - z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{Y} + z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}\right) = 1 - \alpha$. This defines $\bar{y} - z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{y} + z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}$ to be the $(1-\alpha)100\%$ confidence interval, where $z_{\frac{\alpha}{2}}$ =`qnorm(1-alpha/2,mean=0,sd=1)`, that is the $1-\alpha/2$ standard normal quantile.
- Calculate the 95% confidence interval using the above theory using the sample `set.seed(68); sam=rnorm(20,mean=10,sd=4)`

```
> za = qnorm(1-alpha/2, mean = 0, sd = 1)
> L = round(mean(sam) - za * sqrt(var(sam)/n), 2)
> U = round(mean(sam) + za * sqrt(var(sam)/n), 2)
> L; mean(sam); U
[1] 8.92
[1] 10.45931
[1] 12
```

- Use `myboot()` with the same sample to find a 95% bootstrap interval using the same sample. (Decide on the options you need).

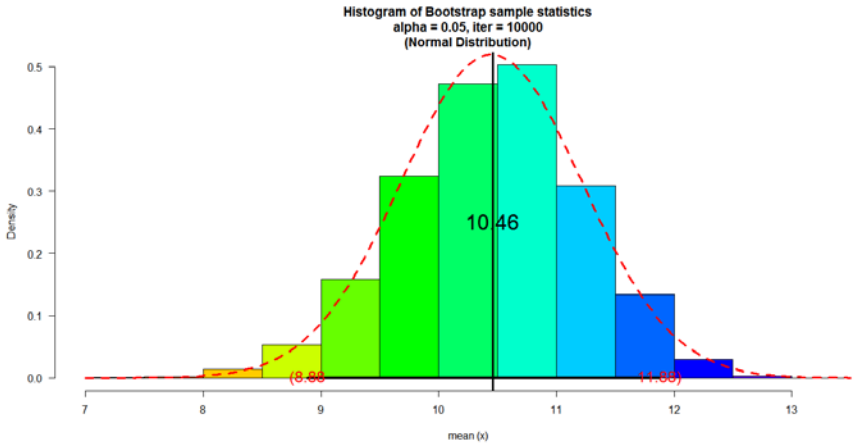


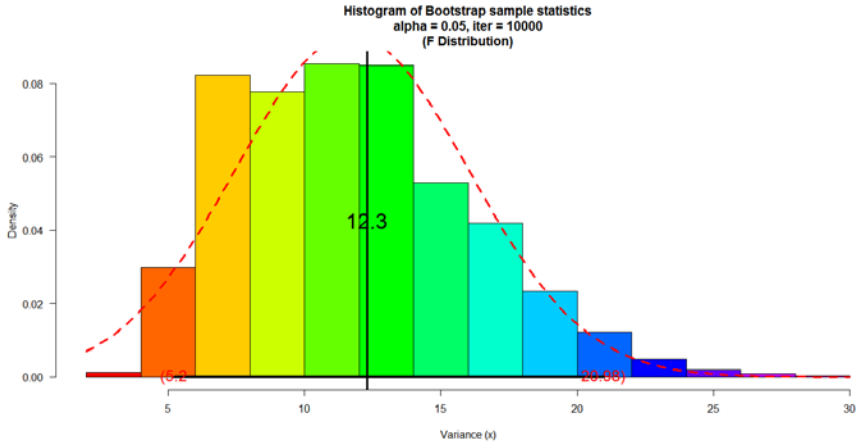
- How do they compare?

```
The values for L, U, and mean are very close as given in
previous sections.
```

################### LAB FINISHES HERE ################################

- Task 8:: Extra for experts!
  - Rewrite myboot() so that it creates interesting plots


Histogram of Bootstrap sample statistics
alpha = 0.05, iter = 10000
(Normal Distribution)
Density
0.0
0.1
0.2
0.3
0.4
0.5
10.46
(8.88
11.98)
7
8
9
10
11
12
13
mean (x)


Histogram of Bootstrap sample statistics
alpha = 0.05, iter = 10000
(F Distribution)
Density
0.00
0.02
0.04
0.06
0.08
12.3
(5.2
28.98)
5
10
15
20
25
30
Variance (x)