STAT 2450 Assignment 5

James MacPhee

Banner: B00768516

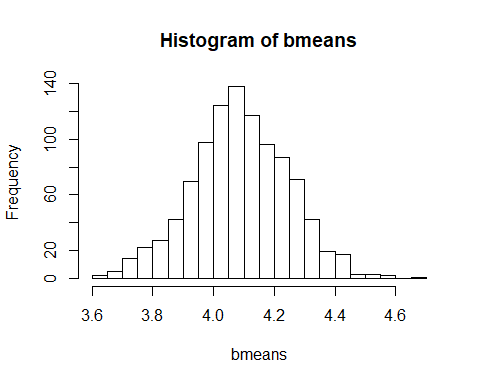
set.seed(87612345)  
data=rnorm(20,mean=4,sd=.75)

t.test(data,conf.level=.95)  
## One Sample t-test  
##   
## data: data  
## t = 24.626, df = 19, p-value = 7.037e-16  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 3.740388 4.435262  
## sample estimates:  
## mean of x   
## 4.087825

bmeans=NULL;  
for(i in 1:1000){  
 bdata=sample(data,20,replace=T)  
 bmeans=c(bmeans,mean(bdata))  
}

(1a)

hist(bmeans,nclass=20)



#I see that this makes the histogram have 21 bars, so I tried to use 'breaks=20' instead.  
#That returned the same histrogram so I read the documentation on hist() and,  
#it turns out that it uses the closet value to the entered one that still makes the graph,  
# looks like a graph. To actually get the specified number of bars one needs to pass in a vector.

Does the histogram look like it has a normal shape?

* Yes, it looks a lot like the normal distribution pdf.

(1b)

quantile(bmeans,probs=c(.025,.975))

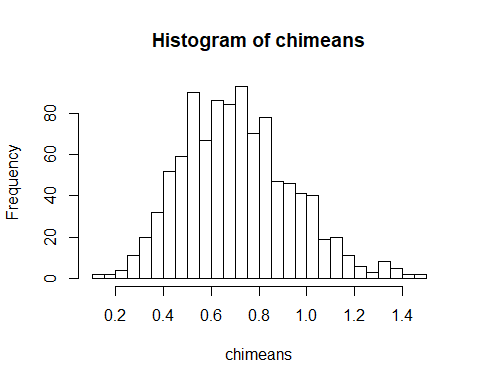
## 2.5% 97.5%   
## 3.761350 4.402784

set.seed(117426)  
data=rchisq(20,1)

(2a)

chimeans=NULL;  
for(i in 1:1000){  
 chidata=sample(data,20,replace=T)  
 chimeans=c(chimeans,mean(chidata))  
}

hist(chimeans,nclass=20)



#Same problem with number of bars as first histogram

(2b)

quantile(chimeans,probs=c(.025,.975))

## 2.5% 97.5%   
## 0.318208 1.213153

(2c)

t.test(data,conf.level=.95)  
## One Sample t-test  
##   
## data: data  
## t = 3.0098, df = 19, p-value = 0.007204  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 0.213148 1.186416  
## sample estimates:  
## mean of x   
## 0.6997818

#Yes, the two confidence intervals were as close as they were in Q1.

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)

var(bootstrapmedians)

## [1] 1.783926

(3b)

bootstrapmeans=apply(M,1,mean)  
var(bootstrapmeans)

## [1] 1.122827

(3c) Yes, I think .9635856 is quite close to 1.

1. myfun(x) returns the sum of all the values of the vector ‘x’.

myfun=function(x){  
 if(length(x)==1){return(exp(x))  
 } else  
 return(exp(x[1])+myfun(x[-1]))  
}  
myfun(c(1:3)) #here's what happens when it is applied to 1:1000

## [1] 30.19287