Economics 403A: Homework 2

Ross Lewis

October 12, 2018

1 - a

```
#Number of observations
n = 1600

#Sample proportion
phat = 8/1600

#Standard Error
se = sqrt(((phat)*(1 - phat))/n)

#To find a 99% confidence interval,
#we add and subtract (2.58 * se) to our sample proportion
lowerbound = phat - (2.58 * se)
upperbound = phat + (2.58 * se)
cat(1600*lowerbound, 'is the lower bound and',1600*upperbound,'is the upper bound.')
```

```
## 0.7209242 is the lower bound and 15.27908 is the upper bound.
```

1 - b

We can find the number of samples from the Error equation:

CI = phat +- z alpha/2 * sqrt((phat(1-phat))/n)

E = z alpha/2 sqrt((phat(1-phat)/n))

n = (phat(1-p)z$^{2)/e}$2

```
#We can find the number of samples from the Error equation:
#CI = phat +- z alpha/2 * sqrt((phat(1-phat))/n)
#E = z alpha/2 sqrt((phat(1-phat)/n))
#n = (phat(1-p)z^2)/e^2

# we want our error to be .008, so
#.008 = 2.58 * sqrt((.005(1-.005)/n)) or
#n = (.005(1-.005)2.58^2)/.008^2
n = .005*(1-.005)*(2.58^2)/(.008^2)
print(n)
```

```
## [1] 517.4311
```

```
#We can also find this by increasing n from 1 to infinity until the correct error is found
#Number of observations
for(n in 1:2000){
  #Sample proportion
  phat = .005

  #Standard Error
  se = sqrt(((phat)*(1- phat))/n)
```

```
  lowerbound = phat - (2.58 * se)
  upperbound = phat + (2.58 * se)

  if(abs(phat-upperbound) < .008 & abs(phat-upperbound) < .008){
    print(n)
    break
  }
}
```

```
## [1] 518
```

1 - c

Without a Phat value, and trying to find the number of samples it would take to get a 99% confidence interval, we have to assume the worst Phat value of .5. Any other value of phat will give you a smaller numerator, and a smaller minimum n.

```
#Without a Phat value, and trying to find the number of
#samples it would take to get a 99% confidence interval,
#we have to assume the worst Phat value of .5.  Any other
#value of phat will give you a smaller numerator, and a
#smaller minimum n.

# we want our error to be .008, so
#.008 = 2.58 * sqrt((.5(1-.5)/n)) or
#n = (.5(1-.5)2.58^2)/.008^2
n = .5*(1-.5)*(2.58^2)/(.008^2)
print(n)
```

```
## [1] 26001.56
```

```
#We can also find this by increasing n from 1 to infinity
#until the correct error is found
#And we can also try different proportions from .01 to .99
#to make sure our answer is correct
#Number of observations
allTrue = T
for(n in 1:100000){
  for(prop in seq(.01,.99,.01)){
    #Sample proportion
    phat = (n*prop)/n

    #Standard Error
    se = sqrt(((phat)*(1- phat))/n)

    lowerbound = phat - (2.58 * se)
    upperbound = phat + (2.58 * se)

    if(abs(phat-upperbound) >= .008 & abs(phat-upperbound) >= .008){
      allTrue = F
      break
    }
  }
  if(allTrue){
    print(n)
```

```
    break
  }
  allTrue = T
}
```

```
## [1] 26002
```

1 - d

We needed 26002 samples for a 99% confidence interval when we had no preliminary estimate, and only 518 when we started with a p hat. This shows that it is much better to have some data to work off of than going in to a problem blind (without any estimations).

2 - a

We can use a binomial distribution we want to find the probability of getting 0 or 1 from a binomial distribution of size 10 and probability .3.

```
#We can use a binomial distribution
#we want to find the probability of getting 0 or 1 from a binomial
#distribution of size 10 and probability .3
print(pbinom(1,size=10,prob=.3))
```

```
## [1] 0.1493083
```

2 - b

We now think p is .3 but it is actually .2 that means there is a 1 - 0.3758096 probability of a type II error (not rejecting the H0 of p=.3 when it is actually 0.2).

```
# we now think p is .3 but it is actually .2
#that means there is a 1 - 0.3758096 probability of a type II error
#(not rejecting the H0 of p=.3 when it is actually 0.2

1-pbinom(1,size=10,prob=.2)
```

```
## [1] 0.6241904
```

2 - c

In this case, H0 = .3 and we reject H0 if 1 or fewer residents in a random sample of 10 residents favor the proposal. The power is simply 1 - the probability of a type II error, which in this case is 0.3758096

```
#in this case, H0 = .3 and we reject H0 if 1 or fewer residents
#in a random sample of 10 residents favor the proposal
#The power is simply 1 - the probability of a type II error, which in this case is 0.3758096

1-(1-pbinom(1,size=10,prob=.2))
```

```
## [1] 0.3758096
```

3 - a

I maximized the probability without going over .1 by both fitting the data to a normal distribution and finding the probability of going over 6 and under 1. I also took the sum of binomail probabilities 0, 1, 7, 8, and 9 for a size of 9 and a probability of .4. Both show c1 = 1 and c2 = 7

```
library(Rlab)
```

```
## Rlab 2.15.1 attached.
```

```
##
## Attaching package: 'Rlab'
```

```
## The following objects are masked from 'package:stats':
##
##     dexp, dgamma, dweibull, pexp, pgamma, pweibull, qexp, qgamma,
##     qweibull, rexp, rgamma, rweibull
```

```
## The following object is masked from 'package:datasets':
##
##     precip
```

```
library(fitdistrplus)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:Rlab':
##
##     michelson
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:Rlab':
##
##     cancer
```

```
## Loading required package: npsurv
```

```
## Loading required package: lsei
```

```
berns = c()
for(i in 1:1000){
  berns = append(berns,sum(rbern(9,.4)))
}

#print(descdist(berns))
#dist <- fitdist(berns,"norm")
prob1 = pnorm(mean = 3.593000,sd = 1.436437,q=1)
#0.03552475

prob7 = pnorm(mean = 3.593000,sd = 1.436437,q=Inf) - pnorm(mean = 3.593000,sd = 1.436437,q=6)
#0.04690104
print(prob1+prob7)
```

```
## [1] 0.08242579
```

```
#the sum of bernouli distributions converges to a binomial
print(dbinom(0,9,.4)+dbinom(x=1,size=9,prob = .4)+dbinom(x=7,size=9,prob = .4)+dbinom(x=8,size=9,prob =
```

```
## [1] 0.09557862
```

3 - b

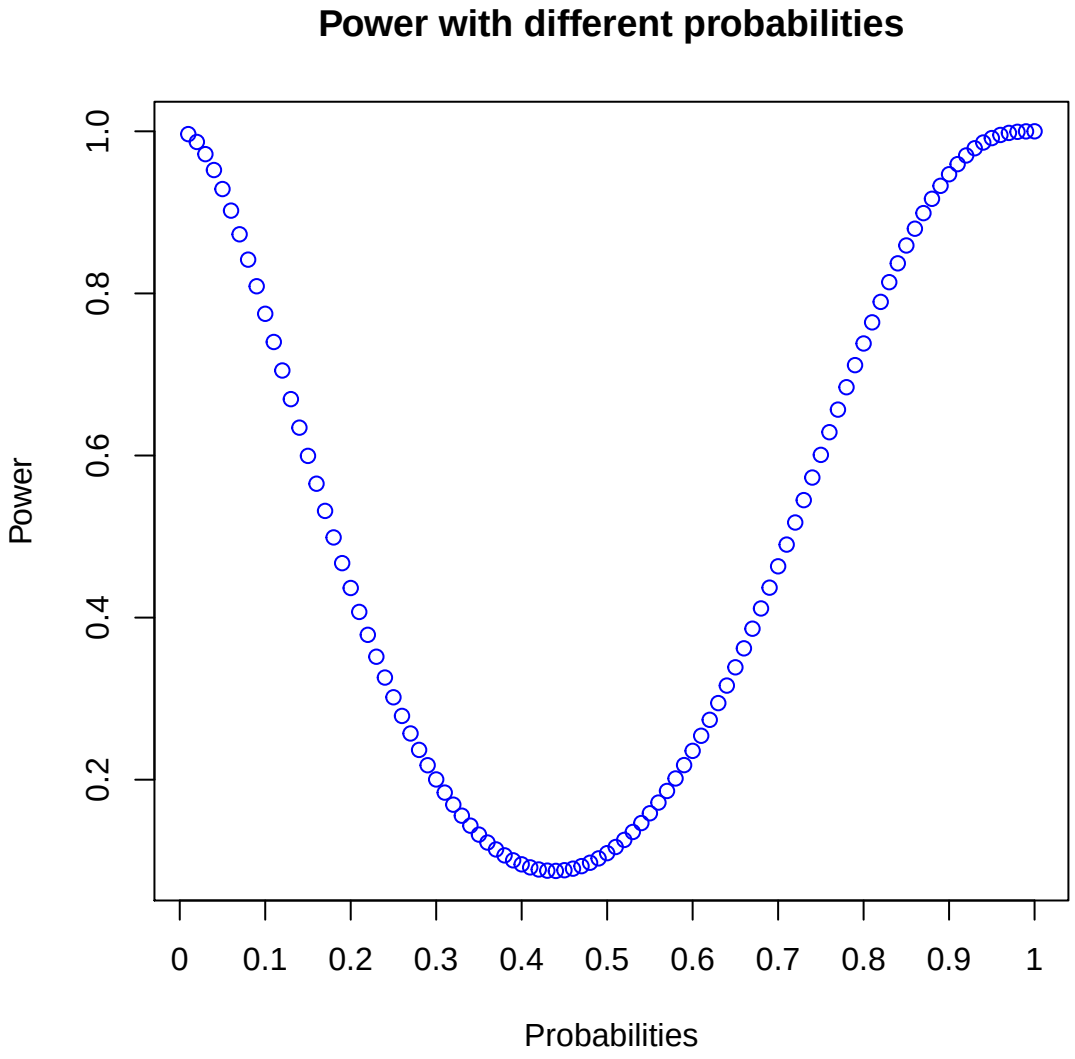
size of test is maximum probability of having a type I error (alpha) *this is the sum of the two probabilities we had in 3 a

```
dbinom(0,9,.4)+dbinom(x=1,size=9,prob = .4)+dbinom(x=7,size=9,prob = .4)+dbinom(x=8,size=9,prob = .4)+d
```

```
## [1] 0.09557862
```

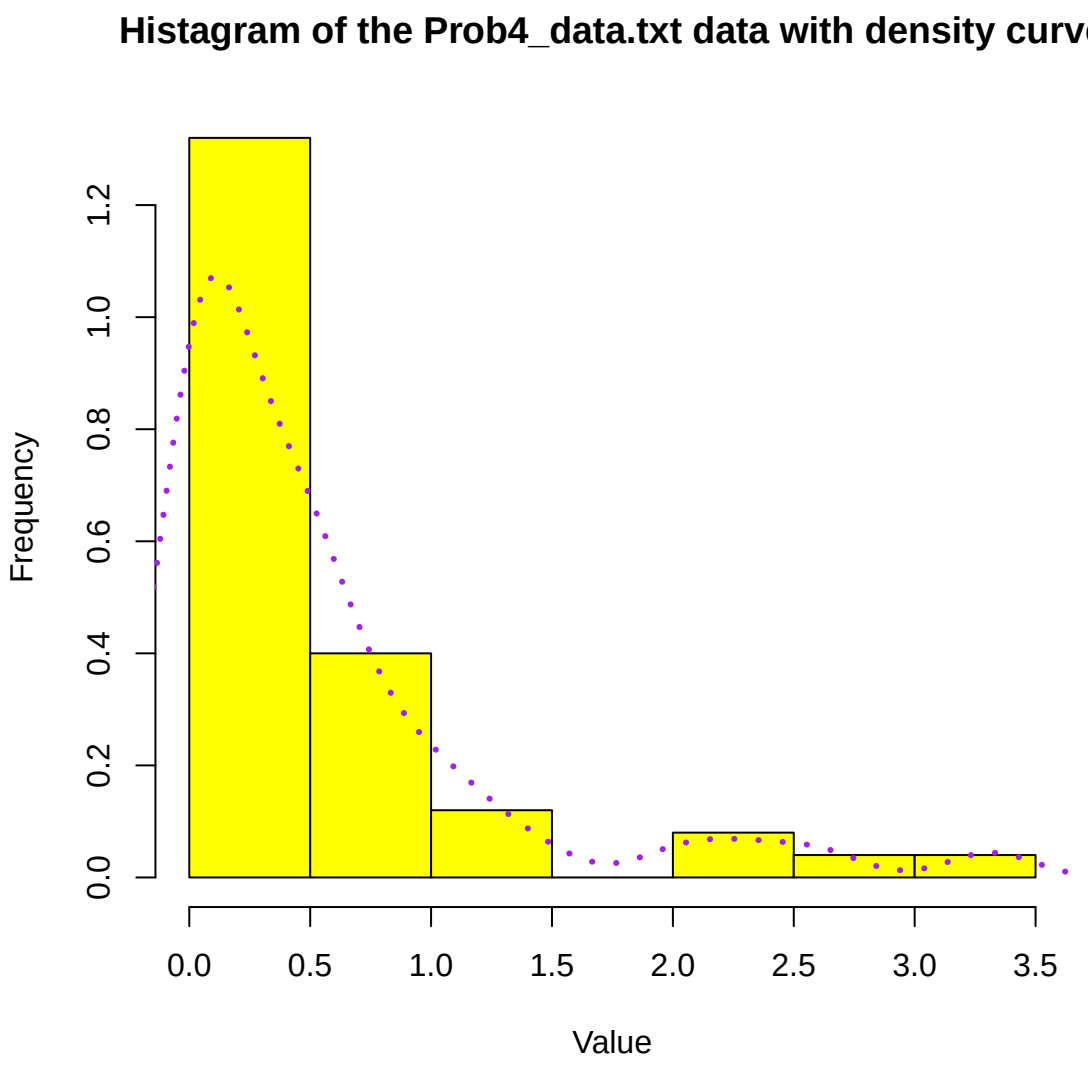
3 - c

```
powers = c()
for(i in seq(.01,1,by=.01)){
  powers = append(powers,dbinom(0,9,i)+dbinom(x=1,size=9,prob = i)+dbinom(x=7,size=9,prob = i)+dbinom(x=
}
plot(powers,main="Power with different probabilities",col='blue',xlab='Probabilities',ylab='Power',xaxt=
axis(1, at=seq(0,100,by=10), labels=seq(0,1,by=.1))
```



4 - a

```
setwd('C:/Users/rossw/Documents/MAE Program/Econometrics 403A/Homework')

data = read.delim("Prob4_data.txt",header = F)

hist(data[,'V1'],prob=T,main='Histagram of the Prob4_data.txt data with density curve',xlab='Value',ylab

lines(density(data[,'V1']),lty='dotted',col='purple',lwd=3)
```



4 - b

E[X] = alpha/beta

E[X^2] = alpha(alpha+1)/ beta^2.

alpha/beta = E[X] = xbar = the mean

alpha/beta^2 = E[X^2] - E[X]^2 = the variance

or alpha(alpha+1)/beta^2 = E[X^2]

now we can solve for alpha and beta give the data

```
#E[X] = alpha/beta
#E[X^2] = alpha(alpha+1)/ beta^2.
#alpha/beta = E[X] = xbar = the mean
#alpha/beta^2 = E[X^2] - E[X]^2 = the variance
# or alpha(alpha+1)/beta^2 = E[X^2]
#now we can solve for alpha and beta give the data
xbar = mean(data['V1'])
n = length(data['V1'])
variance = var(data['V1'])


alpha = xbar^2/variance
beta = xbar/variance
#alpha = xbar^2/((1/n)*sum(xi^2-xbar^2))
#beta = xbar/((1/n)*sum(xi^2-xbar^2))
cat('alpha is', alpha,'\n')
```

```
## alpha is 0.568931
```

```
cat('beta is', beta,'\n')
```

```
## beta is 1.060212
```

```
print(alpha/beta)
```

```
## [1] 0.53662
```

```
print(xbar)
```

```
## [1] 0.53662
```

4 - c

```
alphas = c()
betas = c()
for(i in 1:1000){
  cursamp = sample(data$V1,size = 50,replace = T)
  xbar = mean(cursamp)
  varsamp = var(cursamp)

  alpha = xbar^2/varsamp
  beta = xbar/varsamp

  alphas = append(alphas,alpha)
  betas = append(betas,beta)
}
cat('Bootstrap mean of alpha is',mean(alphas),'\n')
```

```
## Bootstrap mean of alpha is 0.6160656
```

```
cat('Compared to the observed mean which is',0.5805419,'\n')
```

```
## Compared to the observed mean which is 0.5805419
```

```r
print('both are within the confidence interval')
```

```
## [1] "both are within the confidence interval"
```

```r
print('Alpha Bootstrap standard error')
```

```
## [1] "Alpha Bootstrap standard error"
```

```r
sd(alphas)
```

```
## [1] 0.1492564
```

```r
print('Alpha confidence interval')
```

```
## [1] "Alpha confidence interval"
```

```r
quantile(alphas,probs=c(.025,.975))
```

```
##      2.5%     97.5%
## 0.3924725 0.9827953
```

```r
cat('\n')
```

```r
cat('\n')
```

```r
cat('Bootstrap mean of beta is',mean(betas),'\n')
```

```
## Bootstrap mean of beta is 1.182507
```

```r
cat('Compared to the observed mean which is',1.081849,'\n')
```

```
## Compared to the observed mean which is 1.081849
```

```r
print('both are within the confidence interval')
```

```
## [1] "both are within the confidence interval"
```

```r
print('Beta Bootstrap standard error')
```

```
## [1] "Beta Bootstrap standard error"
```

```r
sd(betas)
```

```
## [1] 0.3959385
```

```r
print('Beta confidence interval')
```

```
## [1] "Beta confidence interval"
```

```r
quantile(betas,probs=c(.025,.975))
```

```
##      2.5%     97.5%
## 0.7311756 2.1855083
```

5 - a

```r
temps = c(97.8, 97.2, 97.4,97.6, 97.8, 97.9, 98.0, 98.0, 98.0, 98.1, 98.2, 98.3, 98.3, 98.4, 98.4,
98.4, 98.5, 98.6, 98.6, 98.7,98.8, 98.8, 98.9, 98.9, 99.0)
#H0: mean = 98.6
#H1: mean != 98.6
#mean(temps)
sampvar = sum((temps - mean(temps))^2)/(length(temps)-1)
```

```r
sampsd = sqrt(sampvar)

tstat = (mean(temps)-98.6)/(sampsd/sqrt(length(temps)))
cat('our tstatistic is',tstat,'\n')
```

```
## our tstatistic is -3.484907
```

```r
#tstat = -3.484907
#Degrees of freedom = n-1 = 24
#The P-Value = 0.001912
cat('our p-value is',2*pt(tstat,df = 24),'\n')
```

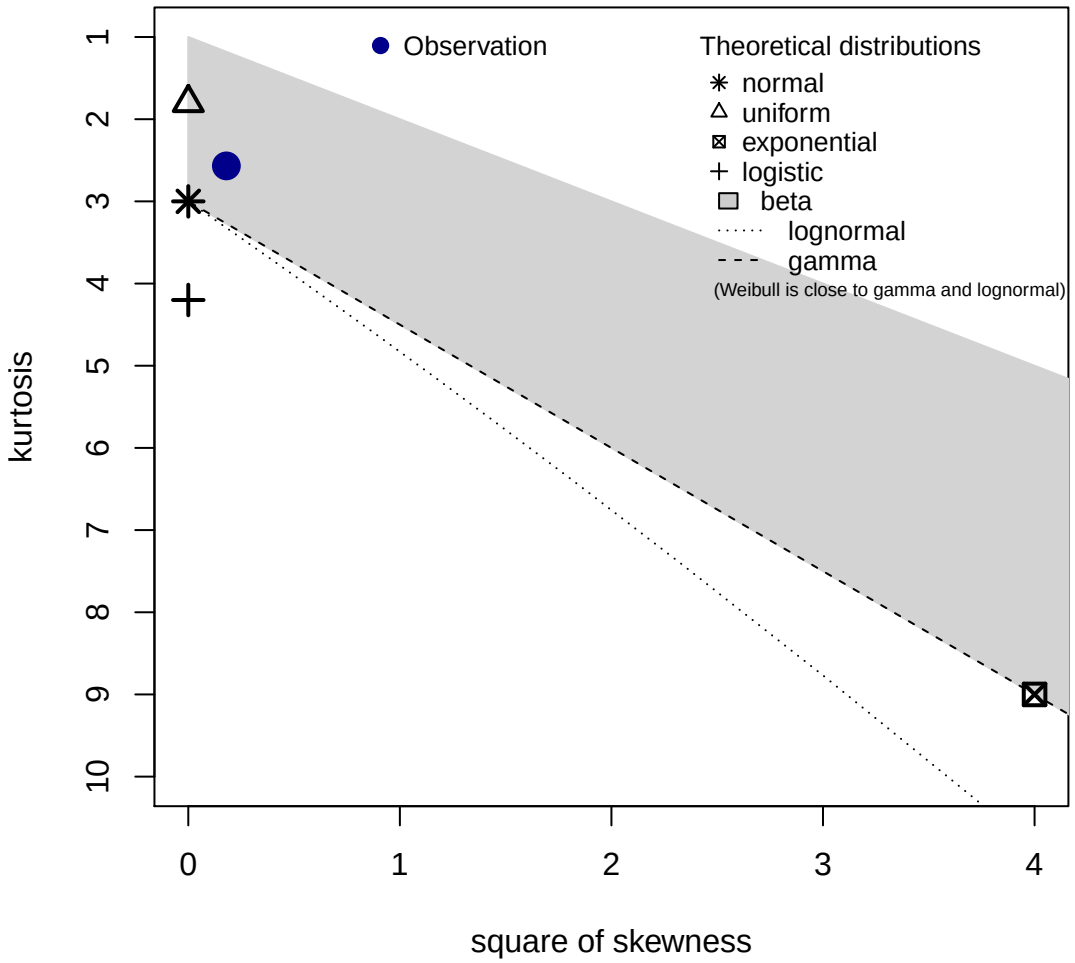
```
## our p-value is 0.001912414
```

```r
t.test(temps,mu=98.6)
```

```
##
##  One Sample t-test
##
## data:  temps
## t = -3.4849, df = 24, p-value = 0.001912
## alternative hypothesis: true mean is not equal to 98.6
## 95 percent confidence interval:
##  98.06501 98.46299
## sample estimates:
## mean of x
##    98.264
```

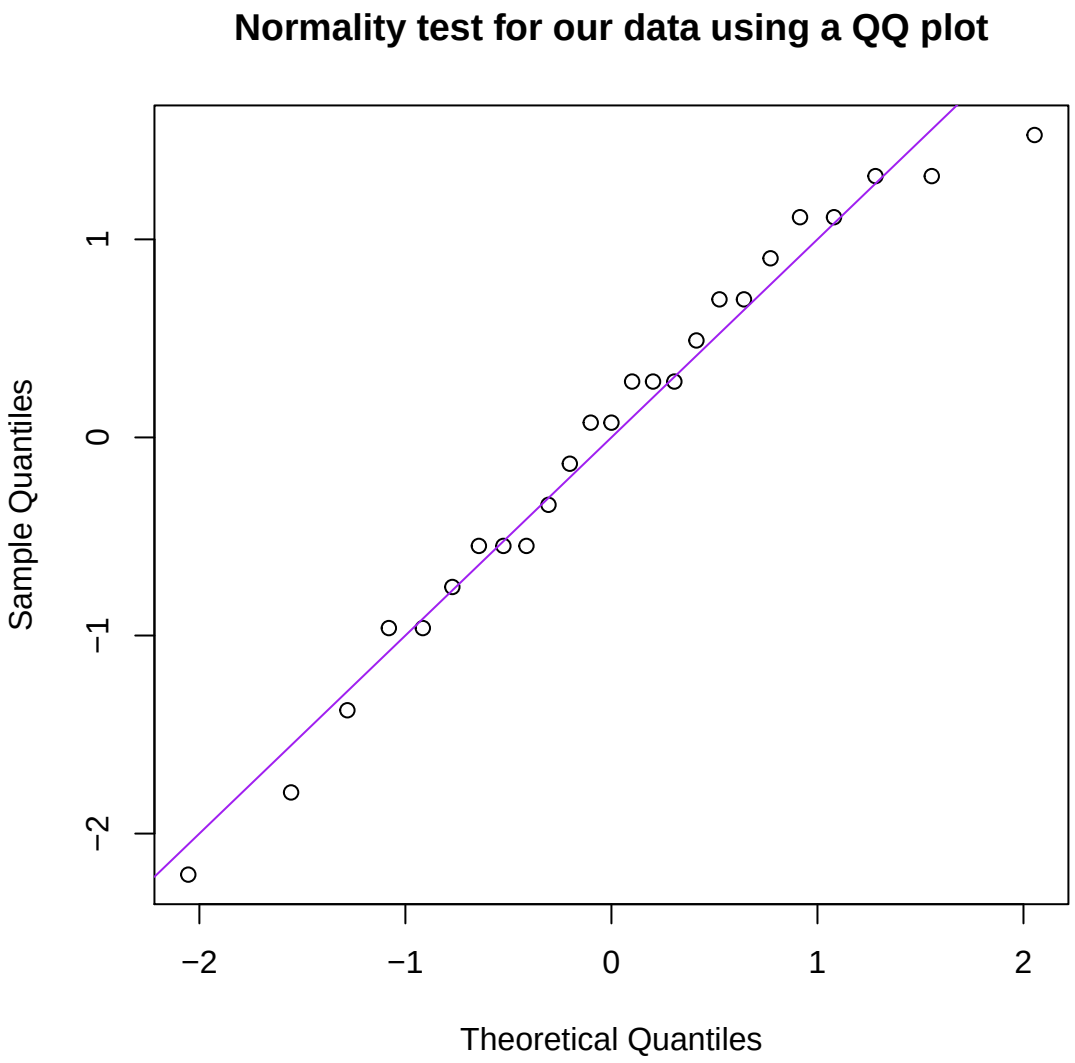
5 - b

```r
temps = c(97.8, 97.2, 97.4,97.6, 97.8, 97.9, 98.0, 98.0, 98.0, 98.1, 98.2, 98.3, 98.3, 98.4, 98.4,
          98.4, 98.5, 98.6, 98.6, 98.7,98.8, 98.8, 98.9, 98.9, 99.0)
#mean(temps)
sampvar = sum((temps - mean(temps))^2)/(length(temps)-1)
sampsd = sqrt(sampvar)
normtest = (temps - mean(temps))/sampsd
#To test for normality, we can look at the Cullen and Frey graph
#As well as the qqnorm plot
descdist(temps)
```




```
## summary statistics
## ------
## min:  97.2   max:  99
## median:  98.3
## mean:  98.264
## estimated sd:  0.4820788
## estimated skewness:  -0.4242381
## estimated kurtosis:  2.568291
```

```r
#Our data is in the beta distribution, but it is close to the normal
qqnorm(normtest,main = 'Normality test for our data using a QQ plot')
#Our data seems to follow a normal
abline(0, 1,col='purple')
```



5 - c

We now think p is 98.6 but it is actually 98.0. We will use a t statistic.

```
# we now think p is 98.6 but it is actually 98.0
#use t statistic

print('The power is')
```

```
## [1] "The power is"
```

```
print(power.t.test(n=25,delta = 98.0 - 98.6,sig.level = .05,sd=sampsd,type = 'one.sample')['power'])
```

```
## $power
## [1] 0.9999683
```

5 - d

Beta is the probability you won't reject the null hypothesis when it is false

```
temps = c(97.8, 97.2, 97.4,97.6, 97.8, 97.9, 98.0, 98.0, 98.0, 98.1, 98.2, 98.3, 98.3, 98.4, 98.4,
98.4, 98.5, 98.6, 98.6, 98.7,98.8, 98.8, 98.9, 98.9, 99.0)

#beta is the probability you won't reject the null hypothesis when it is false

print('The sample size needed is')
```

```
## [1] "The sample size needed is"
```

```
power.t.test(delta = 98.2 - 98.6,sig.level = .05,sd=sd(temps),type = 'one.sample',power = .9)['n']
```

```
## $n
## [1] 17.29755
```

```
print('Or 18 to get a whole number')
```

```
## [1] "Or 18 to get a whole number"
```

5 - e

For a t distribution with a 95% confidence interval and 24 degrees of freedom, our confidence interval will be within 2.064 of the center of the t distribution.

```
temps = c(97.8, 97.2, 97.4,97.6, 97.8, 97.9, 98.0, 98.0, 98.0, 98.1, 98.2, 98.3, 98.3, 98.4, 98.4,
98.4, 98.5, 98.6, 98.6, 98.7,98.8, 98.8, 98.9, 98.9, 99.0)

#for a t distribution with a 95% confidence interval and 24 degrees of freedom, our confidence interval
#will be within 2.064 of the center of the t distribution

sammean = mean(temps)
samvar = sum((temps-sammean)^2)/(length(temps)-1)
samsd = sqrt(samvar)

t = (sammean - 98.6)/(samsd/sqrt(length(temps)))

# so we accept the null hypothesis with 95% confidence if
# -2.064 < t < 2.064

#however, t is -3.484907, so just like in a, we reject the null hypothesis
```

So we accept the null hypothesis with 95% confidence if -2.064 $<$ t $<$ 2.064. However, t is -3.484907, so just like in a, we reject the null hypothesis.

6 - a

A type I error would be noticing a difference when there isn't one. We can assume lambda is 1 and find the probability of getting a y of greater than or equal to 3.2 we can use the exponential function to tell us the probability of getting a y below 3.2.

```
print(pexp(3.2))
```

```
## [1] 0.9592378
```

```
#and subtract 1 from that probability
print(1 - pexp(3.2))
```

```
## [1] 0.0407622
```

```
#0.0407622
```

6 - b

A type II error would be failing to reject the null hypothesis when it is false. So that would be the probability of y being less than 3.2 for an exponential function of rate 3/4 we can use the exponential function to tell us the probability of getting a y below 3.2.

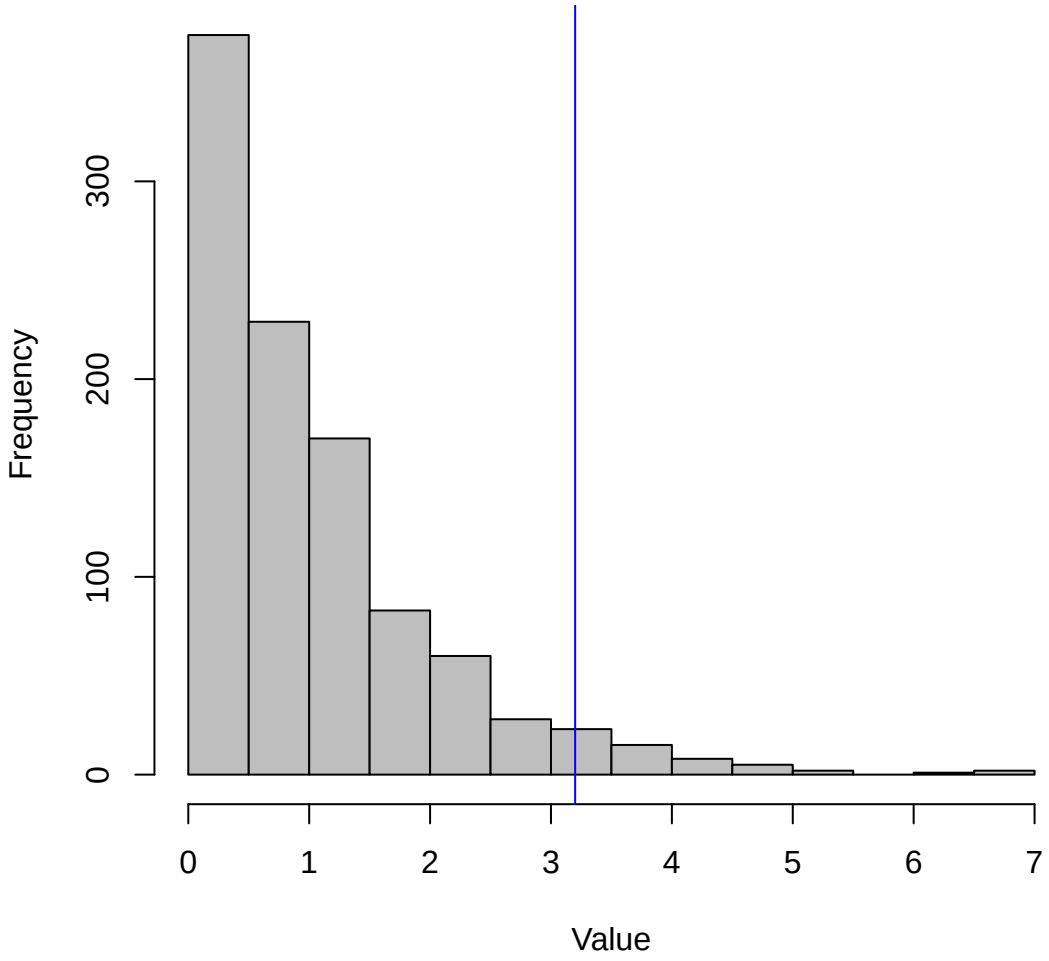
```
print(pexp(3.2,rate = (3/4)))
```

```
## [1] 0.909282
```

6 - c

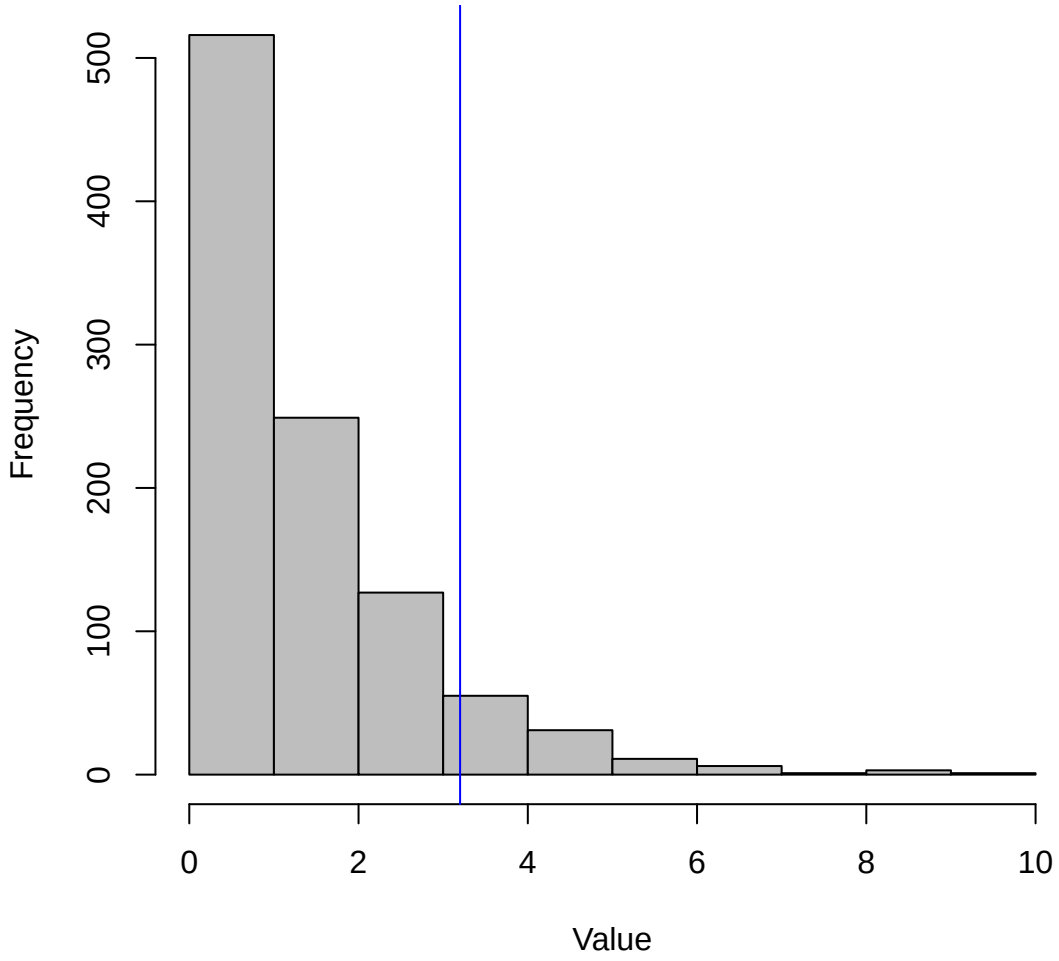
I will take 1000 samples from the exponential distribution. We can also find and plot the integral. The lines show 4.07622% of the samples are to the right of the first line, and 90.9282% of the samples are to the left of the second line.

```
#I will take 1000 samples from the exponential distribution
#We can also find and plot the integral
hist(rexp(1000,rate=1),main="Histogram of 1000 samples from the exponential distribution",
     col='grey',xlab='Value')
abline(v=3.2,col='blue')
```




```
#4.07622% of the samples are to the right of this line

hist(rexp(1000,rate=(3/4)),main="Histogram of 1000 samples from the exponential distribution with rate
     col='grey',xlab='Value')
abline(v=3.2,col='blue')
```

togram of 1000 samples from the exponential distribution with rat




```
#90.9282% of the samples are to the left of this line
```

7 - a

```
#simple random sampling:  sampling without replacement
a = sample(1:10000,500,replace = F)
mean(a)
```

```
## [1] 4733.568
```

7 - b

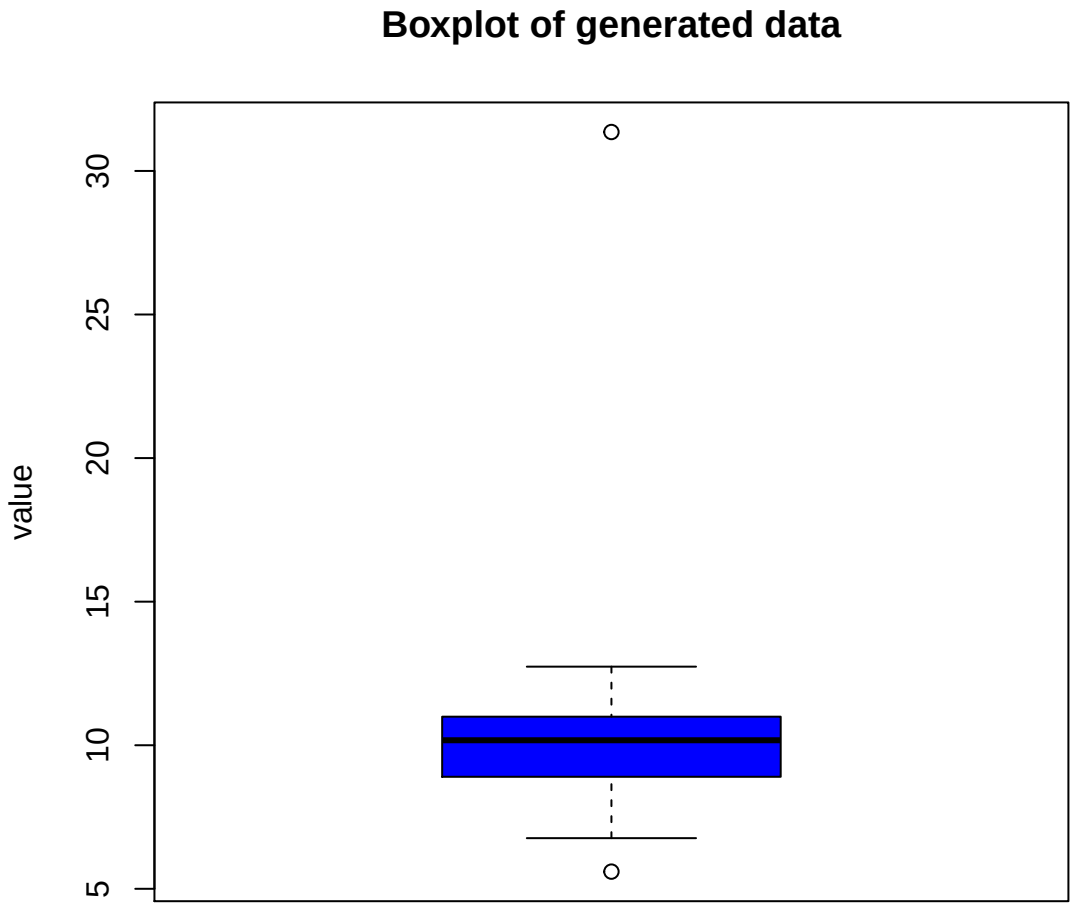
```
#i.i.d sampling:  sampling with replacement
b = sample(1:10000,500,replace = T)
mean(b)
```

```
## [1] 5126.096
```

```
#Alternatively
#i.i.d. sampling:  each sample is independant of the last
#either get every combination of 500 labels from a population of 10000 (choose(500,10000))
#this is not possible
#for this we can sample with replacement
b = sample(1:10000,500,replace = T)
#or another way to do this with a final vector containing unique values is to
#sample with replacement until we have a list of 500 unique labels
b=c()
while(length(b) < 500){
  samp = sample(1:10000,500-length(b),replace=T)
  b = unique(append(b,samp))
}
```

8 - a

```
#a
thirty = rnorm(30,mean=10,sd = 2)
thirtyone = append(thirty,rnorm(1,30,2))
boxplot(thirtyone,main='Boxplot of generated data',ylab='value',col='blue')
```



8 - b

The values from the first normal distribution are all near 10, and the value from the second distribution is marked as an outlier.

8 - c

location = mean? median? spread = variance? standard deviation? interquartile range?

```
print(mean(thirtyone))
```

```
## [1] 10.56673
```

```
print(sd(thirtyone))
```

```
## [1] 4.182964
```

```
print(median(thirtyone))
```

```
## [1] 10.17296
```

```
print(IQR(thirtyone))
```

```
## [1] 2.093024
```

We should use median and interquartile range becuase the outlier at about 30 skews mean and standard deviation median gives us a location closer to 10 and IQR (Q3-Q1) gives us a closer spread than standard deviation.

9

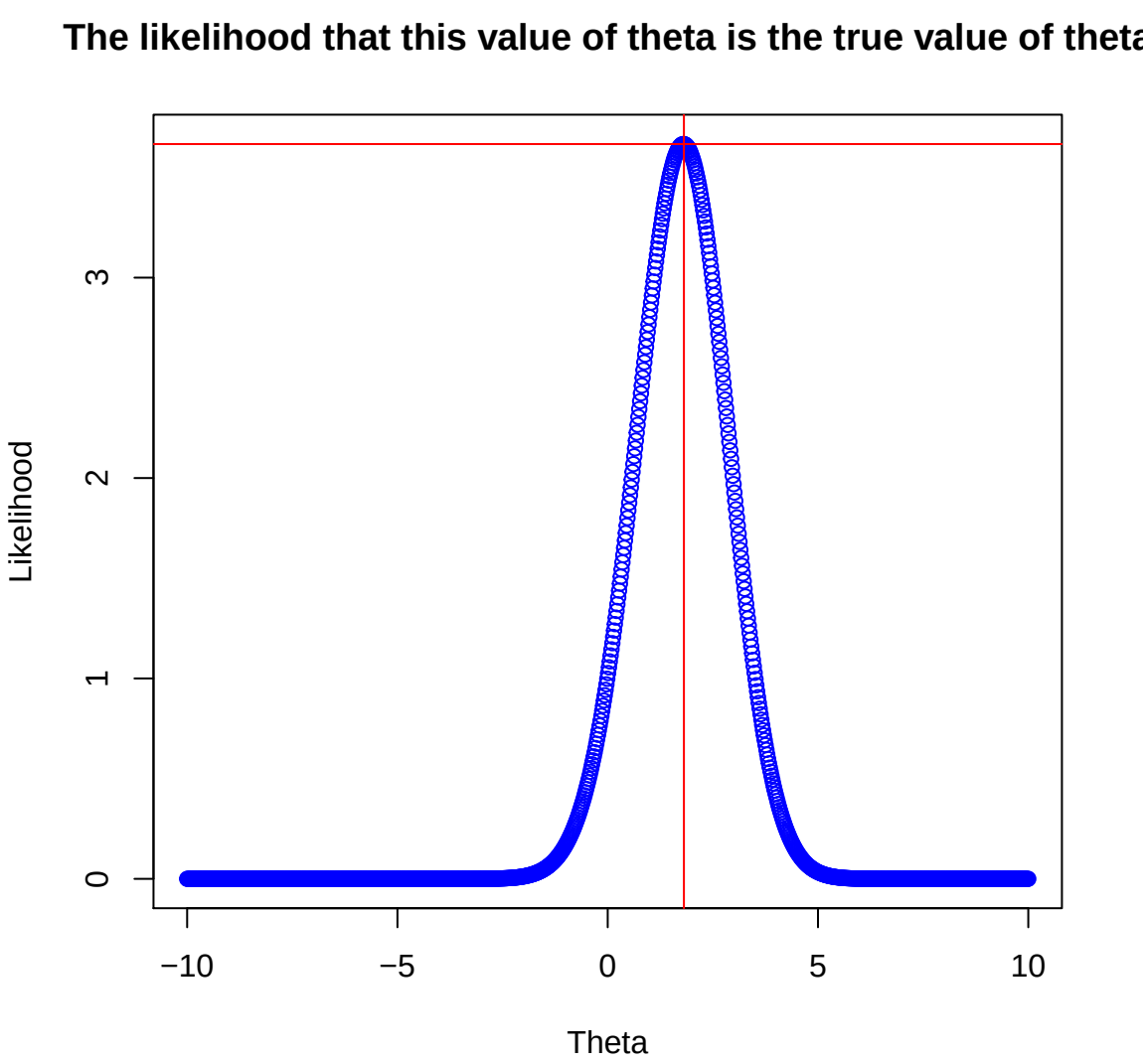
```
maxlikeest = data.frame(seq(-10,10,length.out=1000))

colnames(maxlikeest) = 'theta'
#head(maxlikeest)

likelihood = function(theta){
  return(exp(-(theta - 1)^2/2) + 3*exp(-(theta - 2)^2/2))
}

maxlikeest$eval = likelihood(maxlikeest[,'theta'])
maximum = max(maxlikeest$eval)
cat('The max value is',maximum,'\n')
```

```
## The max value is 3.666615
```

```
cat('and theta at that value is',maxlikeest[which(maxlikeest$eval == maximum),'theta'],'\n')
```

```
## and theta at that value is 1.811812
```

```
plot(maxlikeest,main='The likelihood that this value of theta is the true value of theta',xlab='Theta',y
abline(h = maximum,col='red')
abline(v = maxlikeest[which(maxlikeest$eval == maximum),'theta'],col='red')
```



10 - a

L(theta) = theta$^{4(1\text{-theta})}$16

l(theta) = 4*ln(theta) + 16*ln(1-theta)

d/dtheta(l(theta)) = 4/theta - 16/(1-theta)

d/dtheta(l(theta)) = 0 = 1/theta - 4/(1-theta)

4/(1-theta) = 1/theta

4theta = 1-theta

5theta = 1

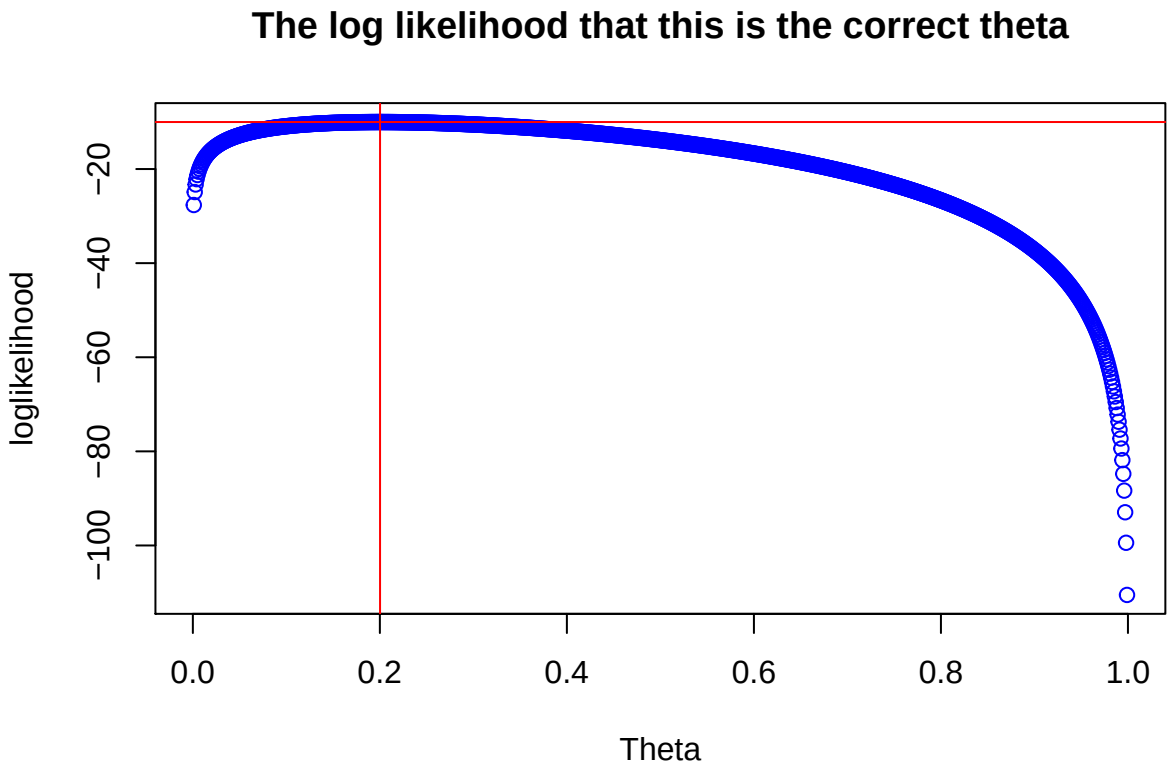
theta = 1/5

```
loglike = function(theta){
  return(4*log(theta) + 16*log(1-theta))
}
```

```r
maxlikeest = data.frame(seq(0,1,length.out=1000))
colnames(maxlikeest) = 'theta'

maxlikeest$eval = loglike(maxlikeest[,'theta'])
plot(maxlikeest,main='The log likelihood that this is the correct theta',xlab='Theta',ylab='loglikeliho

maximum = max(maxlikeest$eval)
print( maxlikeest[which(maxlikeest$eval == maximum),'theta'])
```

```
## [1] 0.2002002
```

```r
abline(h = maximum,col='red')
abline(v = maxlikeest[which(maxlikeest$eval == maximum),'theta'],col='red')
```



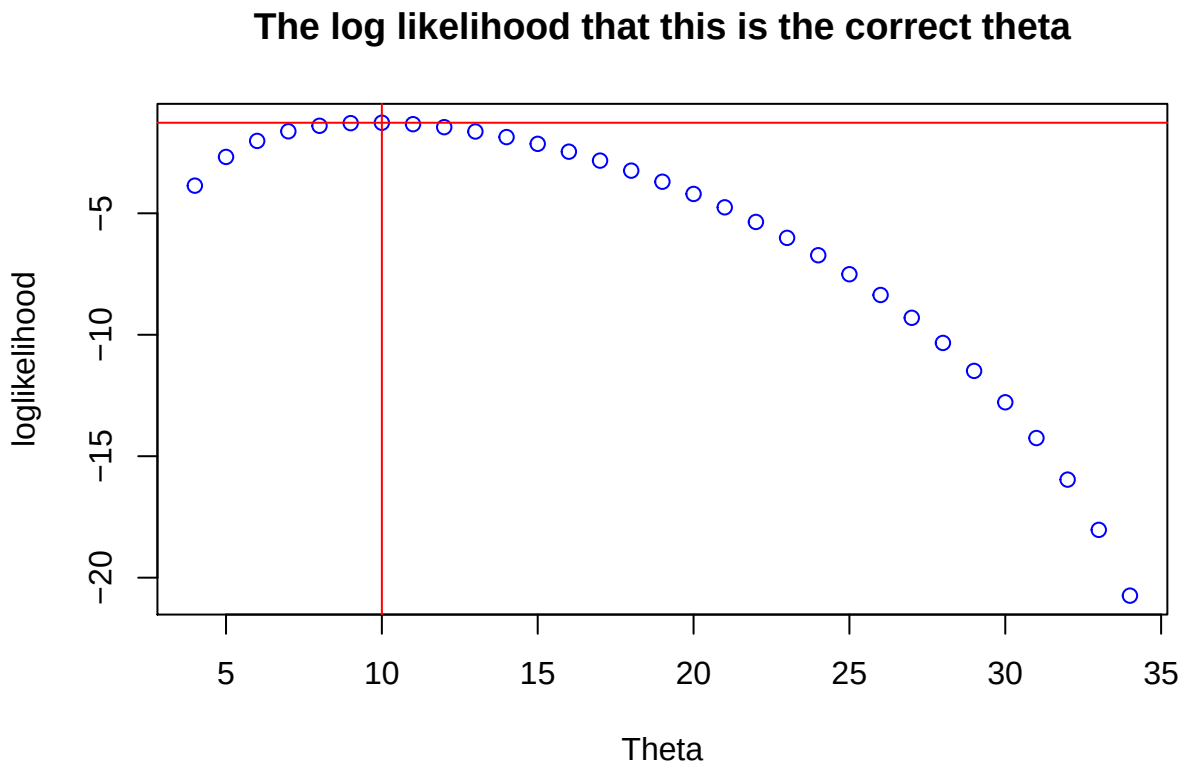
10 - b

```r
fun = function(theta){
  return(log((choose(theta,4)*choose(50-theta,20-4))/choose(50,20)))
}

maxlikeest = data.frame(seq(4,34,by = 1))
colnames(maxlikeest) = 'theta'

maxlikeest$eval = fun(maxlikeest[,'theta'])
plot(maxlikeest,main='The log likelihood that this is the correct theta',xlab='Theta',ylab='loglikeliho
```

```
maximum = max(maxlikeest$eval)
print( maxlikeest[which(maxlikeest$eval == maximum),'theta'])
```

```
## [1] 10
```

```
abline(h = maximum,col='red')
abline(v = maxlikeest[which(maxlikeest$eval == maximum),'theta'],col='red')
```




```
cat('MLE is 10/50 or .2')
```

```
## MLE is 10/50 or .2
```

11

```
meaninterval5 = c()
n = 5
for(i in 1:1000){
  cursamp = rnorm(n,0,1)
  curmean = mean(cursamp)
  curvar = sum((cursamp - curmean)^2)/(length(cursamp)-1)
  cursd = sqrt(curvar)
  if(curmean-(cursd/sqrt(n)) < 0 & curmean+(cursd/sqrt(n)) > 0){
    meaninterval5 = append(meaninterval5,1)
  }else{
    meaninterval5 = append(meaninterval5,0)
  }
```

```
}
cat('the proportion of sample samples of size 5 that have a mean that lie within the confidence interval
```

```
## the proportion of sample samples of size 5 that have a mean that lie within the confidence interval
```

```
meaninterval10 = c()
n = 10
for(i in 1:1000){
  cursamp = rnorm(n,0,1)
  curmean = mean(cursamp)
  curvar = sum((cursamp - curmean)^2)/(length(cursamp)-1)
  cursd = sqrt(curvar)
  if(curmean-(cursd/sqrt(n)) < 0 & curmean+(cursd/sqrt(n)) > 0){
    meaninterval10 = append(meaninterval10,1)
  }else{
    meaninterval10 = append(meaninterval10,0)
  }
}
cat('the proportion of sample samples of size 10 that have a mean that lie within the confidence interva
```

```
## the proportion of sample samples of size 10 that have a mean that lie within the confidence interval
```

```
meaninterval100 = c()
n = 100
for(i in 1:1000){
  cursamp = rnorm(n,0,1)
  curmean = mean(cursamp)
  curvar = sum((cursamp - curmean)^2)/(length(cursamp)-1)
  cursd = sqrt(curvar)
  if(curmean-(cursd/sqrt(n)) < 0 & curmean+(cursd/sqrt(n)) > 0){
    meaninterval100 = append(meaninterval100,1)
  }else{
    meaninterval100 = append(meaninterval100,0)
  }
}
cat('the proportion of sample samples of size 100 that have a mean that lie within the confidence inter
```

```
## the proportion of sample samples of size 100 that have a mean that lie within the confidence interval
```

```
print('The proporitons are similar, only increasing slightly as n increases.')
```

```
## [1] "The proporitons are similar, only increasing slightly as n increases."
```

12

```
data = c(3.27, -1.24, 3.97, 2.25, 3.47, -0.09, 7.45, 6.20, 3.74, 4.12, 1.42, 2.75, -1.48, 4.97, 8.00, 3

n=length(data)

pluginmlemu = mean(data)
pluginmlesd = sd(data)

val = pnorm(3,mean=pluginmlemu,sd=pluginmlesd)
cat('Our estimate for F(3):',val,'\n')
```

```
## Our estimate for F(3): 0.5133075
```

```
mses1000 = c()
for (i in 1:1000){
  cursamp = sample(data,20,replace = T)
  mses1000 = append(mses1000,(val - pnorm(3,mean=mean(cursamp),sd=sd(cursamp)))^2)
}
cat('Mean of 1000 bootstraped squared errors: ', mean(mses1000),'\n')
```

```
## Mean of 1000 bootstraped squared errors:  0.008221095
```

```
mses10000 = c()
for (i in 1:10000){
  cursamp = sample(data,20,replace = T)
  mses10000 = append(mses10000,(val - pnorm(3,mean=mean(cursamp),sd=sd(cursamp)))^2)
}
cat('Mean of 10000 bootstraped squared errors:',  mean(mses10000),'\n')
```

```
## Mean of 10000 bootstraped squared errors: 0.008329659
```