HW3

Hannah Zmuda

12/04/2020

Question 1

Problem

Use Monte Carlo simulation to investigate whether the empirical Type I error rate of the t-test is approximately equal to the nominal significance level α, when the sampled population is non-normal. The t-test is robust to mild departures from normality. Discuss the simulation results for the cases where the sampled population is (i) $\chi^2(1)$, (ii) Uniform(0,2), and (iii) Exponential(rate=1). In each case, test H0 : $\mu = \mu_0$ vs H0 : $\mu \neq \mu_0$, where μ_0 is the mean of $\chi^2(1)$, Uniform(0,2), and Exponential(1), respectively.

Solution

```
library(ggplot2)
#Part a: looking at a type I error
#all follow a similar algorithm as shown on page 193 in SCRR
n <- 100 #number of replicates
a <- 0.05 #significance level alpha
muA <- mean(rchisq(n, df = 1)) #mu0 in part a
muB <- mean(runif(n, 0, 2)) #mu0 in part b
muC <- mean(rexp(n, rate = 1)) #mu0 in part c
#become alternatives in the estimate power of a test (b).

m <- 1000 #number of replicates
pA <- numeric(m)
pB <- numeric(m)
pC <- numeric(m)

for(i in 1:m){
  xA <- rchisq(n, df = 1) #sample dist part a
  xB <- runif(n, 0, 2) #sample dist part b
  xC <- rexp(n, rate = 1) #sample dist part c

  ttestA <- t.test(xA, alternative = "two.sided",mu = muA)
  ttestB <- t.test(xB, alternative = "two.sided",mu = muB)
  ttestC <- t.test(xC, alternative = "two.sided",mu = muC)

  pA[i] <- ttestA$p.value
  pB[i] <- ttestB$p.value
  pC[i] <- ttestC$p.value
}
```

```r
pHatA <- mean(pA <= a)
pHatB <- mean(pB <= a)
pHatC <- mean(pC <= a)

seHatA <- sqrt(pHatA * (1- pHatA)/m)
seHatB <- sqrt(pHatB * (1- pHatB)/m)
seHatC <- sqrt(pHatC * (1- pHatC)/m)
#Means:
cat("The means (of the p-values) are: ", c(pHatA, pHatB, pHatC), "\n")
```

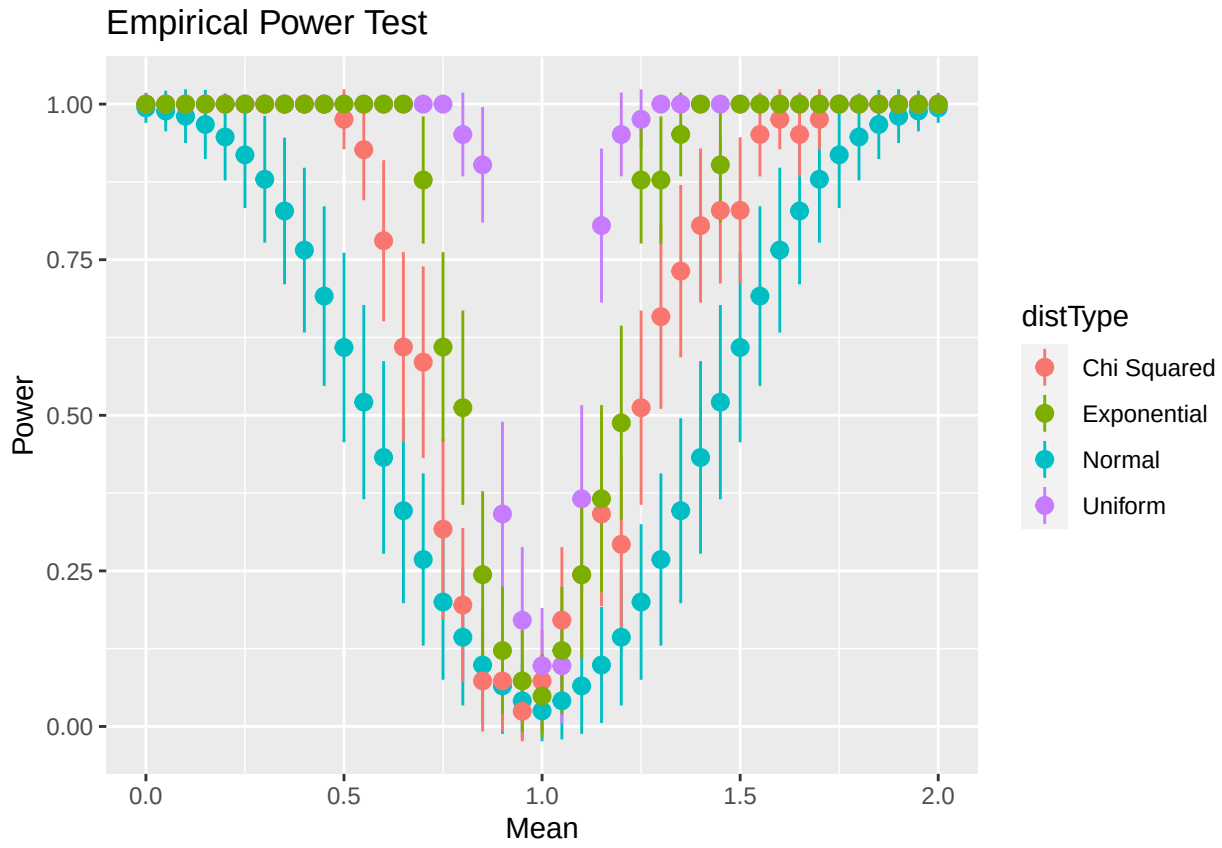
```
## The means (of the p-values) are:  0.108 0.051 0.342
```

```r
#SE:
cat("The standard errors (of the p-values) are: ", c(seHatA, seHatB, seHatC))
```

```
## The standard errors (of the p-values) are:  0.00981509 0.006956939 0.0150012
```

```r
#Part b: Estimating power of a test and outputting empirical power curves of the t-test from the three
#initial data array
mu1 <- c(seq(0,2,1/20))
MC <- length(mu1)
powA <- numeric(MC)
powB <- numeric(MC)
powC <- numeric(MC)
powNorm <- numeric(MC)

#select theta_1 from the parameter subspace
#muA, muB, and muC values
#set for loop
for(j in 1:MC){
  #Chi Squared Distribution
  pvalA <- replicate(MC, expr = {
    xA <- rchisq(n, df = 1) #sample dist part a
    ttestA <- t.test(xA, alternative = "two.sided",mu = mu1[j])
    ttestA$p.value
  })
  powA[j] <- mean(pvalA <= a)
  #Uniform Distribution
  pvalB <- replicate(MC, expr = {
    xB <- runif(n, 0, 2) #sample dist part b
    ttestB <- t.test(xB, alternative = "two.sided",mu = mu1[j])
    ttestB$p.value
  })
  powB[j] <- mean(pvalB <= a)
  #Exponential Distribution
  pvalC <- replicate(MC, expr = {
    xC <- rexp(n, rate = 1) #sample dist part c
    ttestC <- t.test(xC, alternative = "two.sided",mu = mu1[j])
    ttestC$p.value
  })
  powC[j] <- mean(pvalC <= a)
  #Normal Distribution using power.t.test function
```

```r
  powerTest <- power.t.test(n = MC, delta = mu1[j]-1,sig.level = a,alternative = "two.sided")
  powNorm[j] <- powerTest$power

}

#making of the data frame
mean <- c(mu1,mu1,mu1,mu1)
powerR <- c(powNorm, powA, powB, powC)
distType <- c(replicate(MC, "Normal"),replicate(MC, "Chi Squared"),replicate(MC, "Uniform"),replicate(MC

data <- data.frame(mean, powerR,distType)

ggplot(data = data, aes(x = mean, y = powerR, color = distType)) +
  geom_point() +
  labs(x = 'Mean', y = 'Power', title = 'Empirical Power Test') +
  geom_pointrange(data = data, mapping = aes(x = mean, ymin = powerR - 2*(sqrt(powerR * (1- powerR)/MC)
```



Downside of the t-test for non-normal distributions is that it is not as accurate as it is for normal distributions. For the mean of the p-values, all are above the accepted p-value of 0.05 and have empirical power curves skewed away from the power curve of the normal distribution.

Question 2

Part a and b: MC and IS

For part (a) and (b), we are comparing Monte Carlo method and Importance Sampling (IS) to estimate α using the Z-test. In order to do Importance Sampling in part (b), we must first derive the weights: $f(X)/g(X)$. The problem statement tells us that the function $g(x) = Pois(1.5\lambda)$, which means the importance function, $f(x)$, is also based on the Poisson pdf. Therefore,

$$f(x) = \frac{e^{\lambda}\lambda^{x}}{x!}$$

and

$$g(x) = \frac{e^{-1.5\lambda}1.5\lambda^{x}}{x!}$$

```
set.seed(475)
#part a estimate alpha using MCEM
n <- 10 # distribution size
m <- 100 #Monte Carlo sample size
l <- 2 #lambda is equal to 2
#z test with Monte Carlo
ztest <- replicate(m, expr = {
  x <- rpois(n, l) #samples
  ztest <- (mean(x)-2)/(sd(x)/sqrt(n))
})
#alpha estimate
aMC <- mean(ztest > 2.326)
cat("Alpha estimate using MC:", aMC, "\n")
```

```
## Alpha estimate using MC: 0.01
```

```
#part b: estimate alpha using Importance Sampling
g <- function(x) (exp(-1.5*l)*((1.5*l)^x))/factorial(x)
f <- function(x) (exp(-l)*(l^x))/factorial(x)
phi <- function(x) as.numeric((mean(x)-2)/(sd(x)/sqrt(n))  > 2.326)
weight <- function(x) f(x) / g(x)
out <- numeric(m)

out <- replicate(m, expr = {
  x <- rpois(n, l) #samples
  out <- phi(x)*weight(x)
})

isOut <- mean(out)
cat("Alpha estimate using IS:", isOut,"\n")
```

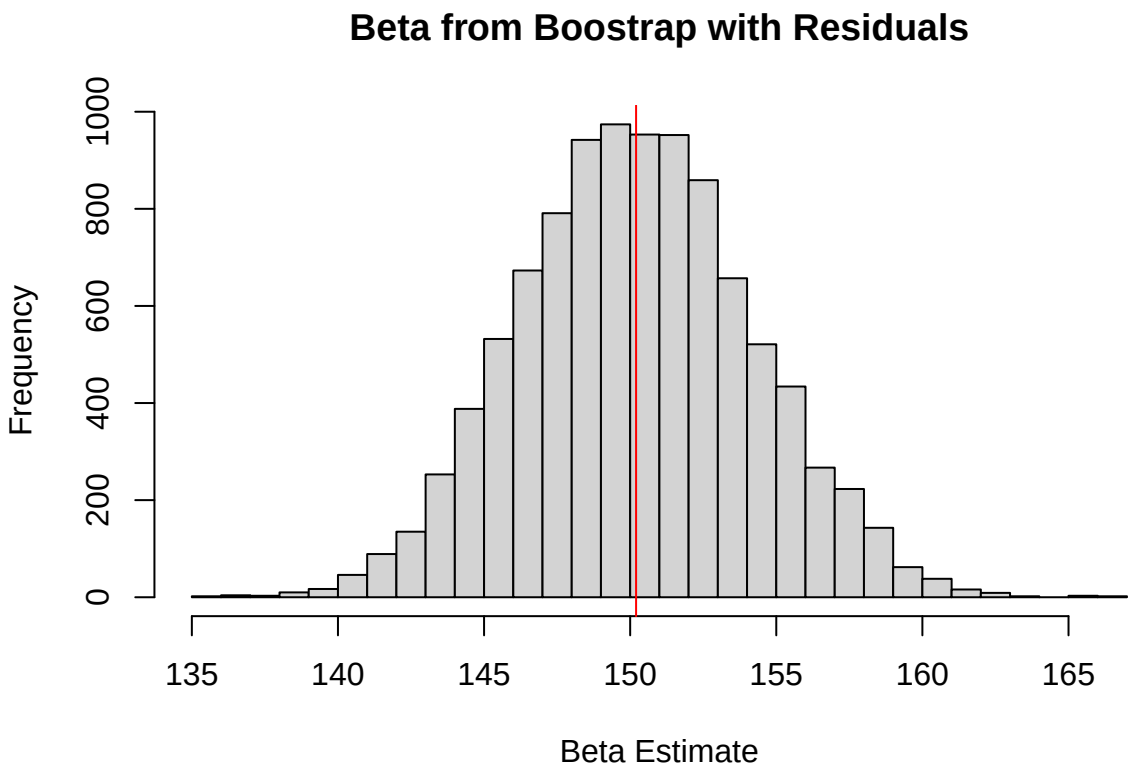
```
## Alpha estimate using IS: 0.008561283
```

Part c: Which is better?

Both outputs of the function are off from the expected 0.05 estimate of α. I would argue that the Monte Carlo is **easier** to implement because the weights do not need to be derived. However, using the weights in the Importance Sampling corrects any bias we might see in the method using Monte Carlo and guarantees and unbiased estimator. Becuase of this, I would choose to use Importance Sampling over Monte Carlo.

Question 3

```r
set.seed(475)
#get data
salmon <-read.table("salmon.dat", header = TRUE)
#initialize values
n <- length(salmon$recruits)
y <- 1/salmon$recruits
x <- 1/salmon$spawners
B <- 10000
beta.hat <- rep(0,B)
#get the linear model and coefficients and first set of residuals
linear.model <- lm(y~x)
#calculate coefficients (slope and intercept)
r <- linear.model$coefficients[1] + (linear.model$coefficients[2] * x)
beta <- (1-linear.model$coef[2])/linear.model$coef[1]
eps <- y - r#residual error

#Bootstrapping the residuals
for(b in 1:B){
  eps.new <- eps[sample(1:n,n,replace = TRUE)]
  yB <- eps.new + r
  fit <- lm(yB ~ x)
  beta.hat[b] <- (1-fit$coef[2])/fit$coef[1]
}
#output
hist(beta.hat,breaks = 25, main = "Beta from Boostrap with Residuals", xlab = "Beta Estimate")
abline(v = mean(beta.hat), col = "red")
```




```
print("BOOTSTRAP THE RESIDUALS")
```

```
## [1] "BOOTSTRAP THE RESIDUALS"
```

```
print("95% CI Beta estimate from Boostrap with Residuals")
```

```
## [1] "95% CI Beta estimate from Boostrap with Residuals"
```

```
quantile(beta.hat,probs = c(0.025,0.975),na.rm=T)
```

```
##     2.5%    97.5%
## 142.6342 158.1336
```

```
cat("Standard Error for Bootstrap:", sd(beta.hat)/n,"\n")
```

```
## Standard Error for Bootstrap: 0.09970293
```

```
#part b: Jackknife after bootstrap
se.beta <- rep(0,B)
beta.jack <- rep(0,B)
for(b in 1:B){
  j <- sample(1:n,n,replace = TRUE)
```

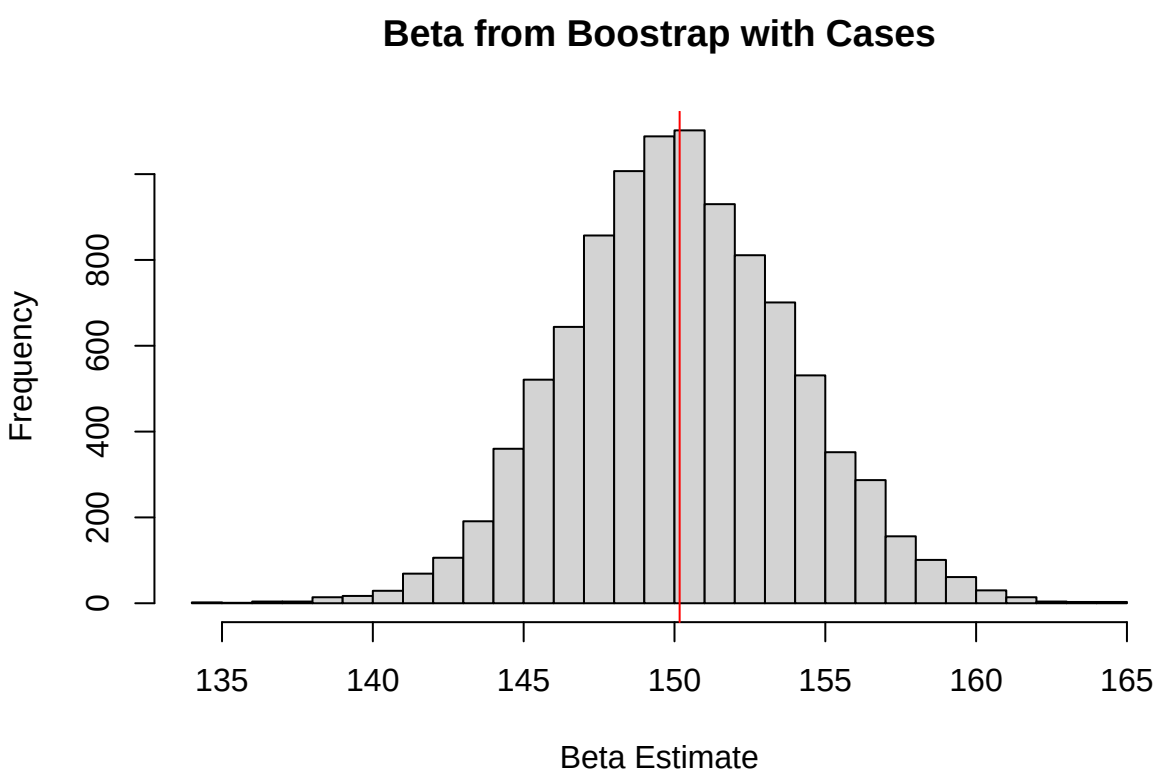
```
  yJ <- eps[-j] + r[-j]
  model.new <- lm(yJ~x[-j])
  beta.jack[b] <- (1-model.new$coef[2])/model.new$coef[1]
}
beta.jack.bar <- mean(beta.jack)
se.beta.jack <- sqrt(((n-1)/n)*sum(beta.jack - beta.jack.bar)^2)
cat("Bias Corrected Estimate:", n*beta - (n-1)*beta.jack.bar,"\n")
```

```
## Bias Corrected Estimate: 157.331
```

```
cat("SE from Jackknife-after-bootstrap:", se.beta.jack)
```

```
## SE from Jackknife-after-bootstrap: 1.303161e-10
```

```
#Bootstrapping the Cases
set.seed(475)
#get data
salmon <-read.table("salmon.dat", header = TRUE)
x <- 1/(salmon$spawners)
y <- 1/(salmon$recruits)
n <- length(x)
B <- 10000
beta.new <- rep(0,B)
beta.jack <- rep(0,B)
model.new <- lm(y ~ x)
beta.true.c <- as.numeric((1-model.new$coef[2])/model.new$coef[1])
for(b in 1:B){
  j <- sample(1:n,n,replace = TRUE)
  x.new <- x[j]
  y.new <- y[j]
  model.new <- lm(y.new~x.new)
  beta.new[b] <- (1-model.new$coef[2])/model.new$coef[1]
}
hist(beta.new,breaks = 25, main = "Beta from Boostrap with Cases", xlab = "Beta Estimate")
abline(v = mean(beta.new), col = "red")
```




```
ci.95.case <- quantile(beta.new,probs = c(0.025,0.975))
print("95% CI Beta estimate from Boostrap with Cases")
```

```
## [1] "95% CI Beta estimate from Boostrap with Cases"
```

```
ci.95.case
```

```
##     2.5%    97.5%
## 143.0399 157.7360
```

```
cat("Standard Error for Bootstrap:", sd(beta.hat)/n,"\n")
```

```
## Standard Error for Bootstrap: 0.09970293
```

```
#part b: Jackknife after bootstrap
se.beta <- rep(0,B)
m.beta <- rep(0,B)
for(b in 1:B){
  j <- sample(1:n,n,replace = TRUE)
  model.new <- lm(y[-j]~x[-j])
  beta.jack[b] <- (1-model.new$coef[2])/model.new$coef[1]
}

beta.jack.bar <- mean(beta.jack)
```

```r
se.beta.jack <- sqrt(((n-1)/n)*sum(beta.jack - beta.jack.bar)^2)
cat("Bias Corrected Estimate:", n*beta.true.c - (n-1)*beta.jack.bar,"\n")
```

```
## Bias Corrected Estimate: 157.331
```

```r
cat("SE from Jackknife-after-bootstrap:", se.beta.jack)
```

```
## SE from Jackknife-after-bootstrap: 1.303161e-10
```

Comparing the two histograms, although they have the mean of the distribution in similar locations, the histogram from the residuals has a flattened top and is slightly more spread on the top compared to the histogram from bootstrapping the cases. Moreover, the Bias Corrected estimate from the Bootstrap the residuals is lower than the actual mean value and the Bias corrected estimate from bootstrap the cases (which is closer to the mean of the histogram).

Question 4

part a

```
set.seed(475)
#get data
cancer <- read.table("cancersurvival.dat", header = TRUE) #1 is stomach cancer, 2 is breast cancer
cancer[,1] <- log(cancer[,1])
breast.cancer <- cancer[cancer[,2]==2,1]
stomach.cancer <- cancer[cancer[,2]==1,1]
#initialize
n <- length(breast.cancer)
B <- 10000
theta.b <- NULL
thetas.b <- rep(0,B)
theta.hat.b <- rep(0,n)
theta.s <- NULL
thetas.s <- rep(0,B)
theta.hat.s <- rep(0,n)
psi.b <- rep(0,n)
psi.s <- rep(0,n)

#Original estimates
theta.b <- mean(breast.cancer)
theta.s <- mean(stomach.cancer)

#Bootstrap!!
for(i in 1:B){
  rand.sample <- sample(1:n,n,replace = TRUE)
  cancer.new.s <- stomach.cancer[rand.sample]
  cancer.new.b <- breast.cancer[rand.sample]
  thetas.b[i] <- mean(cancer.new.b)
  thetas.s[i] <- mean(cancer.new.s)
}
ci.95.b <- quantile(thetas.b,c(0.025,0.975),na.rm=T)
ci.95.s <- quantile(thetas.s,c(0.025,0.975),na.rm=T)

#BCA
for(j in 1:n){
  theta.hat.b[j] <- mean(breast.cancer[-j])
  theta.hat.s[j] <- mean(stomach.cancer[-j])
}
for(k in 1:n){
  psi.b[k] <- mean(theta.hat.b[-k])-theta.hat.b[k]
  psi.s[k] <- mean(theta.hat.s[-k])-theta.hat.s[k]
}
a.b <- ((1/6)*sum(psi.b^3))/((sum(psi.b^2))^(3/2))
a.s <- ((1/6)*sum(psi.s^3))/((sum(psi.s^2))^(3/2))
b.b <- qnorm(mean(thetas.b<theta.b),0,1)
b.s <- qnorm(mean(thetas.s<theta.s),0,1)
beta1.b <- pnorm(b.b + (b.b+qnorm(.025,0,1))/(1-a.b*(b.b+qnorm(.025,0,1))))
beta2.b <- pnorm(b.b + (b.b+qnorm(.975,0,1))/(1-a.b*(b.b+qnorm(.975,0,1))))
beta1.s <- pnorm(b.s + (b.s+qnorm(.025,0,1))/(1-a.s*(b.s+qnorm(.025,0,1))))
```

```r
beta2.s <- pnorm(b.s + (b.s+qnorm(.975,0,1))/(1-a.s*(b.s+qnorm(.975,0,1))))
ci.95.bc.b = quantile(thetas.b,c(beta1.b,beta2.b),na.rm=T)
ci.95.bc.s = quantile(thetas.s,c(beta1.s,beta2.s),na.rm=T)
#output
theta.b #observed estimate theta
```

```
## [1] 6.558603
```

```r
print("95% Confidence Interval for Breast Cancer (Basic Bootstrap)")
```

```
## [1] "95% Confidence Interval for Breast Cancer (Basic Bootstrap)"
```

```r
ci.95.b #95% ci from basic bootstrap
```

```
##     2.5%    97.5%
## 5.537453 7.396224
```

```r
print("95% Confidence Interval for Breast Cancer (Basic BCa)")
```

```
## [1] "95% Confidence Interval for Breast Cancer (Basic BCa)"
```

```r
ci.95.bc.b #95% ci from BCA
```

```
## 0.9418057%  95.09835%
##   5.329144   7.286088
```

```r
print("95% Confidence Interval for Stomach Cancer (Basic Bootstrap)")
```

```
## [1] "95% Confidence Interval for Stomach Cancer (Basic Bootstrap)"
```

```r
theta.s #observed estimate theta
```

```
## [1] 4.96792
```

```r
ci.95.s #95% ci from basic bootstrap
```

```
##     2.5%    97.5%
## 4.064429 5.190881
```

```r
print("95% Confidence Interval for Stomach Cancer (Basic BCa)")
```

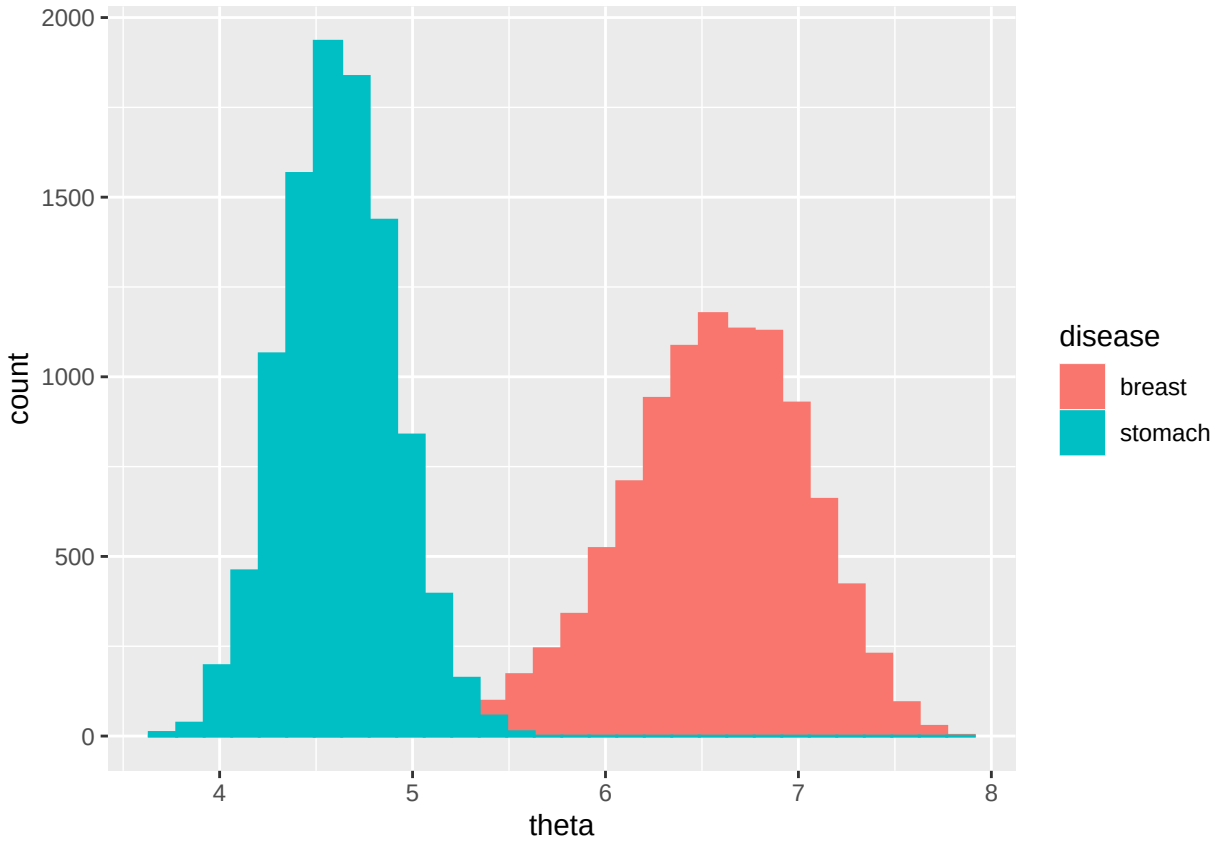
```
## [1] "95% Confidence Interval for Stomach Cancer (Basic BCa)"
```

```r
ci.95.bc.s #95% ci from BCA
```

```
## 67.88046% 99.99967%
##  4.750756  5.614601
```

```
cancer.output <- data.frame(
  theta = c(thetas.s,thetas.b),
  disease = c(rep('stomach',length(thetas.s)),rep('breast',length(thetas.b)))
)
ggplot(cancer.output,aes(x = theta, fill = disease, color = disease)) + geom_histogram(position = "iden
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
library(ggplot2)
#get data
cancer <- read.table("cancersurvival.dat", header = TRUE) #1 is stomach cancer, 2 is breast cancer
cancer[,1] <- log(cancer[,1])
breast.cancer <- cancer[cancer[,2]==2,1]
stomach.cancer <- cancer[cancer[,2]==1,1]
#initialize
n <- length(breast.cancer)
B <- 100000
#Bootstrap t
#initialize data
thetas.b <- rep(0,B)
t.b <- rep(0,B)
ci.95.t.b <- rep(0,2)
thetas.s <- rep(0,B)
t.s <- rep(0,B)
```

```r
ci.95.t.s <- rep(0,2)
#get original theta value from line 257, use to calculate covariance
theta.b <- mean(breast.cancer)
sigma.b <- sd(breast.cancer)/sqrt(length(breast.cancer))
theta.s <- mean(stomach.cancer)
sigma.s <- sd(stomach.cancer)/sqrt(length(stomach.cancer))
#Bootstrap loop
for(d in 1:B){
  rand.sample <- sample(1:n,n,replace = TRUE)
  cancer.new.s <- stomach.cancer[rand.sample]
  cancer.new.b <- breast.cancer[rand.sample]
  thetas.b[d] <- mean(cancer.new.b)
  thetas.s[d] <- mean(cancer.new.s)
  sigmas.b <- sd(cancer.new.b)/sqrt(length(breast.cancer))
  sigmas.s <- sd(cancer.new.s)/sqrt(length(stomach.cancer))
  t.b[d] = (thetas.b[d]-theta.b)/sigmas.b #reference distribution
  t.s[d] = (thetas.s[d]-theta.s)/sigmas.s #reference distribution
}
ci.95.t.b[1] = theta.b - sigma.b*quantile(t.b,.975,na.rm=T)
ci.95.t.b[2] = theta.b - sigma.b*quantile(t.b,.025,na.rm=T)
print("95% Confidence Interval for Breast Cancer")
```

```
## [1] "95% Confidence Interval for Breast Cancer"
```

```r
ci.95.t.b
```

```
## [1] 4.274203 7.399529
```

```r
ci.95.t.s[1] = theta.s - sigma.s*quantile(t.s,.975,na.rm=T)
ci.95.t.s[2] = theta.s - sigma.s*quantile(t.s,.025,na.rm=T)
print("95% Confidence Interval for Stomach Cancer")
```

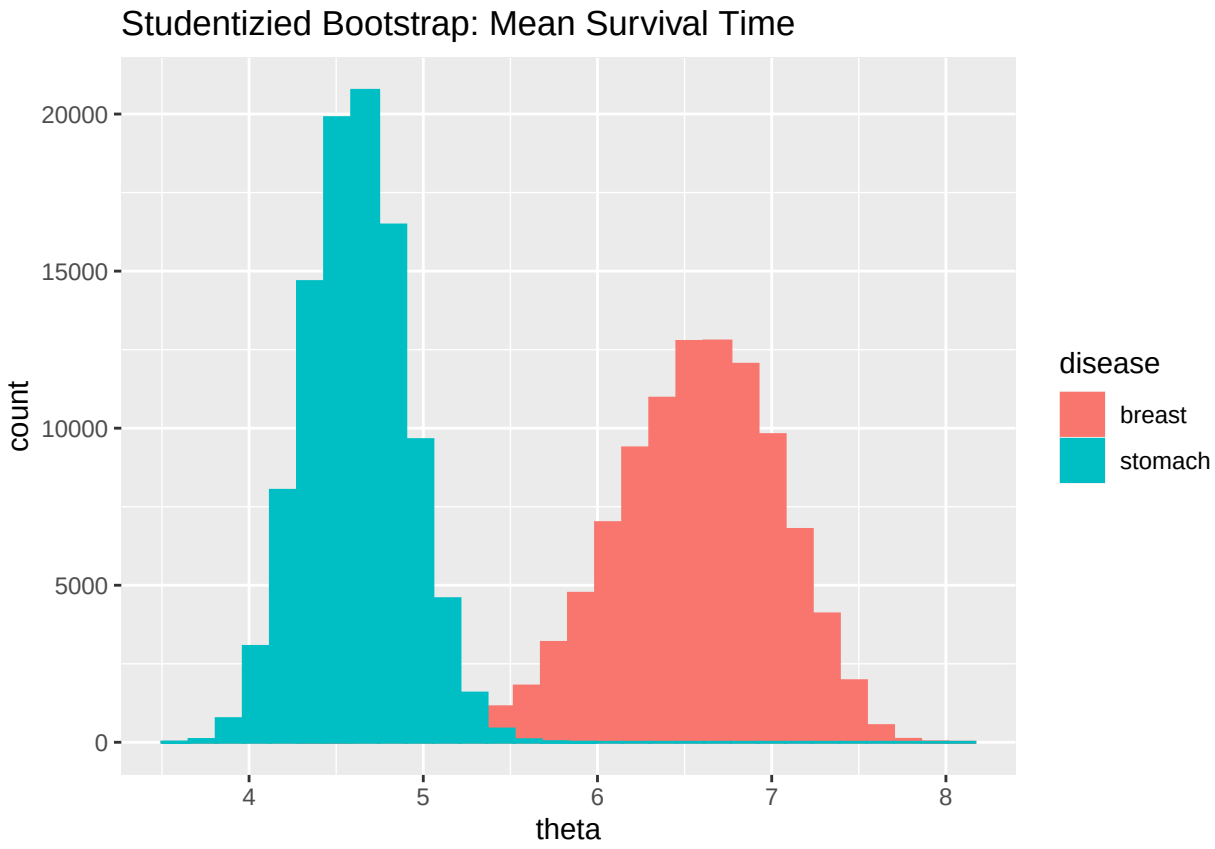
```
## [1] "95% Confidence Interval for Stomach Cancer"
```

```r
ci.95.t.s
```

```
## [1] 4.671640 6.564703
```

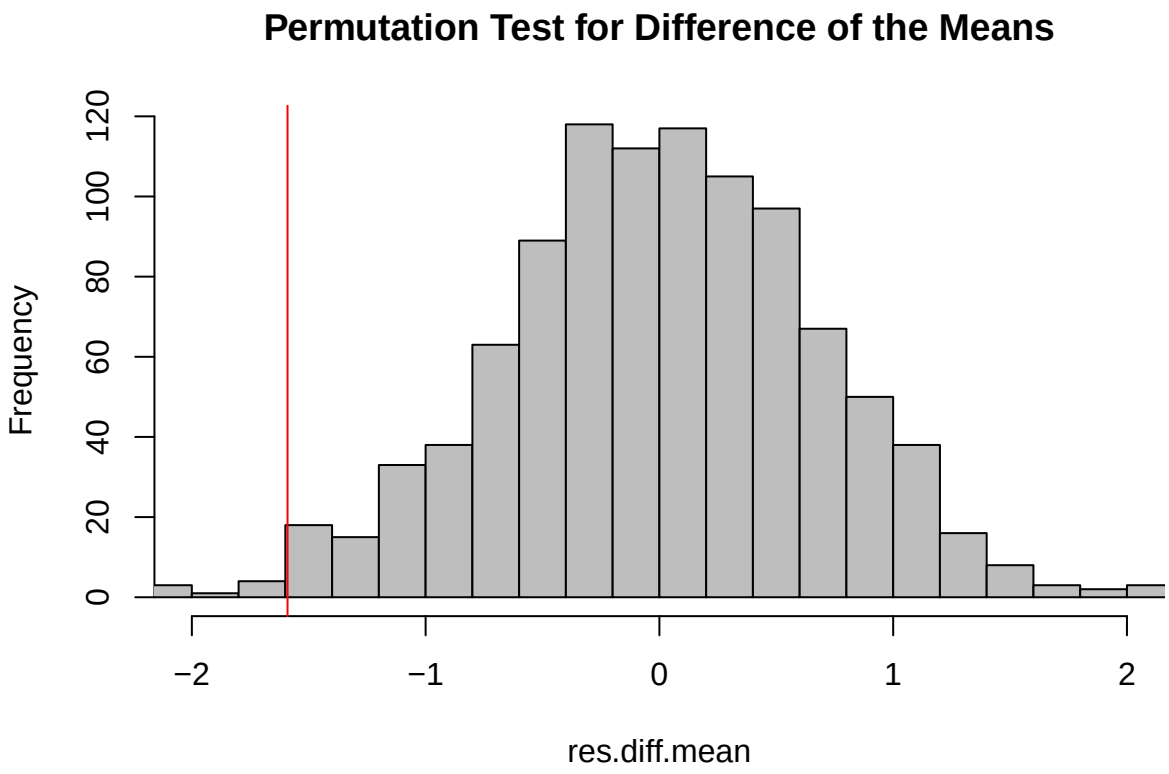
```r
cancer.output <- data.frame(
  theta = c(thetas.s,thetas.b),
  disease = c(rep('stomach',length(thetas.s)),rep('breast',length(thetas.b)))
)
ggplot(cancer.output,aes(x = theta, fill = disease, color = disease)) + geom_histogram(position = "iden
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



part b

```
set.seed(475)
#get data
cancer <- read.table("cancersurvival.dat", header = TRUE) #1 is stomach cancer, 2 is breast cancer
cancer[,1] <- log(cancer[,1])
#initialize
P <- 1000 #Number of permutation
theta.b <- mean(cancer[cancer$disease == 2,1])
theta.s <- mean(cancer[cancer$disease == 1,1])
res.diff.mean <- rep(0,P)
#Loop for permutation test
for(p in 1:P){
  perm <- sample(nrow(cancer))
  dat <- transform(cancer, disease = disease[perm])
  thetas.b <- mean(dat[dat$disease == 2,"survivaltime"])
  thetas.s <- mean(dat[dat$disease == 1,"survivaltime"])
  res.diff.mean[p] <- thetas.s - thetas.b
}
obs.diff.mean <- theta.s-theta.b
hist(res.diff.mean, breaks = 25, col = "gray", xlim = c(-2,2), main = "Permutation Test for Difference
abline(v = obs.diff.mean, col = "red")
```




```
quantile(res.diff.mean,probs = c(0.025,0.975))
```

```
##      2.5%     97.5%
## -1.406922  1.244655
```

```
obs.diff.mean
```

```
## [1] -1.590684
```

Because the observed mean is outside of the 2.5% confidence interval, we can reject the null hypothesis that there is no difference in the mean survival time for breast and stomach cancer.

part c

```
set.seed(475)
#get data
cancer <- read.table("cancersurvival.dat", header = TRUE) #1 is stomach cancer, 2 is breast cancer
cancer <- cancer[cancer$disease == 2,]
log.cancer <- log(cancer[,1])
thetas.b <- replicate(B,expr = {
  mean(log.cancer[sample(length(log.cancer),replace = TRUE)])
})
```

```
true.thetas <- replicate(B,expr = {
  mean(cancer[sample(nrow(cancer),replace = TRUE),1])
})
quantile(exp(thetas.b), probs = c(0.025,0.975))
```

```
##      2.5%     97.5%
##  260.4049 1624.4914
```

```
quantile(true.thetas, probs = c(0.025,0.975))
```

```
##      2.5%     97.5%
##  751.4545 2132.0932
```

The first confidence interval calculated by returning the log scale data to the original scale while the second confidence interval is calculated from the original dataset. The first quantile is significantly skewed away from the second quantile set. Although this is only looking at breast cancer mean survival times, exponentiation after the bootstrap and confidence interval construction skews the data. Moreover, unlike in part (a), we are not utilizing the variance in this example which is why the first quantile from this problem is overall lower in confidence interval values compared to those calculated in part (a) and in teh confidence interval calculated using the original data.

Question 5

part i

In order to compute the boot strap failure, we first need to find the Maximum Likelihood Estimator (MLE) of θ. The likelihood is $L(\theta|X_1, ..., X_n)$, which expands out to:

$$L(\theta) = (\frac{1}{\theta})^n \Pi_{i=1}^n f(X_n|\theta)$$

$$L(\theta) = (\frac{1}{\theta})^n 1(X_1 > 0) 1(X_n < \theta)$$

From here we can see that X_n is the MLE because if $\theta < X_n$ the likelihood function becomes $\frac{1}{\theta^n} X_1$ which is exponential decreasing because of the θ^n term in the denominator. If $\theta \geq X_n$ the function becomes dictated by X_n. This also supports the following:

$$lim_{n->\infty} P(-n(X_n - \theta) \leq t)$$

if we get the MLE, X_n by itself:

$$= lim_{n->\infty} P(X_n \leq \theta - \frac{t}{n})$$

and because all t values are greater than zero:

$$= 1 - (1 - \frac{t}{\theta}\frac{1}{n})^n$$

$$= 1 - e^{-t/\theta}$$

part ii Say we have $T = -n(X_n - \theta)$ and $T^* = -n(X_n^* - X_n)$, and we know all t values are greater than 0. Say we want to find $P(T^* = 0)$.

$$P(T^* = 0) => 0 = -n(X_n^* - X_n)$$

$$P(X_n^* = X_n)$$

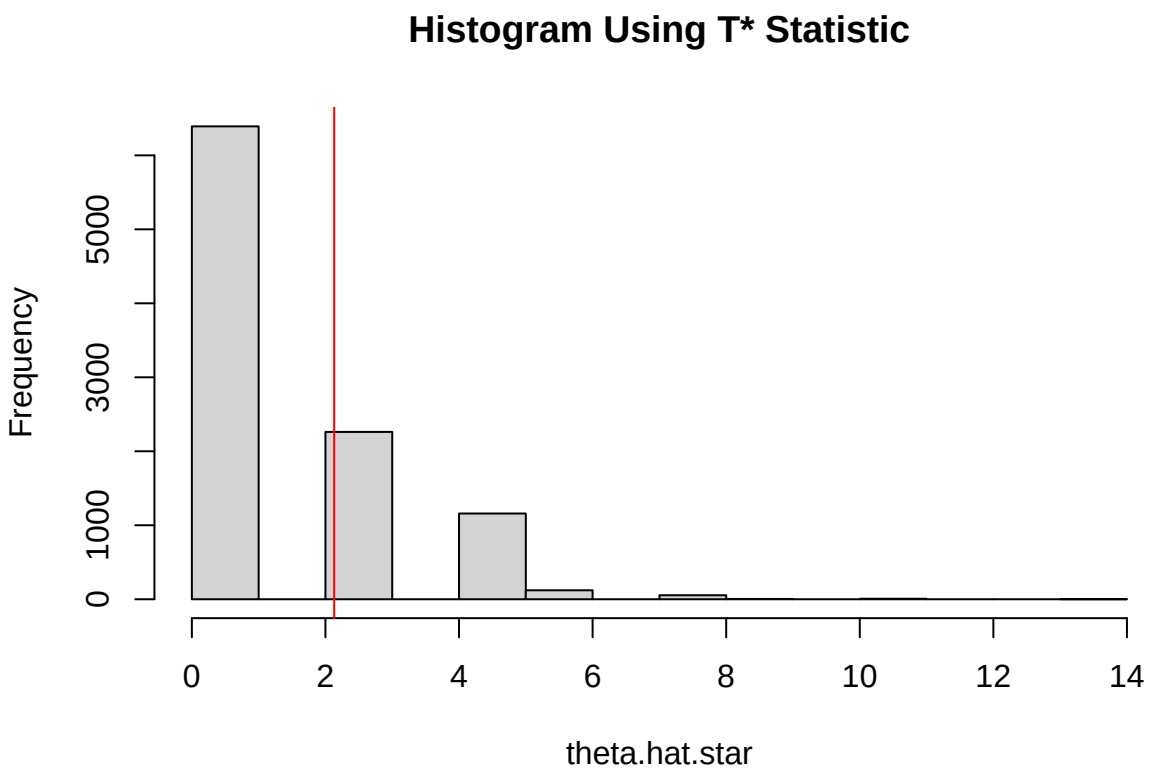
$$P(X_n^* = X_n) = 1 - (1 - \frac{1}{n})^n$$

$$= 1 - e^{-1} \approx 0.632$$

Hence the distribution of T^* is not close to the distribution of T, and behave badly for small t.

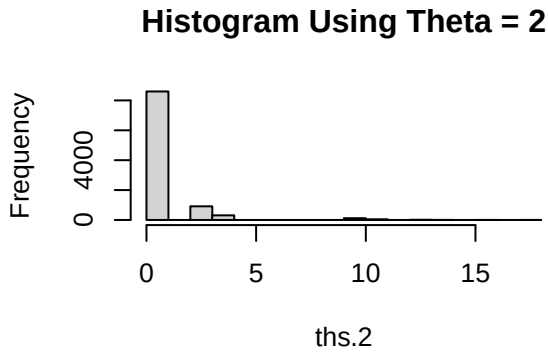
part iii

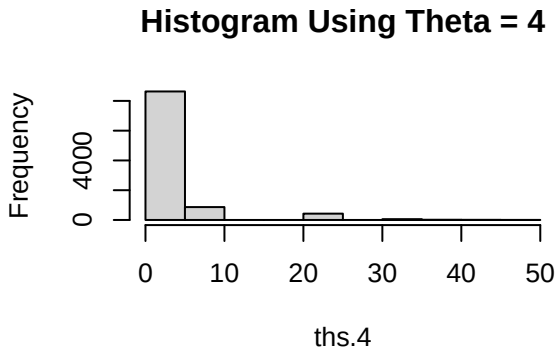
```
set.seed (475)
theta = 1
n = 500#sample length
itr = 10000 #Bootstrap iteration length
x <- runif(n, min = 0, max = theta) #uniform distribution,observed data
theta.hat <- -n*(max(x)-theta)
theta.hat.star <- numeric(itr)

for(i in 1:itr){
  samp = sample(1:n,n,replace = TRUE)
  x.star = x[samp]#x.star is pseudo bootstrap data
  theta.hat.star[i] <- -n*(max(x.star)-max(x))
}

hist(theta.hat.star, main = "Histogram Using T* Statistic")
abline(v = theta.hat, col = "red")
```

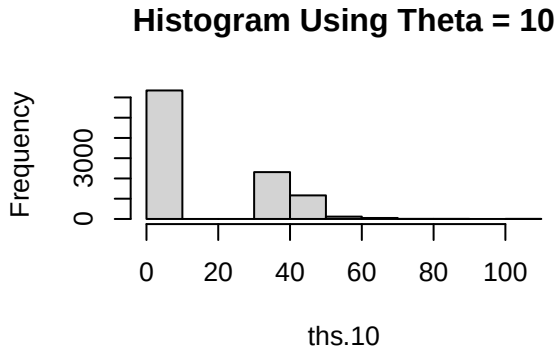


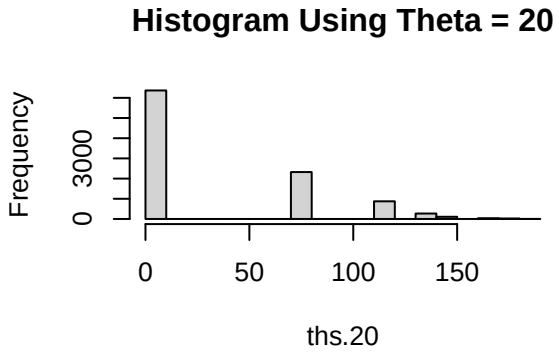

```r
#Testing failure
theta = c(2,4,10,20)
x.2 <- runif(n, min = 0, max = 2)
x.4 <- runif(n, min = 0, max = 4)
x.10 <- runif(n, min = 0, max = 10)
x.20 <- runif(n, min = 0, max = 20)
ths.2<- numeric(itr)
ths.4<- numeric(itr)
ths.10<- numeric(itr)
ths.20<- numeric(itr)
for(i in 1:itr){
  samp = sample(1:n,n,replace = TRUE)
  x.star.2 = x.2[samp]#x.star is pseudo bootstrap data
  x.star.4 = x.4[samp]
  x.star.10 = x.10[samp]
  x.star.20 = x.20[samp]
  ths.2[i] <- -n*(max(x.star.2)-max(x.2))
  ths.4[i] <- -n*(max(x.star.4)-max(x.4))
  ths.10[i] <- -n*(max(x.star.10)-max(x.10))
  ths.20[i] <- -n*(max(x.star.20)-max(x.20))

}

old.par <- par(mfrow = c(2,2))
hist(ths.2, main = "Histogram Using Theta = 2")
hist(ths.4, main = "Histogram Using Theta = 4")
```

```
hist(ths.10, main = "Histogram Using Theta = 10")
hist(ths.20, main = "Histogram Using Theta = 20")
```










```
par(old.par)
```

As we can see from the above histogram, the T^* statistic has a **point mass at zero** even when we make theta very large (shown above). Because of this, the bootstrap fails for the T^* statistic.

part iv.

Yes, because m-out-n bootstrap uses the likelihood as the bootstrap statistic which we showed in part i that it depends on θ. Moreover, m (the iteration value and indexing value in bootstrap cases) is strategically chosen so it converges to infinity, which is what we want the statistic of interest to do ($\hat{X}_n$).