HW3

Hannah Zmuda

11/18/2020

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

```
#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:  0.153 0.08 0.066
```

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

```
## The standard errors (of the p-values) are:  0.0113838 0.008579044 0.007851369
```

```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
```

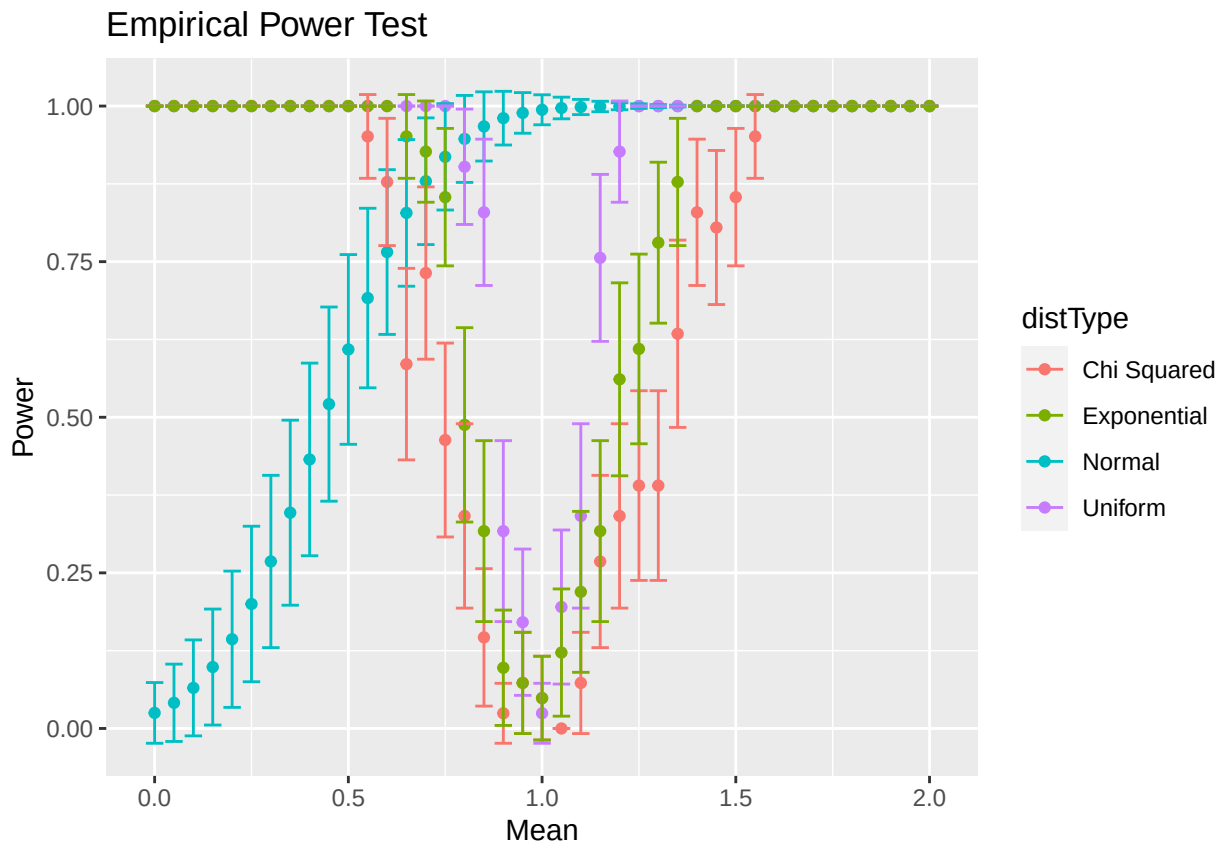
```r
  powerTest <- power.t.test(n = MC, delta = mu1[j],sig.level = a,alternative = "two.sided")
  powNorm[j] <- powerTest$power

}

#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_errorbar(data = data, mapping = aes(x = mean, ymin = powerR - 2*(sqrt(powerR * (1- powerR)/MC)),
```



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 <- 100 # 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: 0.02
```

```
#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: 0.02424404
```

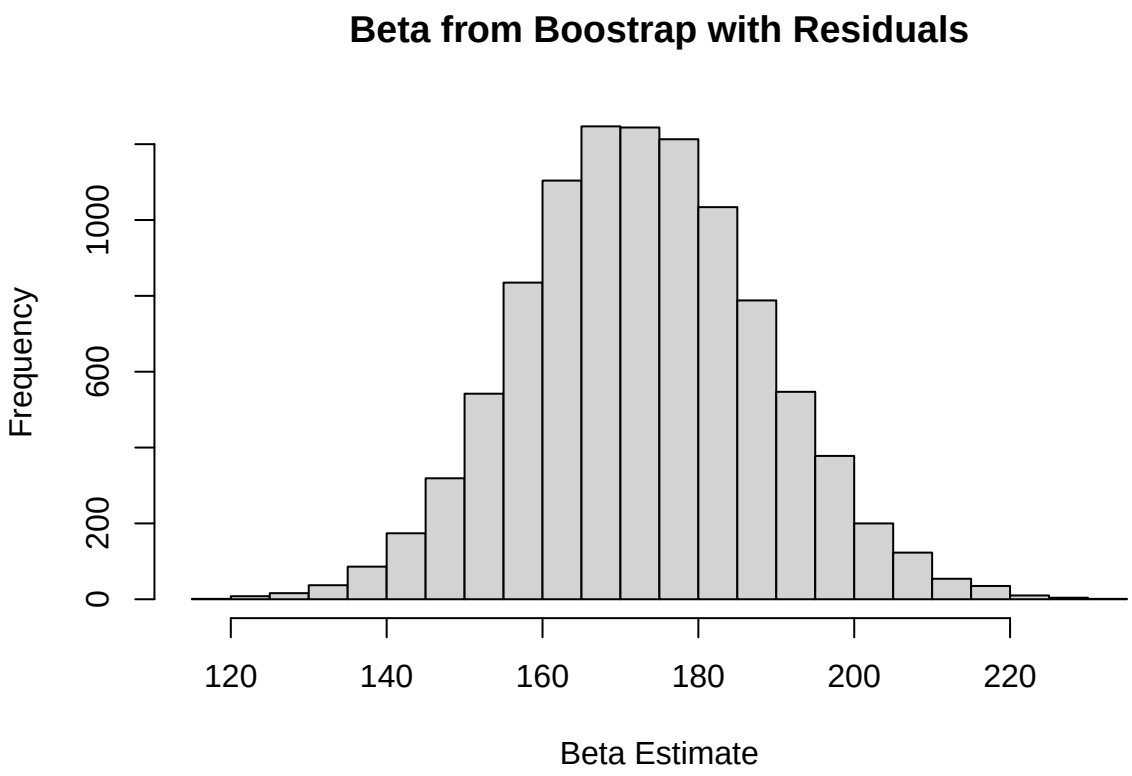
Part c: Which is better?

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, IS estimates the α value closer to the expected value, compared to using Monte Carlo by itself.

Question 3

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

#Bootstrapping the residuals
for(b in 1:B){
  eps.new <- eps[sample(1:n,n,replace = TRUE)]
  yB <- eps.new + beta
  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")
```




```
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%
## 142.9012 204.2250
```

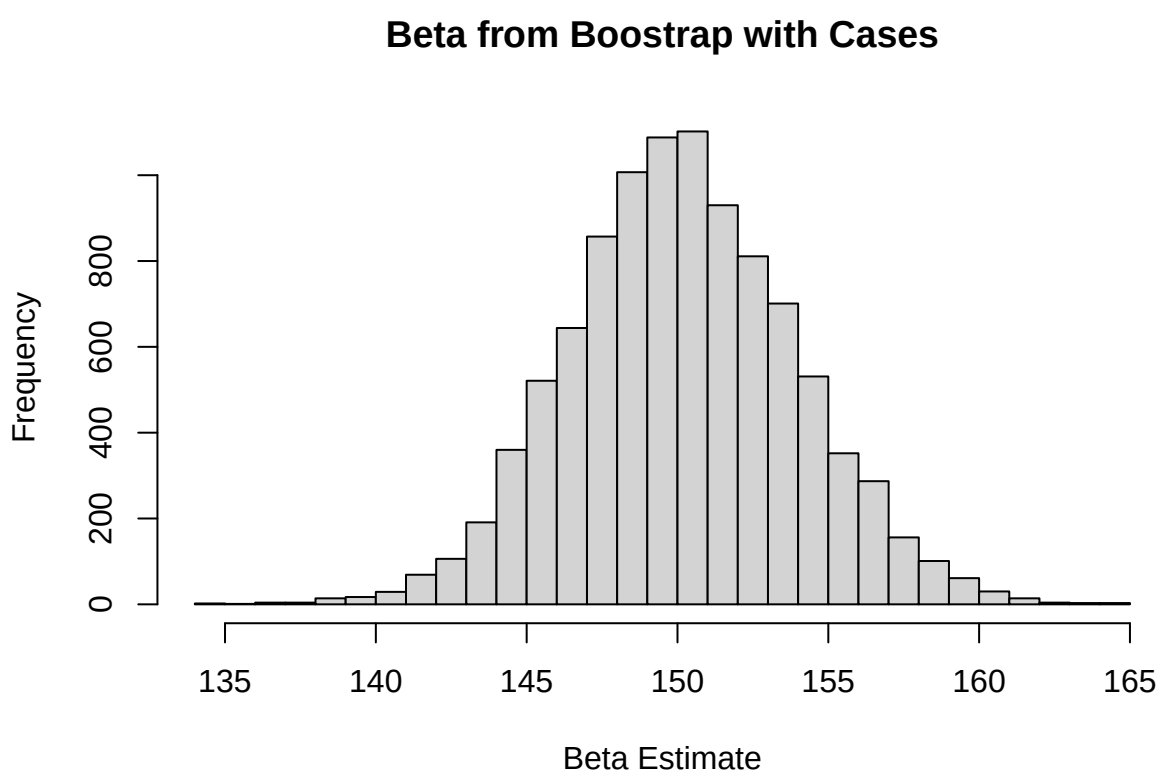
```
#part b: Jackknife after bootstrap
se.beta <- rep(0,B)
for(b in 1:B){
  se.beta[b] <- sd(beta.hat[-b])
}
se.beta.bar <- mean(se.beta)
se.beta.jack <- (B-1)/B*sum((se.beta-se.beta.bar)^2)
cat("Bias Corrected Estimate:", (B-1)*(se.beta.bar-sd(beta.hat)),"\n")
```

```
## Bias Corrected Estimate: -0.0003966144
```

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

```
## SE from Jackknife-after-bootstrap: 0.01226572
```

```r
#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)
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")
```




```r
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"
```

```r
ci.95.case
```

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

```r
#part b: Jackknife after bootstrap
se.beta <- rep(0,B)
m.beta <- rep(0,B)
for(b in 1:B){
  se.beta[b] <- sd(beta.new[-b])
}
se.beta.bar <- mean(se.beta)
se.beta.jack <- (B-1)/B*sum((se.beta-se.beta.bar)^2)
cat("Bias Corrected Estimate:", (B-1)*(se.beta.bar-sd(beta.new)),"\n")
```

```
## Bias Corrected Estimate: -0.0001014934
```

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

```
## SE from Jackknife-after-bootstrap: 0.0007641416
```

Question 4

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

#Original estimtaes
model <- lm(cancer[,2]~cancer[,1])
theta <- as.numeric(model$coefficients[2]/model$coefficients[1])

#Bootstrap!!
for(i in 1:B){
  cancer.new <- cancer[sample(1:n,n,replace = TRUE),]
  model.new <- lm(cancer.new[,2]~cancer.new[,1])
  thetas[i] <- as.numeric(model.new$coefficients[2]/model.new$coefficients[1])
}
ci.95 <- quantile(thetas,c(0.025,0.975),na.rm=T)

#BCA
for(j in 1:n){
  mod = lm(cancer[-j,2] ~ cancer[-j,1])
  theta.hat[j] <- as.numeric(mod$coefficients[2]/mod$coefficients[1])
}
for(k in 1:n){
  psi[k] <- mean(theta.hat[-k])-theta.hat[k]
}
a <- ((1/6)*sum(psi^3))/((sum(psi^2))^(3/2))
b <- qnorm(mean(thetas<theta),0,1)
beta1 <- pnorm(b + (b+qnorm(.025,0,1))/(1-a*(b+qnorm(.025,0,1))))
beta2 <- pnorm(b + (b+qnorm(.975,0,1))/(1-a*(b+qnorm(.975,0,1))))
ci.95.bc = quantile(thetas,c(beta1,beta2),na.rm=T)
#output
theta #observed estimate theta
```

```
## [1] 0.0002209304
```

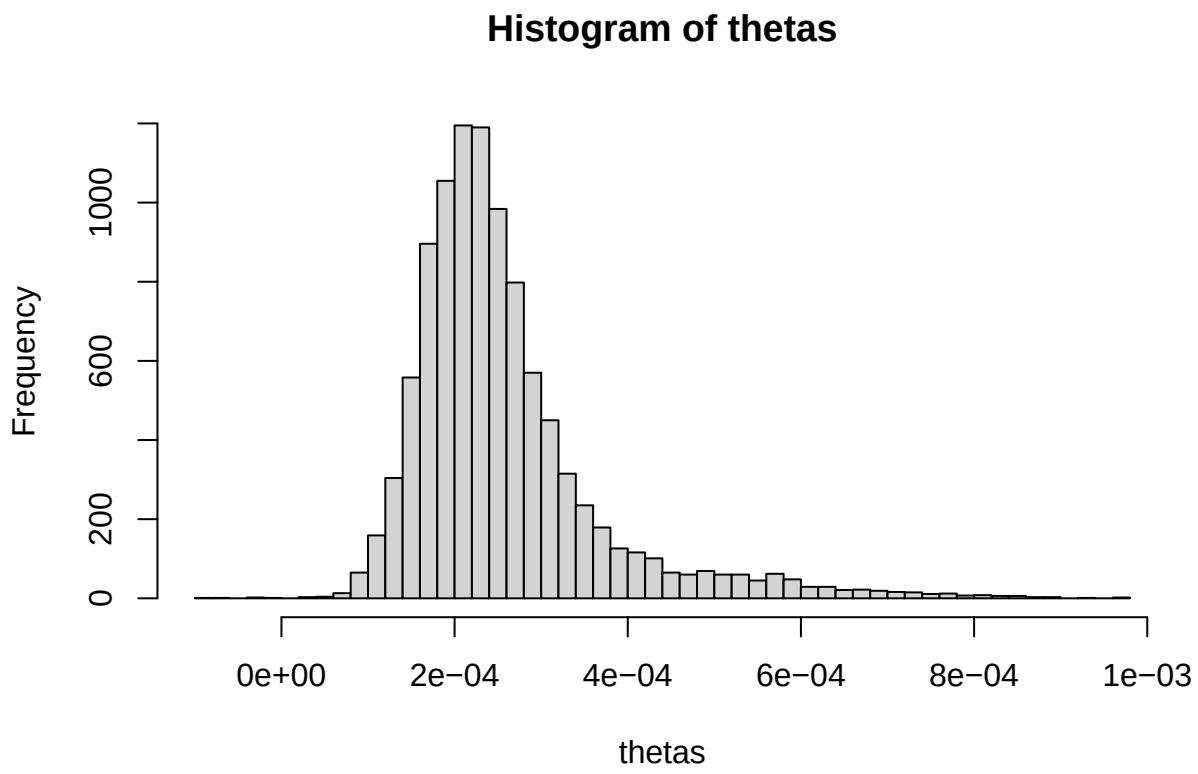
```
ci.95 #95% ci from basic bootstrap
```

```
##          2.5%         97.5%
## 0.0001202085 0.0005851816
```

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

```
##   0.4463402%    92.56757%
## 0.0000878076 0.0004276131
```

```r
hist(thetas,breaks=40)
```

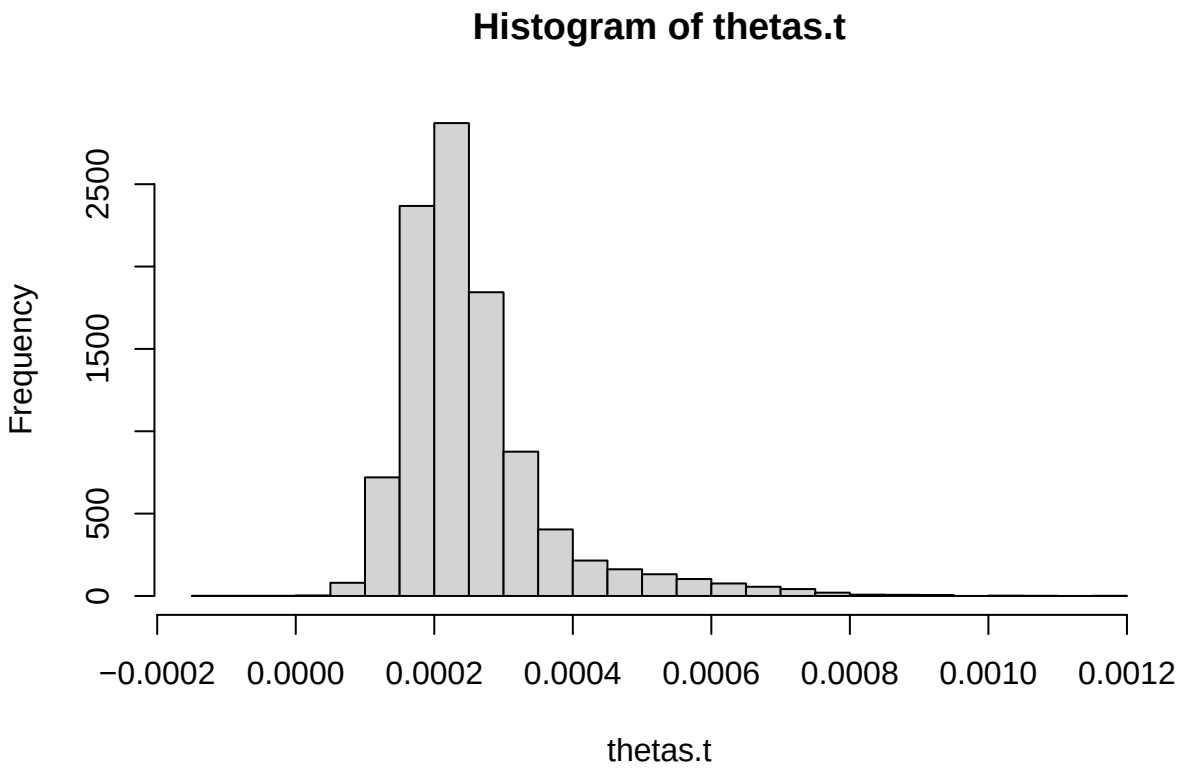



```r
#Bootstrap t
#initialize data
thetas.t <- rep(0,B)
G <- rep(0,B)
ci.95.t <- rep(0,2)
#get original theta value from line 257, use to calculate covariance
cov.t <- summary(model)$cov
b.t <- (theta^2)*(cov.t[2,2]/(model$coef[2]^2) +
       cov.t[1,1]/(model$coef[1]^2) - 2*cov.t[1,2]/(prod(model$coef)))
sigma.t <- sqrt(b.t)
#Bootstrap loop
for(d in 1:B){
  cancer.new <- cancer[sample(1:n,n,replace = TRUE),]
  model.new <- lm(cancer.new[,2]~cancer.new[,1])
  thetas.t[d] <- as.numeric(model.new$coefficients[2]/model.new$coefficients[1])
  covs.t <- summary(model)$cov
  #equation based on delta method
  bs.t <- (thetas.t[d]^2)*(covs.t[2,2]/(model.new$coef[2]^2) +
       covs.t[1,1]/(model.new$coef[1]^2) - 2*covs.t[1,2]/(prod(model.new$coef)))
  sigmas.t <- sqrt(bs.t)
  G[d] = (thetas[d]-theta)/sigmas.t
}
ci.95.t[1] = theta - sigma.t*quantile(G,.975,na.rm=T)
```

```r
ci.95.t[2] = theta - sigma.t*quantile(G,.025,na.rm=T)
ci.95.t
```

```
## [1] -0.0001425911  0.0003197586
```

```r
hist(thetas.t,breaks = 40)
```



Question 5