Stat428 HW3

Question1

```r
sample_mix = function(n){
  X = rcauchy(n)
  return(X)
}
tmean = function (X,k){
  X_s = sort(X)
  tm = mean(X_s[(k+1):(length(X)-k)])
  return (tm)
}
trimmed_mse <- function(m, n, k){
  thetahats <- numeric(m)
  for(j in 1:m){
    X <- sample_mix(n)
    thetahats[j] <- tmean(X, k)
  }
  mse.est <- mean((thetahats - 0)^2)
  se.mse <- sd((thetahats-0)^2)/sqrt(m)
  return(c(mse.est, se.mse))
}
```

```r
ks = 1:9
result_1 = sapply (ks, FUN = trimmed_mse, m=10000, n=20)
rownames(result_1)=c('MSE', 'SE')
colnames(result_1)=paste0('k=',1:9)
```

```r
result_1
```

```
##           k=1        k=2         k=3         k=4         k=5         k=6
## MSE 1.5930993 0.44161236 0.251746352 0.183829096 0.154295422 0.143106144
## SE  0.1697393 0.02128215 0.005619902 0.003230343 0.002711382 0.002382265
##             k=7         k=8         k=9
## MSE 0.133501677 0.137562780 0.138392126
## SE  0.002325111 0.002322455 0.002324297
```

Question 2

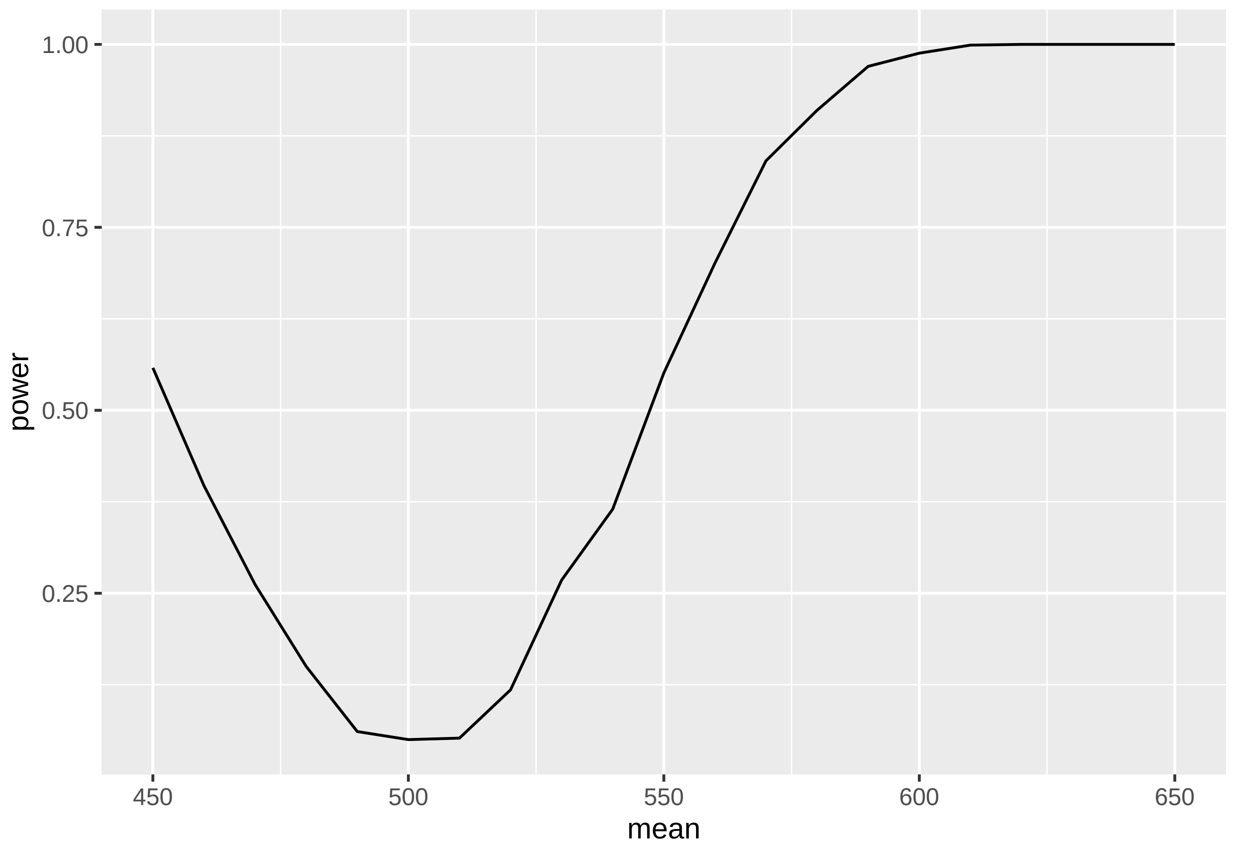
```r
n <- 20
m <- 1000
mu0 <- 500
sigma <- 100
mu <- c(seq(450, 650, 10))
M <- length(mu)
```

```r
power <- numeric(M)
for (i in 1:M) {
  mu1 <- mu[i]
  pvalues <- replicate(m, expr = {
    x <- rnorm(n, mean = mu1, sd = sigma)
    ttest <- t.test(x,alternative = "two.sided", mu = mu0)
    ttest$p.value})
  power[i] <- mean(pvalues <= .05)
}
power
```

```
##  [1] 0.558 0.397 0.262 0.150 0.061 0.050 0.052 0.118 0.268 0.365 0.551 0.701
## [13] 0.841 0.910 0.970 0.988 0.999 1.000 1.000 1.000 1.000
```

```r
se <- sqrt(power * (1-power) / m)
df <- data.frame(mean=mu, power=power, upper=power+2*se, lower=power-2*se)
```

```r
library(ggplot2)
ggplot(df,aes(x=mean, y=power)) + geom_line()
```



Question 3

Since X1, . . . , Xn to have a log-normal distribution Therefore we can take xbar to be the mean of Y where Y is equal to log(x) with x having a normal distribution.

```r
n = 300
m = 10000
alpha = 0.05
conint_0 = numeric(m)
for (i in 1:m){
  x = log(rlnorm(n))
  xbar = mean(x)
  U= xbar - (qnorm(0.975)*sd(x)/sqrt(n))
  V= xbar + (qnorm(0.975)*sd(x)/sqrt(n))
  conint_0[i] = 1*(0>U & 0<V)
}
mean(conint_0)
```

```
## [1] 0.9481
```

```r
n = 500
m = 10000
alpha = 0.05
conint_1 = numeric(m)
for (i in 1:m){
  x = log(rlnorm(n))
  xbar = mean(x)
  U= xbar - (qnorm(0.975)*sd(x)/sqrt(n))
  V= xbar + (qnorm(0.975)*sd(x)/sqrt(n))
  conint_1[i] = 1*(0>U & 0<V)
}
mean(conint_1)
```

```
## [1] 0.9428
```

```r
n = 1000
m = 10000
alpha = 0.05
conint_2 = numeric(m)
for (i in 1:m){
  x = log(rlnorm(n))
  xbar = mean(x)
  U= xbar - (qnorm(0.975)*sd(x)/sqrt(n))
  V= xbar + (qnorm(0.975)*sd(x)/sqrt(n))
  conint_2[i] = 1*(0>U & 0<V)
}
mean(conint_2)
```

```
## [1] 0.947
```

Question 4

```r
alpha = 0.05;
n = 20;
m = 1000;
```

```
UCL = numeric(m);
LCL = numeric(m);

for(i in 1:m)
{
    x = rchisq(n, 2);
    LCL[i] = mean(x) - qt(alpha / 2, df = n-1, lower.tail = FALSE) * sd(x)/sqrt(n)
    UCL[i] = mean(x) + qt(alpha / 2, df = n-1, lower.tail = FALSE) * sd(x)/sqrt(n)
    x1=c(LCL[i],UCL[i])
}
x1
```

```
## [1] 0.6901277 1.8287244
```

```
mean(LCL < 2 & UCL > 2)
```

```
## [1] 0.909
```

We notice that the UCL obtained by the example provided a much larger value than the approach taken by us in this question. On several runs of the example we produced upper confidence limits of UCL = 6.628, UCL = 7.348, UCL = 9.621, etc. whereas by running the above code we are getting 3.097236, 3.476735, 2.502649

Question 5

```
type1 <- matrix(NA, nrow = 6, ncol = 5)
ns <- c(5, 10, 20, 30, 50, 100)
ps <- c(.1, .3, .5, .7, .9)
rownames(type1) <- paste('n=', ns)
colnames(type1) = c ('.1', '.3', '.5', '.7', '.9')

 t_decision <- function(X, p, alpha){
   n <- length(X)
  xbar <- mean(X)
  T_stat <- (xbar - p)/(sqrt(p*(1-p)/n))
  reject <- 1*(abs(T_stat) > qt((1-alpha/2), df = (n-1)))
  return (reject)
 }

m <- 10000
for(i in 1:length(ns)){
  n <- ns[i]
  decision <- numeric(m)
  for(j in 1:m){
    X <- rbinom(n,size=1, prob = 0.1)
    decision[j] <- t_decision(X, p= 0.1, alpha = .05) }
type1[i,1] <- mean(decision) }
for(i in 1:length(ns)){
  n <- ns[i]
  decision <- numeric(m)
  for(j in 1:m){
    X <- rbinom(n,size=1, prob = 0.3)
    decision[j] <- t_decision(X, p= 0.3, alpha = .05) }
```

```r
type1[i,2] <- mean(decision) }
for(i in 1:length(ns)){
  n <- ns[i]
  decision <- numeric(m)
  for(j in 1:m){
    X <- rbinom(n,size=1, prob = 0.5)
    decision[j] <- t_decision(X, p= 0.5, alpha = .05) }
type1[i,3] <- mean(decision) }
for(i in 1:length(ns)){
  n <- ns[i]
  decision <- numeric(m)
  for(j in 1:m){
    X <- rbinom(n,size=1, prob = 0.7)
    decision[j] <- t_decision(X, p= 0.7, alpha = .05) }
type1[i,4] <- mean(decision) }
for(i in 1:length(ns)){
  n <- ns[i]
  decision <- numeric(m)
  for(j in 1:m){
    X <- rbinom(n,size=1, prob = 0.9)
    decision[j] <- t_decision(X, p= 0.9, alpha = .05) }
type1[i,5] <- mean(decision) }
round(type1,3)
```

```
##            .1    .3    .5    .7    .9
## n= 5   0.007 0.002 0.000 0.002 0.011
## n= 10  0.015 0.011 0.021 0.010 0.014
## n= 20  0.046 0.026 0.042 0.027 0.041
## n= 30  0.027 0.029 0.040 0.025 0.027
## n= 50  0.030 0.043 0.034 0.045 0.031
## n= 100 0.062 0.039 0.057 0.039 0.065
```

We notice that in majority of the cases as the sample size is increasing so is the empirical type 1 error rate. Additionally, the empirical type 1 error rate is approaching the significance level of 0.05 as n increases.