Midterm

1. Let T3 be a t-random variable with 3 degrees of freedom. We want to estimate the threshold value c such that P (T4 < c) = 0.025 by simulation.

(a) Simulate ten thousand T4 random variables, by only using the normal random number function 'rnorm' (without using functions such as pt, qt, dt, rt, rf, pf, rchisq and rgamma etc. and without using any packages. You can use any functions not related to random variables).

```
n <- 4
Z <- rnorm(10000)
X <- numeric(10000)
for(i in 1:n)
{
  X <- X + ((rnorm(10000))^2)
}

Tn <- Z/sqrt(X/n)
```

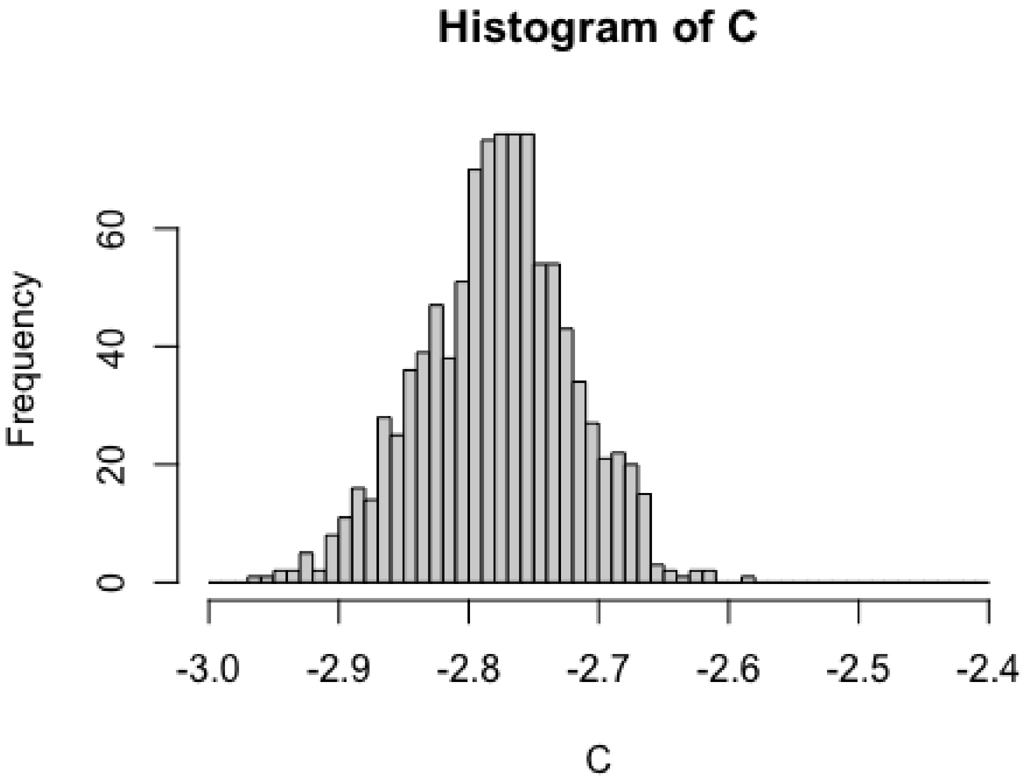
(b) Calculate the 2.5 percentile of the data in (a). (You can use any popular definition of the percentile.) It is an estimate for c.

```
percentile <- function(Tn, x)
{
  return(sort(Tn)[10000*x])
}

percentile(Tn,.025)

## [1] -2.802027
```

(c) Repeat (a) and (b) for 1000 times, and make a histogram of the estimate for c. Make the bin widths equal to 0.01.

```
cEstimate <- function(n, p)
{
  Z <- rnorm(10000)
  X <- numeric(10000)
  for(i in 1:n)
  {
    X <- X + ((rnorm(10000))^2)
  }
  Tn <- Z/sqrt(X/n)
  C <- percentile(Tn,p)
  return(C)
}

C <- replicate(1000,cEstimate(4,.025))
```

```
hist(C, seq(from=-3, to=-2.4, by=.01))
```



(d) Calculate the theoretical value of c. You can use any function.

```
theoreticalC <- qt(0.025,4)
```

2. The MSFT.csv includes the daily price of Microsoft stock from October 1, 2011 to September 30, 2021 (only for business days). There are 2516 observations (n = 2516). We want to know the distribution of daily asset return.

(a) Make the daily asset return series DX. DX is defined by “[the (natural) logarithm of Adjusted Close Price on Day t)] minus [the logarithm of Adjusted Close Price on Day (t − 1)]” for all possible t’s. Note that the sample size for DX is (n − 1). Report the sample mean and the sample standard deviation of DX.

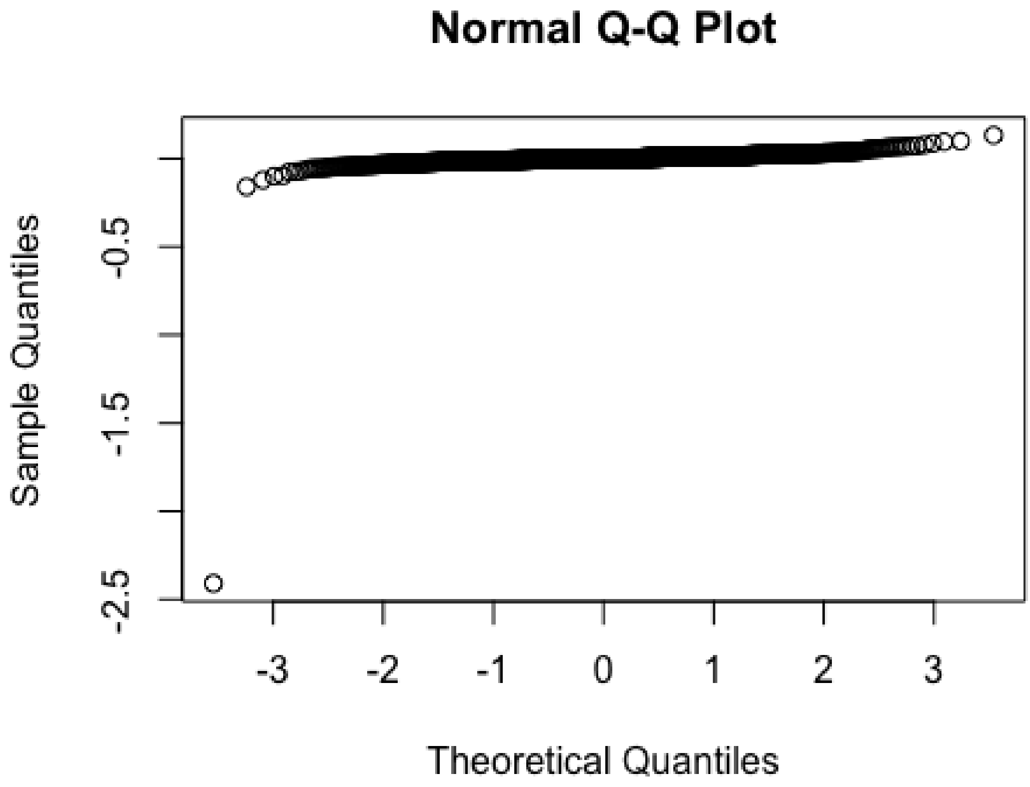
```
MSFT <- read.csv("MSFT.csv")

DX <- log(MSFT$Close[2:length(MSFT$Date)])-log(MSFT$Close[1:length(MSFT$Date)
])

print(paste("Mean: ",mean(DX)))

## [1] "Mean:  1.29122718029685e-05"

print(paste("Standard Deviation: ",sd(DX)))
```

```
## [1] "Standard Deviation:  0.0506697699955427"
```

(b) Make a normal Q-Q plot for DX. Can we conclude DX approximately follows a normal distribution? Justify your answer briefly.

We can conclude that this is not a normal distrubution due to the smallest and highest values deviating from the line and the middle being too thick.

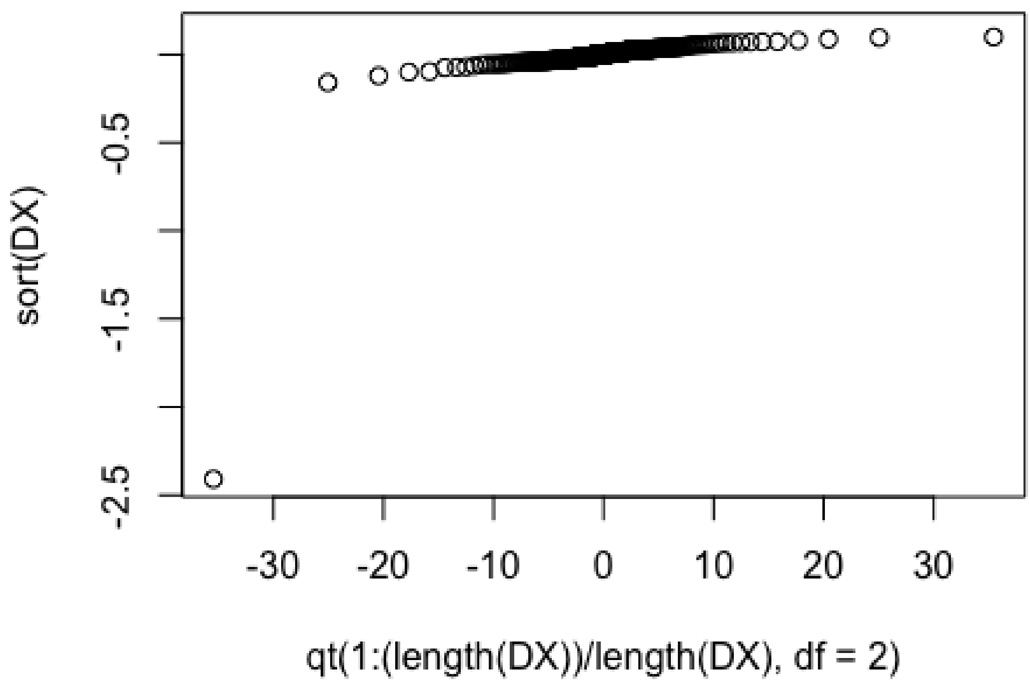
```
qqnorm(DX)
```



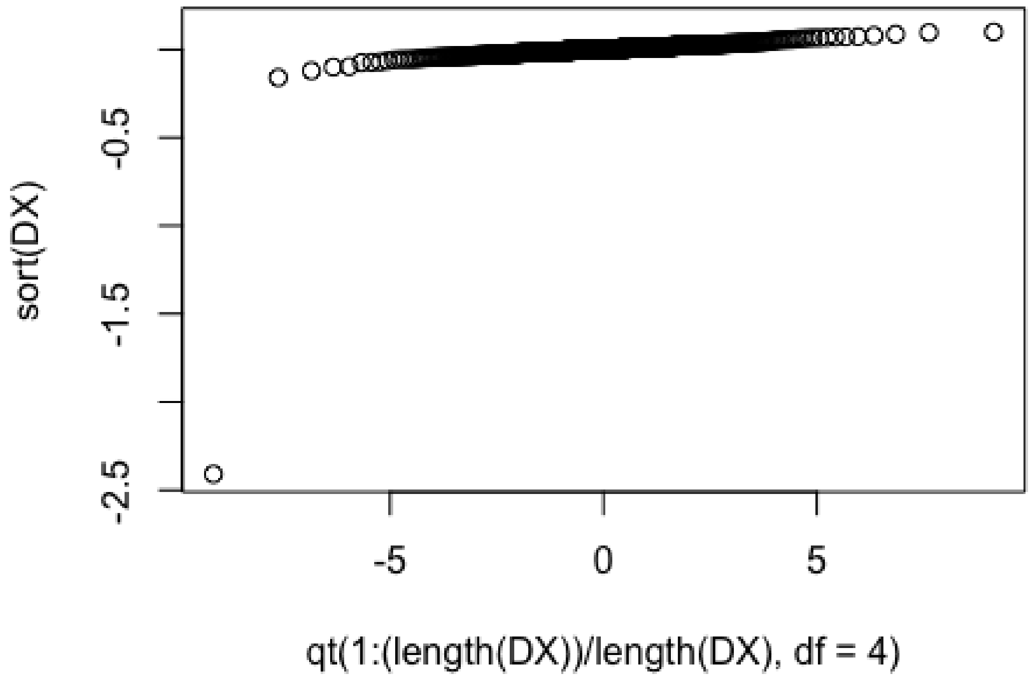
(c) We want to see if DX follows a t2, t4 or t6 distribution. Let y(1), y(2), · · · , y(n−1) be DX sorted by increasing order. Make a t2 Q-Q Plot, that is, a scatter plot of () where F−1 is the inverse cumulative distribution function (CDF) of t2-distribution. Make a similar Q-Q plot for t4- and t6-distributions also. Which distribution looks closest to the distribution of DX (when we ignore the mean and standard deviation)? Justify your answer briefly.

t6 distribution looks closest due to middle thickness and the beginning and ending deviations.

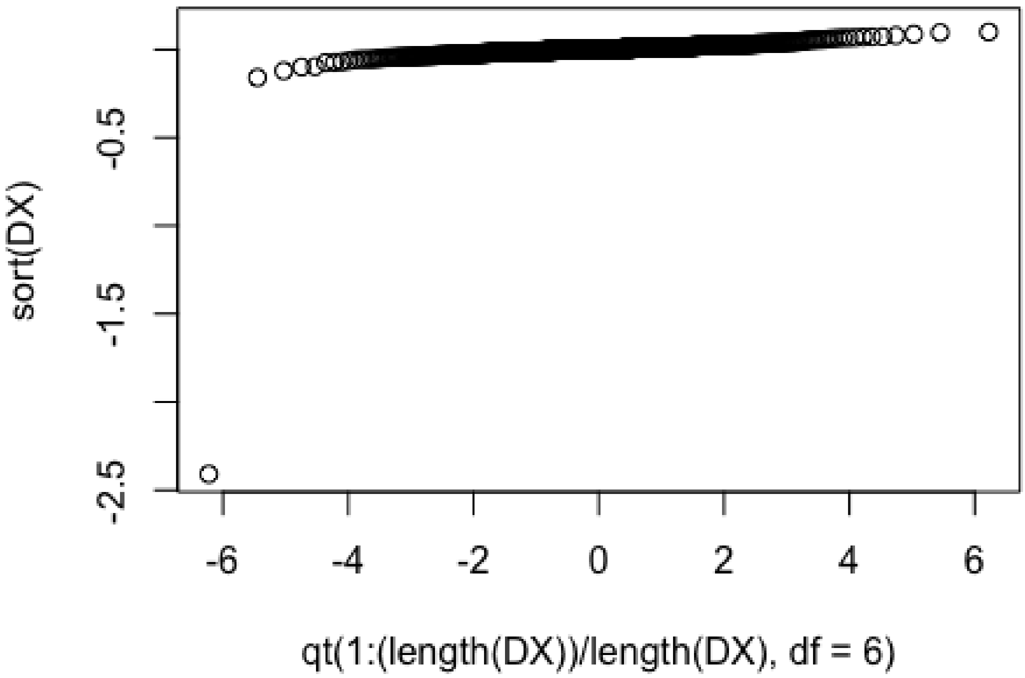
```
qqplot(qt(1:(length(DX))/length(DX),df=2),sort(DX)) #t2
```



```
qqplot(qt(1:(length(DX))/length(DX),df=4),sort(DX)) #t4
```



```
qqplot(qt(1:(length(DX))/length(DX),df=6),sort(DX)) #t6
```



3. (25 points) We want to approximate the area of x2 +y2 ≤ 1 (x ≥ 0, y ≥ 0) by simulation (note that the exact area is π/4). The simulation procedure is as follows: Step 1. Generate a pair of uniform random numbers (xi, yi) on a square [0, 1] × [0, 1] for i = 1,⋯ ,n. Step 2. Count the number of observations which satisfy x2i + yi2 ≤ 1, then divide the number by n. The obtained value is an approximated area. Run the procedure with n = 1,2,4,8,⋯ ,2k,.⋯, until the absolute error size became less than 0.001. Report the final sample size n and the absolute value of the final error size. For reproducibility, set a random number seed at the beginning of your entire code (for example, type set.seed(1014)).

```
set.seed(1014)

n <- 0
error <- .1
i <- 0

while(abs(error) >= .001)
{
  i <- i + 1
  n <- 2^i

  x <- runif(n)
  y <- runif(n)
  a <- (sum((((x^2)+(y^2))<=1))/n)

  error <- a-(pi/4)
}

print(paste("Final Sample Size: ",n))

## [1] "Final Sample Size:  65536"
```

4. (25 points)

(a) Create a function named pprod which calculates the product of two polynomials (a0 +a1X +⋯+anXn) and (b0 +b1X +⋯+bmXm). The inputs are two vec- tors (a0,a1,⋯ ,an) and (b1,b2,⋯ ,bm), and the output is the vector (c0,⋯ ,cn+m) such that c0 +c1x+⋯cn+mxn+m =(a0 +a1X+⋯+anXn)(b0 +b1X+⋯+bmXm). For example, since (1 + x)(3 + 2x2) = 3 + 3x + 2x2 + 2x3, we expect the following result:

pprod(c(1,1),c(3,0,2)) [1] 3 3 2 2

```
pprod <- function(P1,P2)
{
  if(length(P1) > length(P2))
  {
    g <- P1
    l <- P2
  }
  else
  {
    g <- P2
    l <- P1
  }

  M <- matrix(0,length(l),length(g)+(length(l)-1))

  for(i in 1:length(l))
  {
    P <- l[i]*g
    for(j in 1:(length(P)))
    {
      M[i,(i-1)+j] <- P[j]
    }
  }
  return(colSums(M))
}
```

(b) Expand the following polynomial by using the function ‘pprod’ in (a).

```
pprod(pprod(c(1,1/sqrt(2),1/2),c(1,1/sqrt(3),1/3)),pprod(c(1,1/2,1/4),c(1,1/s
qrt(5),1/5)))
```

```
## [1] 1.000000000 2.231670646 3.131843602 2.777442788 1.821754255 0.83828717
1
## [7] 0.285411795 0.061519436 0.008333333
```