Tutorial 4

Daning Bi

March 13rd, 2016

Question 1 and 2 (CB5.52, 5.53)

To generate a discrete random variable such as binomial, hypergeometric, and negative binomial distributed variable, we can start with a uniform distributed random variable and convert it to be a target random variable by calculating the cumulative distribution function for different values and insert it.

Part a

To generate a binomial random variable, first of all I generate a uniform random variable. Secondly, I calculated the CDF for all possible values of the random variable by cumulating the PMF of binomial distribution. Lastly, I look up the uniform random variable in the CDF table to find the smallest value which correspond to a CDF value larger than the uniform random variable. The smallest value I found is a random variable of the target discrete distribution.

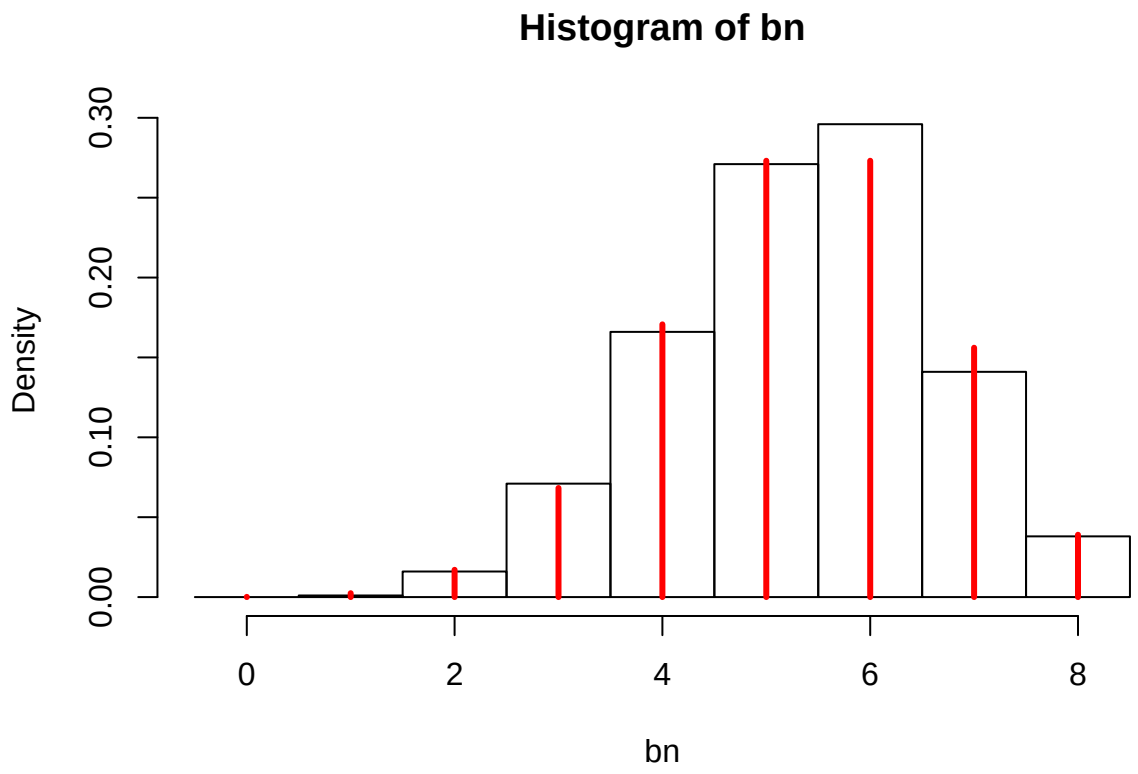
Following is the function I wrote for generating binomial random variable:

```
set.seed(1000)
binr <- function(n=1, p=.5, N=1000){
    ur <- runif(N)
    bn <- rep(n+1,N)
    x <- rep(0,n+1)
    for (i in 0:n){
        #x[i+1]<-pbinom(i,n,p,lower.tail = T)
        if (i == 0){
            x[i+1] <- choose(n,i)*p^i*(1-p)^(n-i)
        } else {
            x[i+1] <- x[i] + choose(n,i)*p^i*(1-p)^(n-i)
        }
    }
    for (i in 1:N){
        if (ur[i]<=x[1]){
            bn[i]=0
            next
        }
        for (j in 1:n){
            if (ur[i] > x[j] & ur[i]<=x[j+1]){
                bn[i]=j
                break
            }
        }
    }
    max<-max(bn)
    min<-min(bn)
    mean <- mean(bn)
    var <- var(bn)
    bins = c(-0.5:(n+0.5))
```

```
    plot = hist(bn, breaks = bins, freq = F)
    y <- 0:n
    lines(x = y, y = dbinom(y,n,p), type = "h", col = "red", lwd = 3)
    return(list(max = max, min = min, bn = bn, hist = plot, mean = mean, var = var))
}
```

We can use it to generate 1000 binomial random variables and compare the mean, variance, histogram with the theoretical values:

```
br <- binr(8,2/3,N = 1000)
```



The plot of histogram as well as the theoretical PMF has been incorporated in the function with the red lines denoting the theoretical PMFs. Next, we can compare the mean, variacne with the theoretical values.

```
c(br$mean, 8*2/3)
```

```
## [1] 5.332000 5.333333
```

```
c(br$var, 8*2/3*(1-2/3))
```

```
## [1] 1.703479 1.777778
```

As we can see, the mean are quite closed, though there is a difference in variance and histogram.

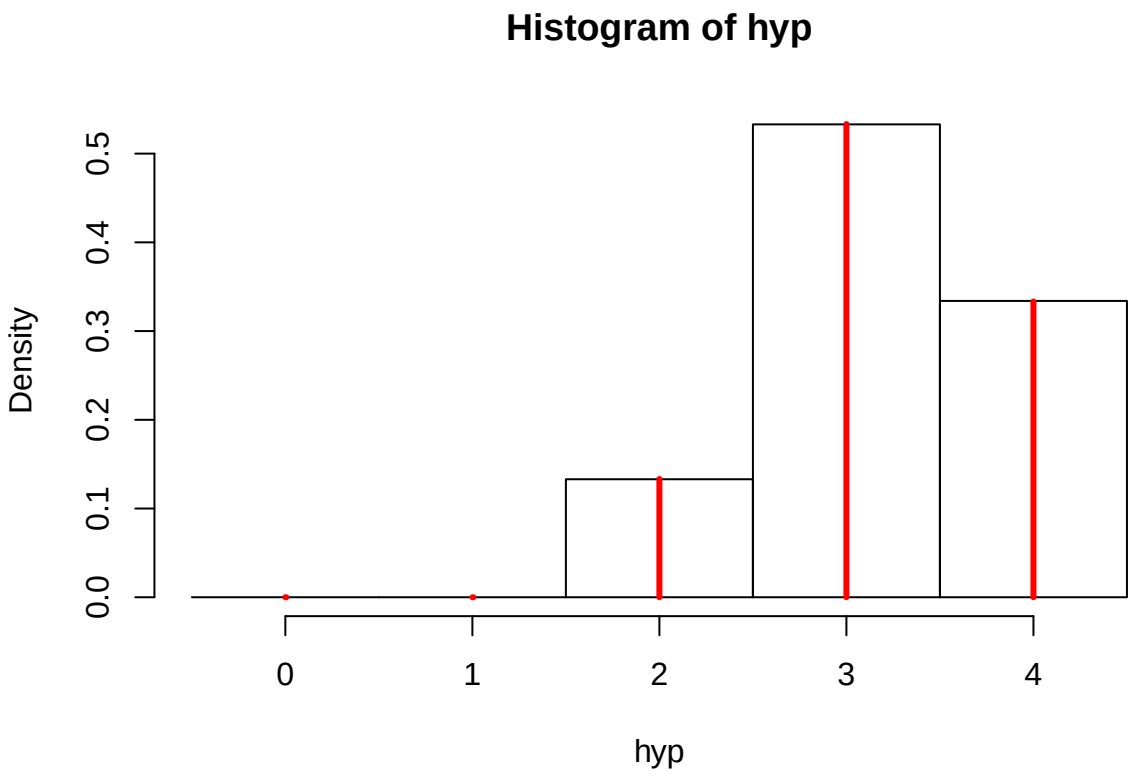
Part b

here we can apply the same logic and method to generate hypergeometric random variables. Following is the function I defined to generate the random variable, calculate mean, variance and plot histogram:

```
set.seed(1000)
hyperr <- function(n=1, m=1, k=1, N=1000){
    ur <- runif(N)
    hyp <- rep(k+1,N)
    x <- rep(0,k+1)
    for (i in 0:k){
        #x[i+1]<-phyper(i,m,n-m,k,lower.tail = T)
        if (i == 0) {
            x[i+1] <- choose(m,i)*choose(n-m,k-i)/choose(n,k)
        } else {
            x[i+1] <- x[i] + choose(m,i)*choose(n-m,k-i)/choose(n,k)
        }
    }

    for (i in 1:N){
        if (ur[i]<=x[1]){
            hyp[i]=0
            next
        }
        for (j in 1:k){
            if (ur[i] > x[j] & ur[i]<=x[j+1]){
                hyp[i]=j
                break
            }
        }
    }
    max<-max(hyp)
    min<-min(hyp)
    mean <- mean(hyp)
    var <- var(hyp)
    bins = c(-0.5:(k+0.5))
    plot = hist(hyp, breaks = bins, freq = F)
    y <- 0:k
    lines(x = y, y = dhyper(y,m,n-m,k), type = "h", col = "red", lwd = 3)
    return(list(max = max, min = min, hyp = hyp, hist = plot, mean = mean, var = var))
}
```

Similarly, we can use it to generate the target random variable:

```
hyr <- hyperr(n = 10,m =  8,k =  4,N =  1000)
```




```
c(hyr$mean, 8*4/10)
```

```
## [1] 3.201 3.200
```

```
c(hyr$var, 8*4/10*(10-4)*(10-8)/10/(10-1))
```

```
## [1] 0.4270260 0.4266667
```

Here we found that all the mean, variance and histogram are quite closed to the theoretical ones.

Part c

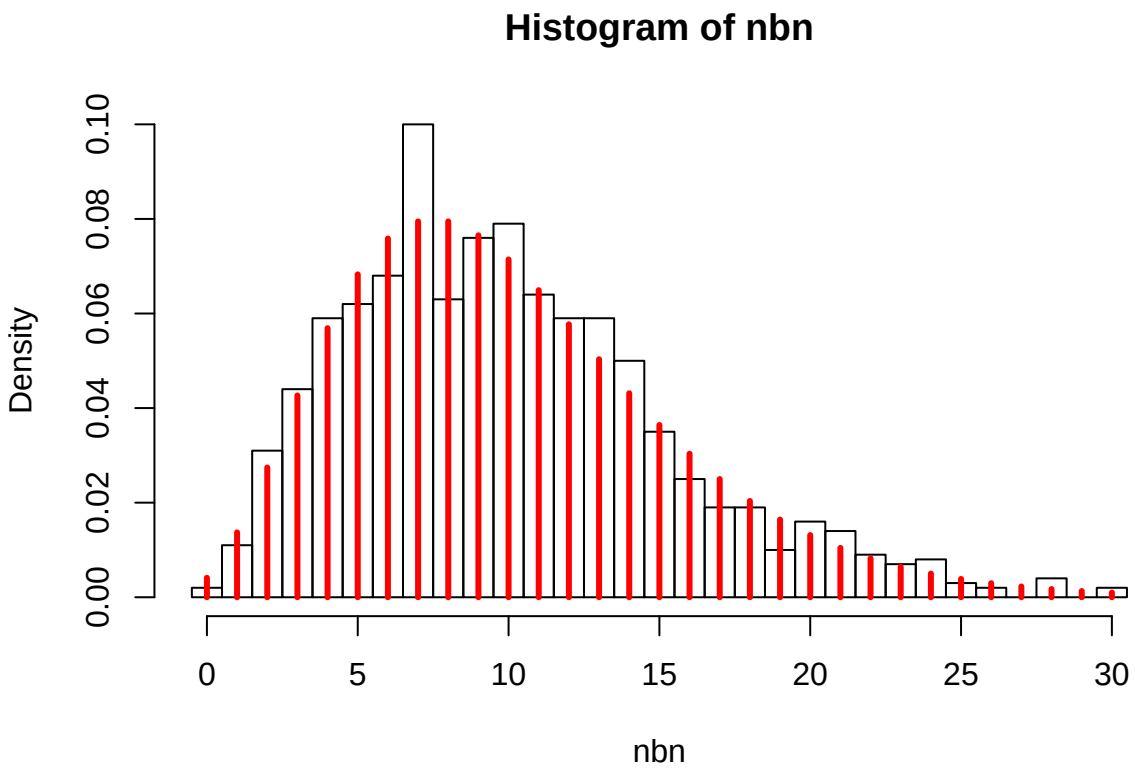
Following exactly the same idea we can define another function to generate negative binomial random variables:

```
set.seed(1000)
nbinr <- function(n=1, p=.5, N=1000){
    ur <- runif(N)
    nbn <- rep(n+100,N)
    x <- rep(0,n+100)
    for (i in 0:(n+100)){
        x[i+1]<-pnbinom(q=i,size = n, prob = p,lower.tail = T)
    }
```

```
    for (i in 1:N){
        if (ur[i]<=x[1]){
            nbn[i]=0
            next
        }
        for (j in 1:(n+100)){
            if (ur[i] > x[j] & ur[i]<=x[j+1]){
                nbn[i]=j
                break
            }
        }
    }
    max<-max(nbn)
    min<-min(nbn)
    mean <- mean(nbn)
    var <- var(nbn)
    bins = c(-0.5:(max(nbn)+0.5))
    plot = hist(nbn, breaks = bins, freq = F)
    y <- 0:max(nbn)
    lines(x = y, y = dnbinom(y,n,p), type = "h", col = "red", lwd = 3)
    return(list(max = max, min = min, nbn = nbn, hist = plot, mean = mean, var = var))
}
```

Then, we can use the function to find histogram, mean and variance of the negative binomial random variables:

```
nr <- nbinr(5,1/3,1000)
```




```
c(nr$mean, 5*(1-1/3)/(1/3))
```

```
## [1]  9.96 10.00
```

```
c(nr$var, 5*(1-1/3)/(1/3)^2)
```

```
## [1] 28.23463 30.00000
```

It seems that the variance and histogram are not quite closed to the theoretical values when sample size is only 1000, though the mean is quite close.

Question 3 (CB5.54)

Part a

For this question, I firstly generated 5000 simulation of poisson random variables with sample size is 13 which is the same as the sample data we have. The mean parameter of the 5000 simulation is assumed to be the mean of the sample data, whcih can be calculated as such:

```
data <- c(158,143,106,57,97,80,109,109,350,224,109,214,84)
mean <- mean(data)
mean
```

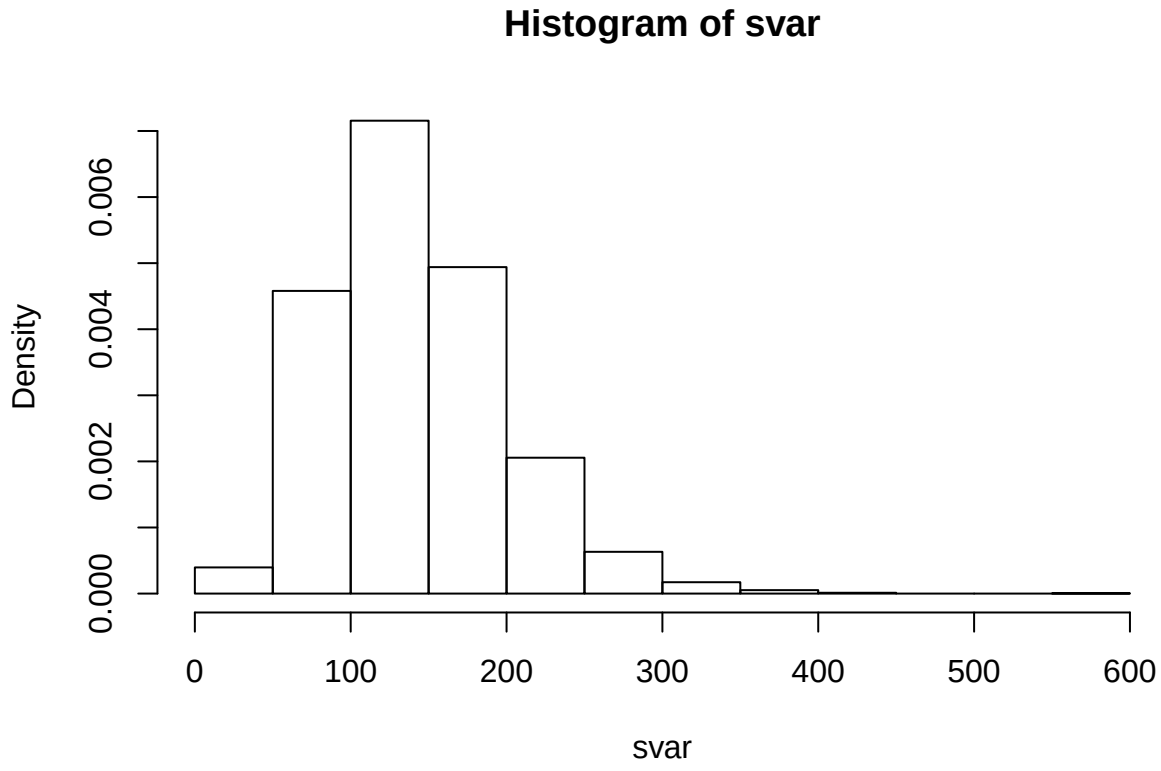
```
## [1] 141.5385
```

After we found the mean, we can generate 5000 simulation of possison samples with sample size 13 by following codes:

```
set.seed(1000)
svar <- rep(0,5000)
for ( i in 1:5000){
    poir <- rpois(13,lambda = mean)
    svar[i] <- var(poir)
}
```

After that, we can have a check of the histogram of the simulated sample variance and calculating the probability that we can have a sample with the sample variance larger than what we observed:

```
hist(svar, freq = F)
```




```
var <- var(data)
var
```

```
## [1] 6370.603
```

```
prob <- length(svar[svar>var])/5000
prob
```

```
## [1] 0
```

The probability here we have is 0, which indicates that we should not expect such a large sample variance of the observation under the possion assumption. Hence we should consider about other distributions such as negative binomial.

Part b

If we assumed the data followed negative binomial distribution, then we can estimate the two parameters using method of moments: (i.e. sample mean and sample variance)

```
r = mean^2/(var-mean)
r
```

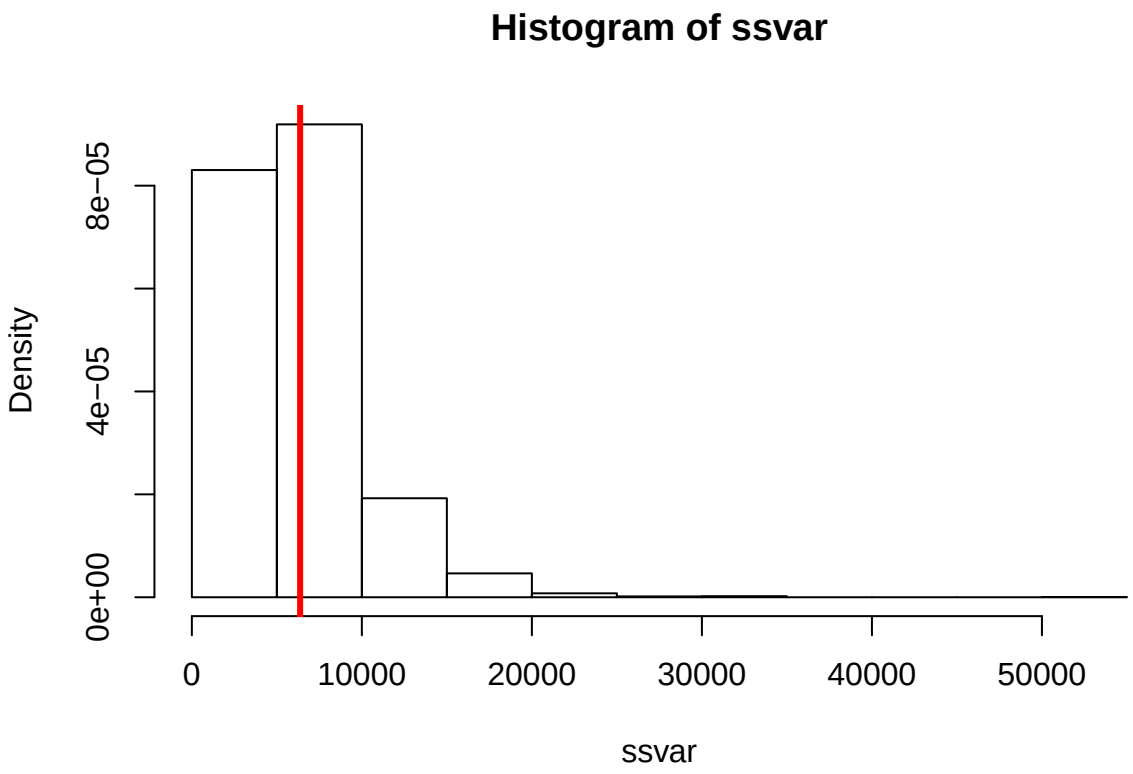
```
## [1] 3.216075
```

```
pr = 1/(1+mean/r)
pr
```

```
## [1] 0.02221744
```

In addition, we can repeat what we have done in part a but replace the possion assumption to be negative binomial assumption:

```
set.seed(1000)
ssvar <- rep(0,5000)
for ( i in 1:5000){
    poir <- rnbinom(13, size =r,  prob = pr)
    ssvar[i] <- var(poir)
}
hist(ssvar, freq = F)
abline(v = var, col = "red", lwd = 3)
```




```
prob2 <- length(ssvar[ssvar>var])/5000
prob2
```

```
## [1] 0.4024
```

The probability here is about 0.4 which is reasonable.

Question 4 (CB5.68)

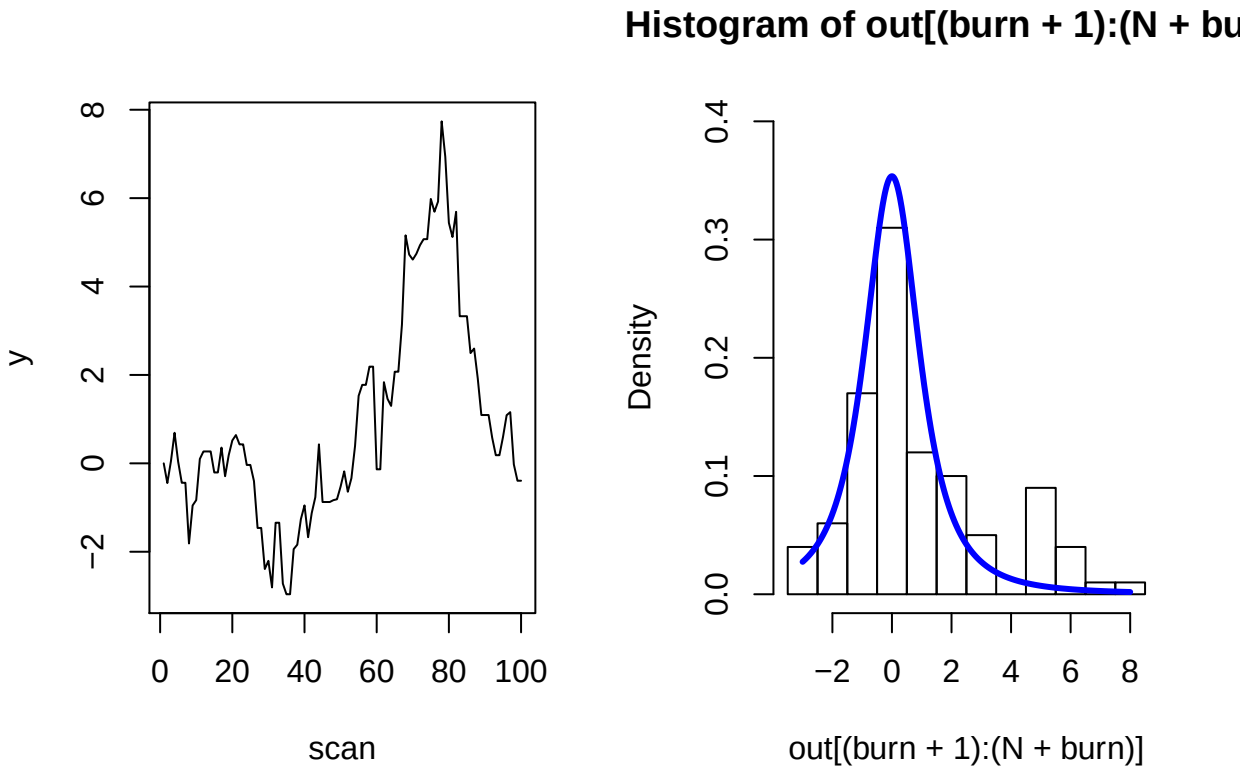
In Question 4 I used three different ways to generate the t distribution with ν.

Part a

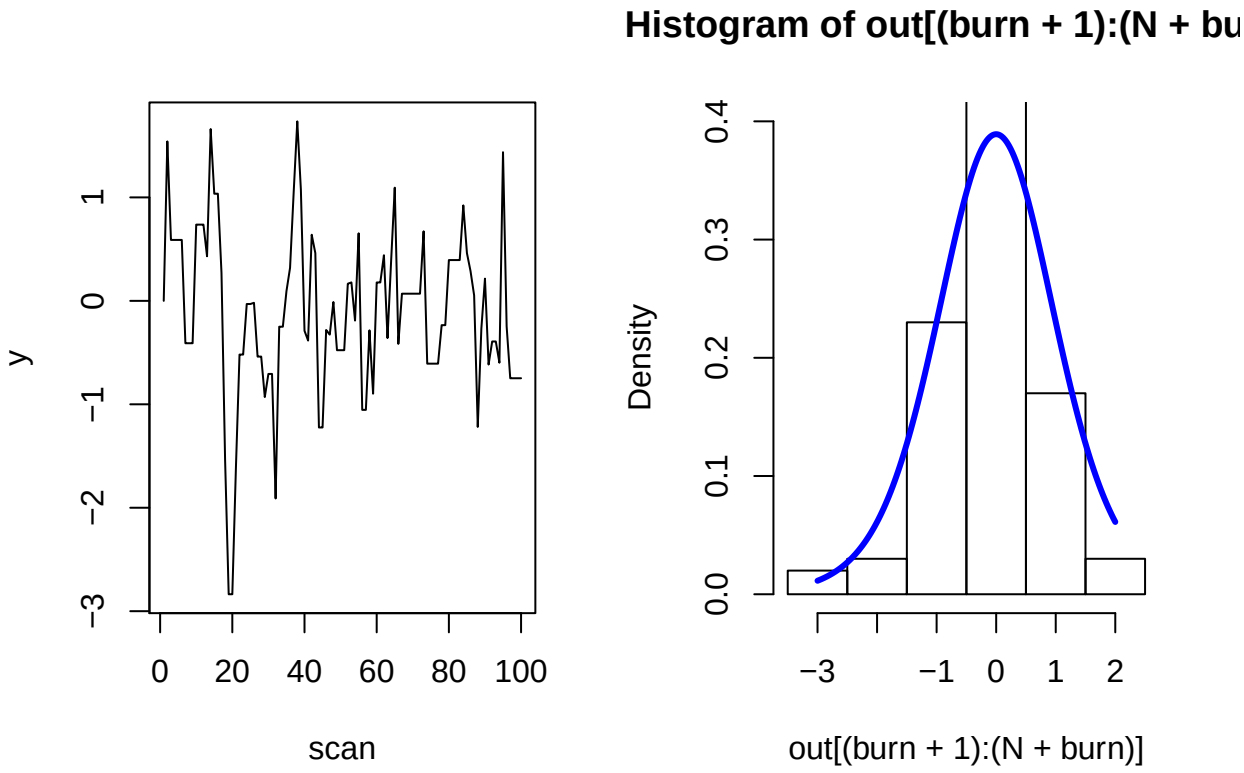
The first method I used is Metropolis algorithm MCMC method. Here we want to generate t distributed random variables based on normal density (normal random variables).

First of all, I selected a starting value and generate a normal random variables. Then, I computed the ratio of densities of target variable at the normal random variable point and the starting point. If the ratio is larger than 1, I will accpet the new variable from normal distribution, if the ratio is smaller than 1, I need to generate another uniform random variable. If the uniform random variable is smaller than the ratio, I will accept the new point, but if the variable is larger than the ratio, the new point is not accpeted, we will still accpet the last point (i.e. the starting point for the first time). Lastly, I need to follow this steps to generate an approximate t distribution with degree of freedom is ν.

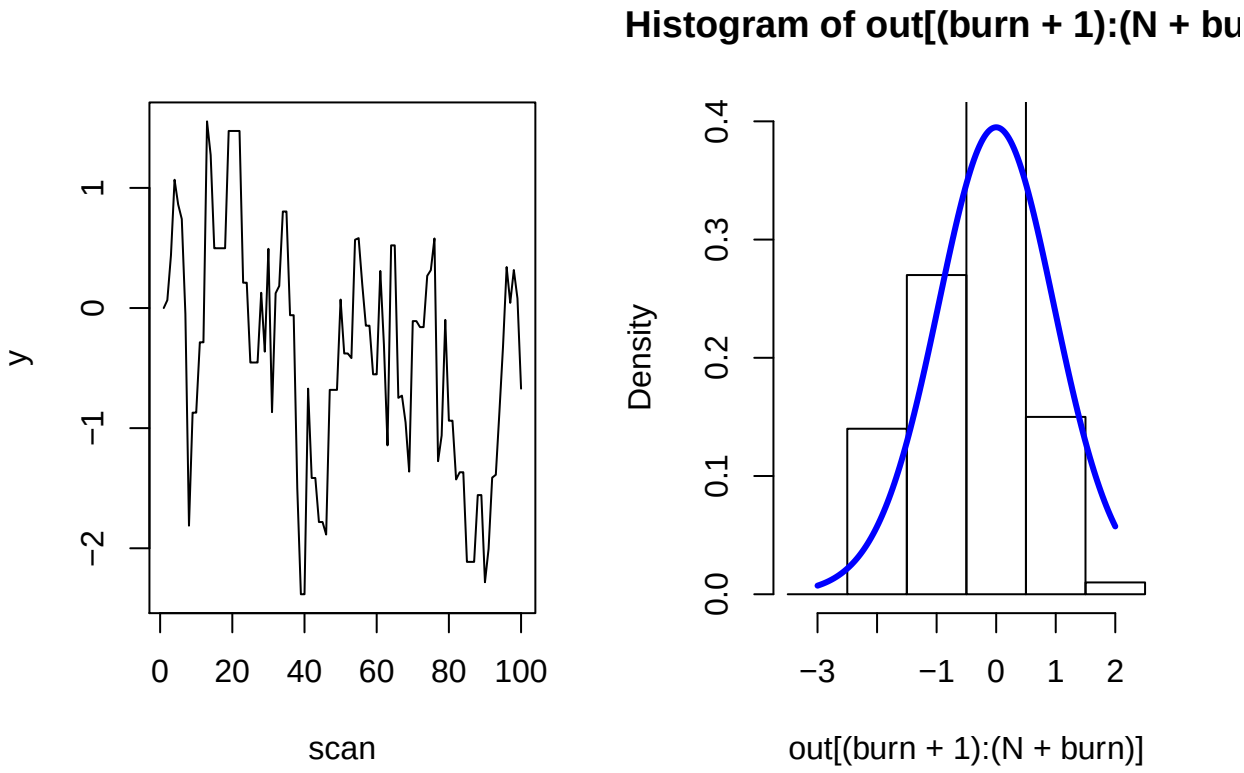
```r
set.seed(1000)
mcmct <- function(df, N, delta, burn){
    out <- rep(0,N+burn)
    acc <- 0

    ##density
    #pt

    ##starting value
    y <- 0
    out[1] <- y

    ##tuning parament
    delta <- 1

    ##mcmc
    for (i in 2:N+burn){
        y.star <- rnorm(1, mean = y, sd = 1*delta)
        r <- dt(y.star, df)/dt(y, df)
        rho <- min(r,1)

        if(runif(1) <= rho){
            y <- y.star
            acc <- acc + 1
        }
        out[i] <- y
    }
    par(mfrow=c(1,2))
    bins <- (floor(min(out[(burn+1):(N+burn)]))-0.5):(ceiling(max(out[(burn+1):(N+burn)])+0.5))
    plot1 <- plot(out[(burn+1):(N+burn)], type = "l", ylab = "y",xlab = "scan")
    plot2 <- hist(out[(burn+1):(N+burn)], breaks = bins, prob = T, ylim =c(0,0.4))
    y <- seq(floor(min(out[(burn+1):(N+burn)])), ceiling(max(out[(burn+1):(N+burn)])), by = 0.01)
    lines(y, dt(y, df), type = "l", col = "blue", lwd = 3)
    #lines(y, dnorm(y,0,1), type = "l", col = "red", lwd = 3)
    return(list(out = out[(burn+1):(N+burn)], plot = plot1, hist = plot2, acc = acc))
}

mc1 <- mcmct(2, 100, 0.1, 100)
```




```
mc2 <- mcmct(10, 100, 0.1, 100)
```




```
mc3 <- mcmct(25, 100, 0.1, 100)
```



Part b

In order to use Accept/Reject method, we need a starting distribution, where I used Cauchy distribution.

Firstly, I need to calculate the maximum value of the ratio of the target t density and Cauchy density. Which could be defined as M:

$$M = \sup_x \frac{f_T(x)}{f_C(x)} < \infty$$

Then, I need to generate a pair of random variables, one from the starting distribution (i.e. Cauchy distribution), denoted as V and the other from the uniform distribution, denoted as U. Following that, we need to compare the second random variable U to $\frac{1}{M} f_T(V)/f_C(V)$ and the first random variable V will be accepted if and only if:

$$U < \frac{1}{M} f_T(V)/f_C(V)$$

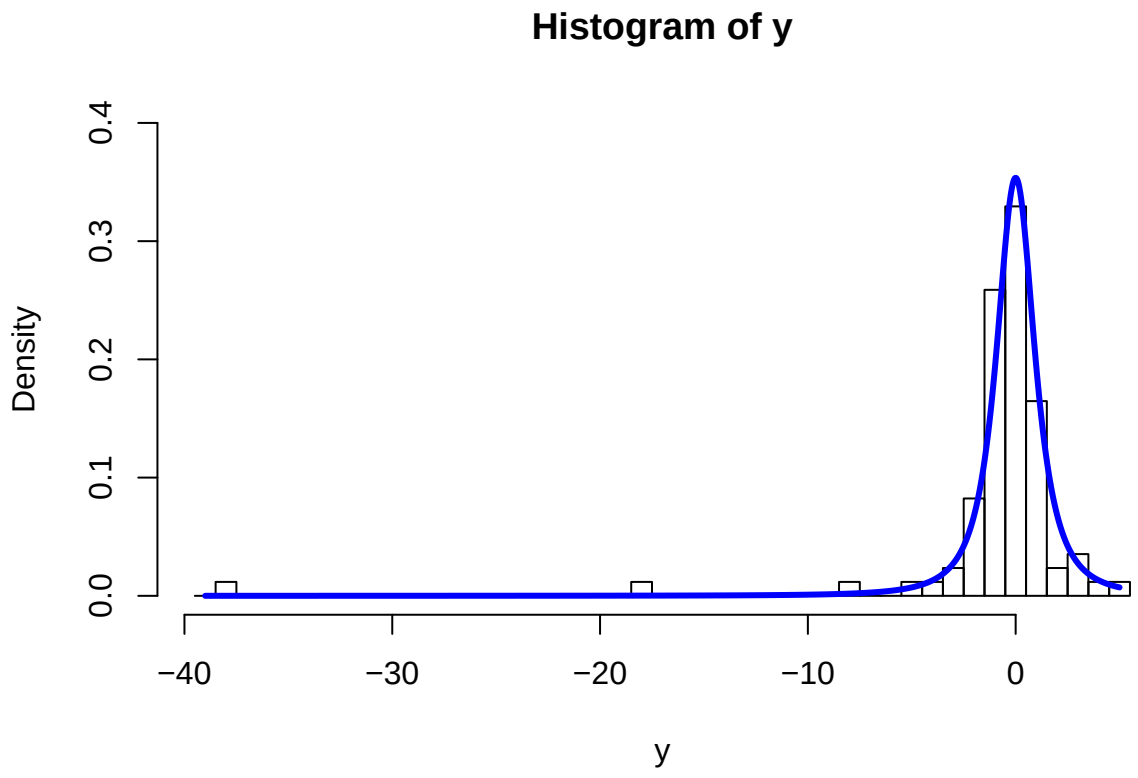
Following is the code to generate Student's t distribution from Cauthy distribution:

```
set.seed(1000)
art <- function(df, N){
    x <- seq(0,1,by = 0.001)
    M <- max(dt(x,df = df)/dcauchy(x,0,1))
    y <- NULL
    for (i in 1:N){
        u <- runif(1,0,1)
        v <- rcauchy(1,0,1)
        if (u*M*dcauchy(v)<dt(v,df = df)){
```

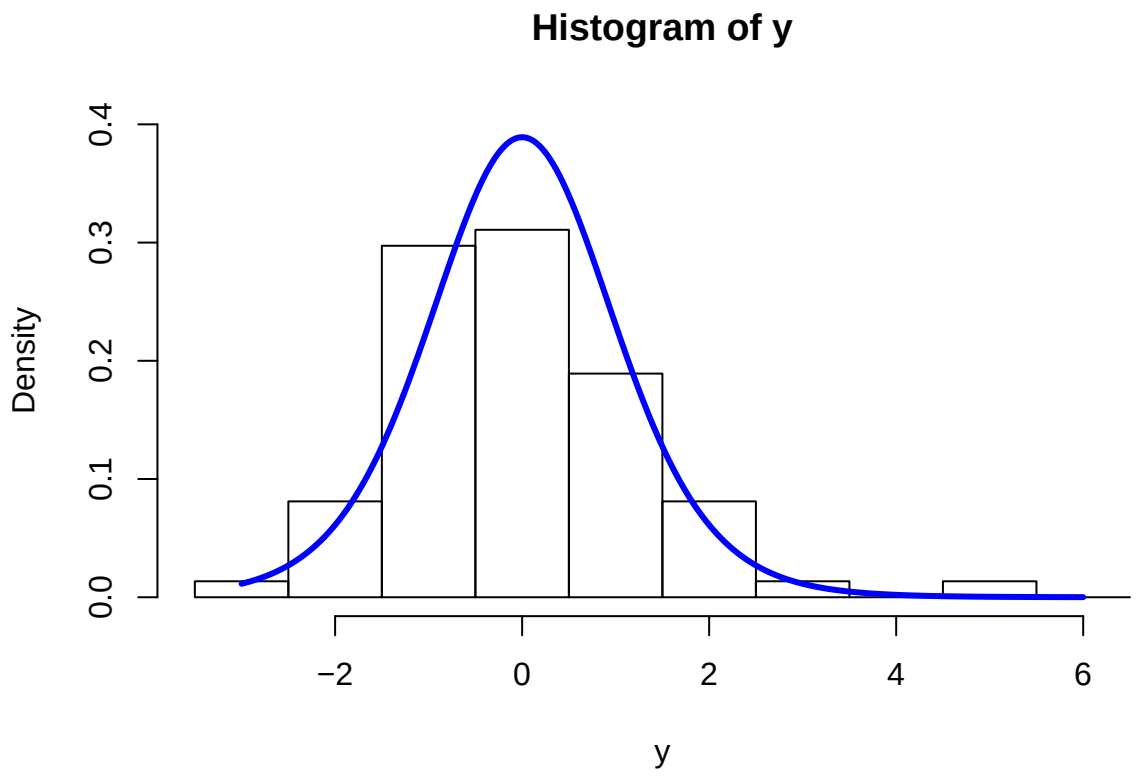
```
            y.i <- v
            y <- c(y,y.i)
        }
    }
    par(mfrow = c(1,1))
    num <- length(y)
    bins <- (floor(min(y))-0.5):(ceiling(max(y))+0.5)
    plot <- hist(y, breaks = bins, prob = T, ylim =c(0,0.4))
    z <- seq(floor(min(y)), ceiling(max(y)), by = 0.01)
    lines(z, dt(z, df), type = "l", col = "blue", lwd = 3)
    return(list(out = y, num = num, plot = plot))
}
```

Then, we can compare the histograms with the theoretical densities:

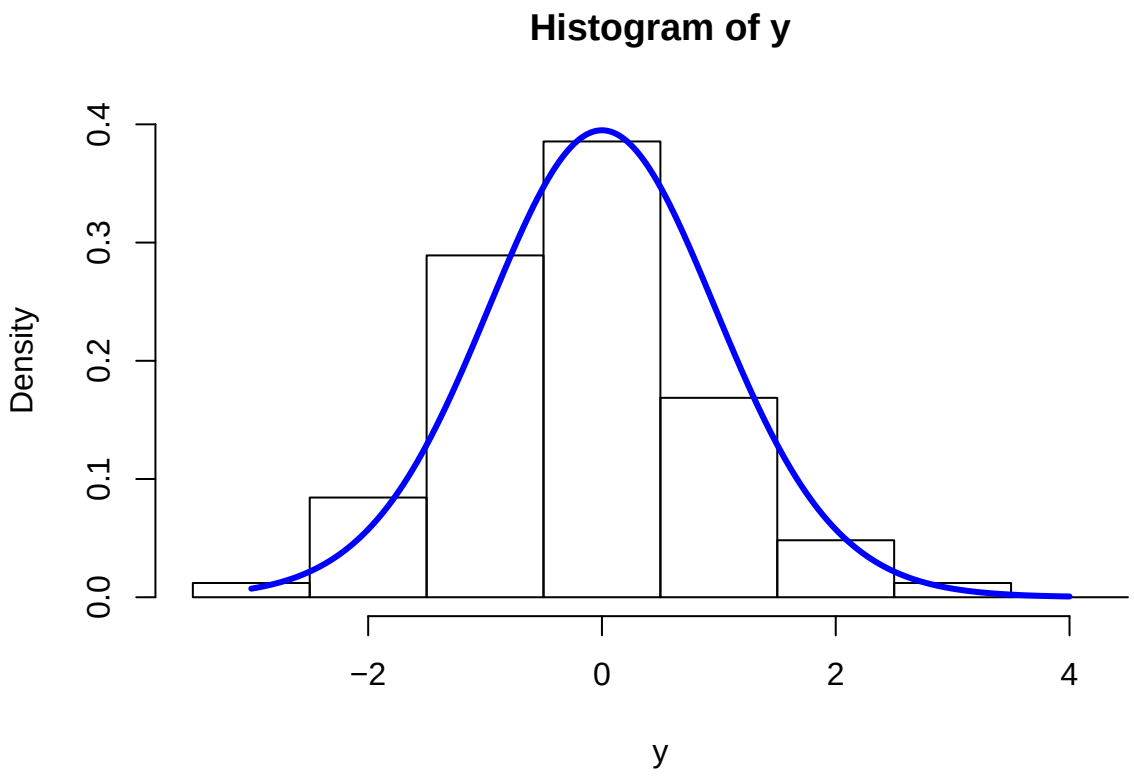
```
ar1 <- art(2,100)
```




```
ar2 <- art(10,100)
```




```
ar3 <- art(25,100)
```




```
c(ar1$num, ar2$num, ar3$num)
```

```
## [1] 85 74 83
```

Here the sample size is smaller than 100 as we need to reject some outcomes with the probability defined by the above ratio.

Part c

To generate t distribution, I used the transformation from standard normal distribution to get the target distribution:

$$T = \frac{v_i}{\sqrt{\frac{1}{\nu}\sum_{i=1}^{\nu} z_i^2}}$$

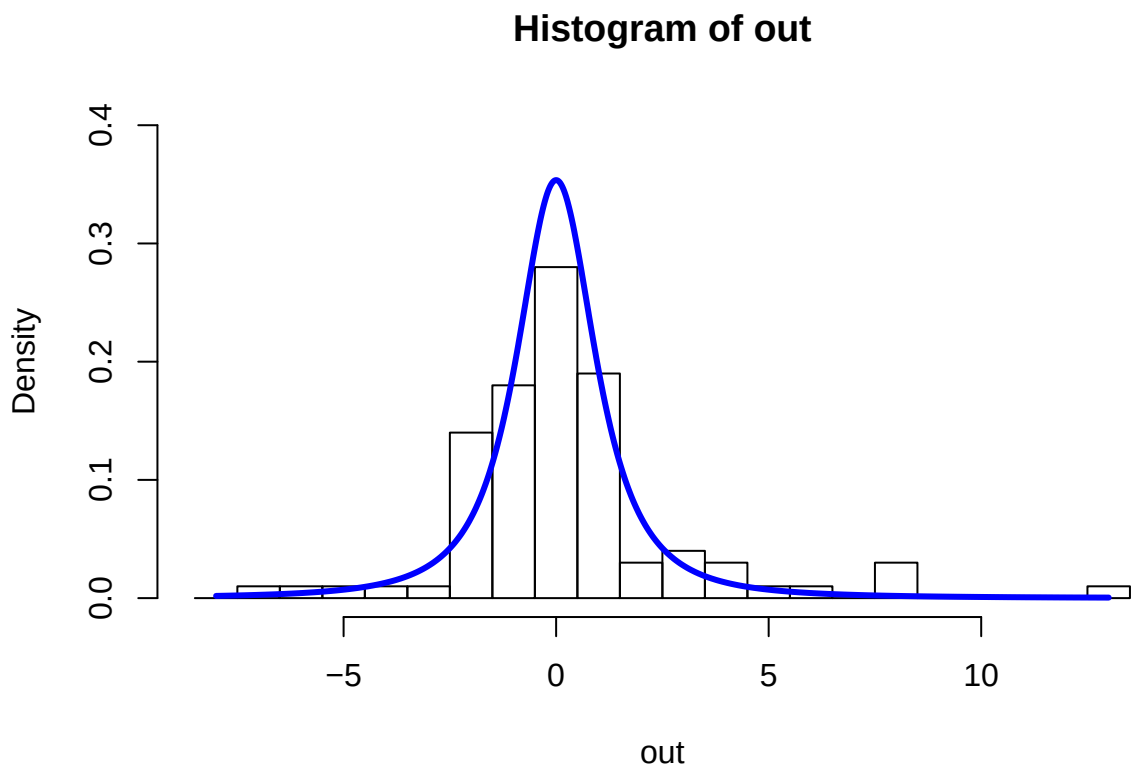
where z_i and v_i are iid normally distributed and ν is degree of freedom of the target t distribution. Therefore we can use the following code to generate t distribution:

```
set.seed(1000)
transt <- function(df ,N){
    out <- rep(0,N)
    for (i in 1:N){
        v <- rnorm(1,0,1)
        zsqr <- rep(0,df)
```

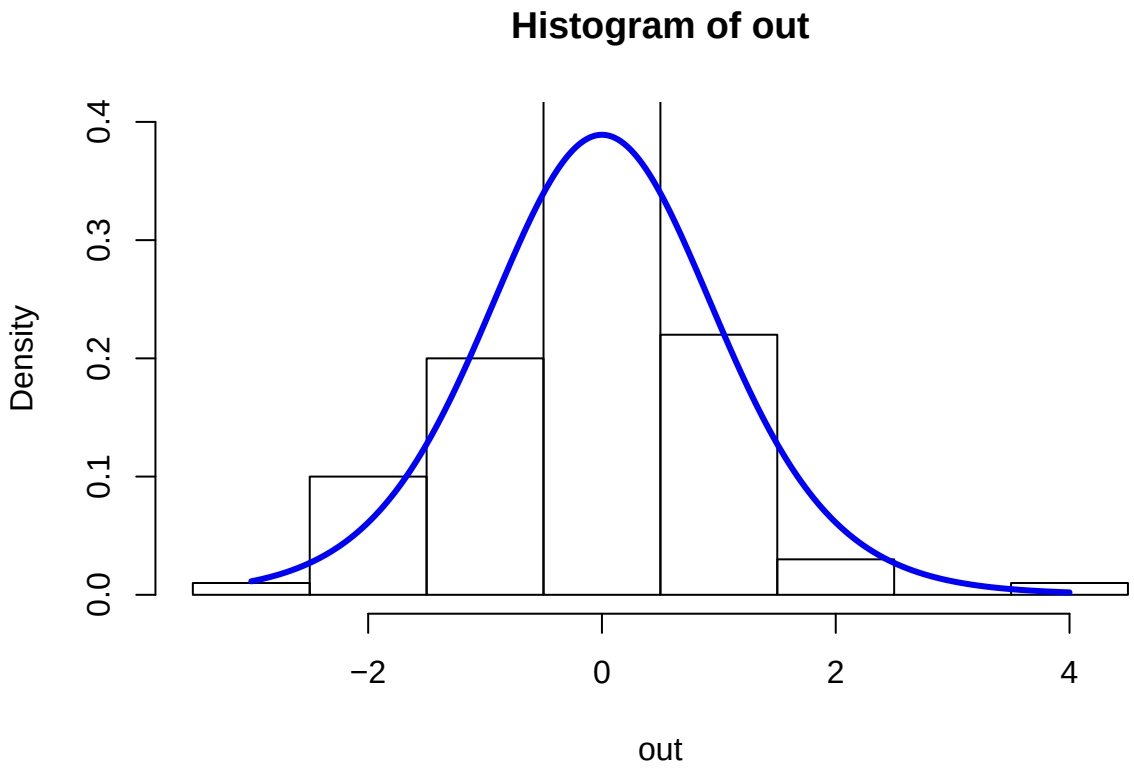
```
        for (j in 1:df){
            z <- rnorm(1,0,1)
            zsqr[j] <- z^2
        }
        out[i] <- v/sqrt(sum(zsqr)/df)
    }
    par(mfrow=c(1,1))
    bins <- (floor(min(out))-0.5):(ceiling(max(out))+0.5)
    plot <- hist(out, breaks = bins, prob = T, ylim = c(0,0.4))
    x <- seq(floor(min(out)), ceiling(max(out)), by = 0.01)
    lines(x, dt(x, df), type = "l", col = "blue", lwd = 3)
    return(list(out = out, plot = plot))
}
```

For the cases that degree of freedom is 2, 10 and 25 we can plot the histograms and compared it with the theoretical densities.

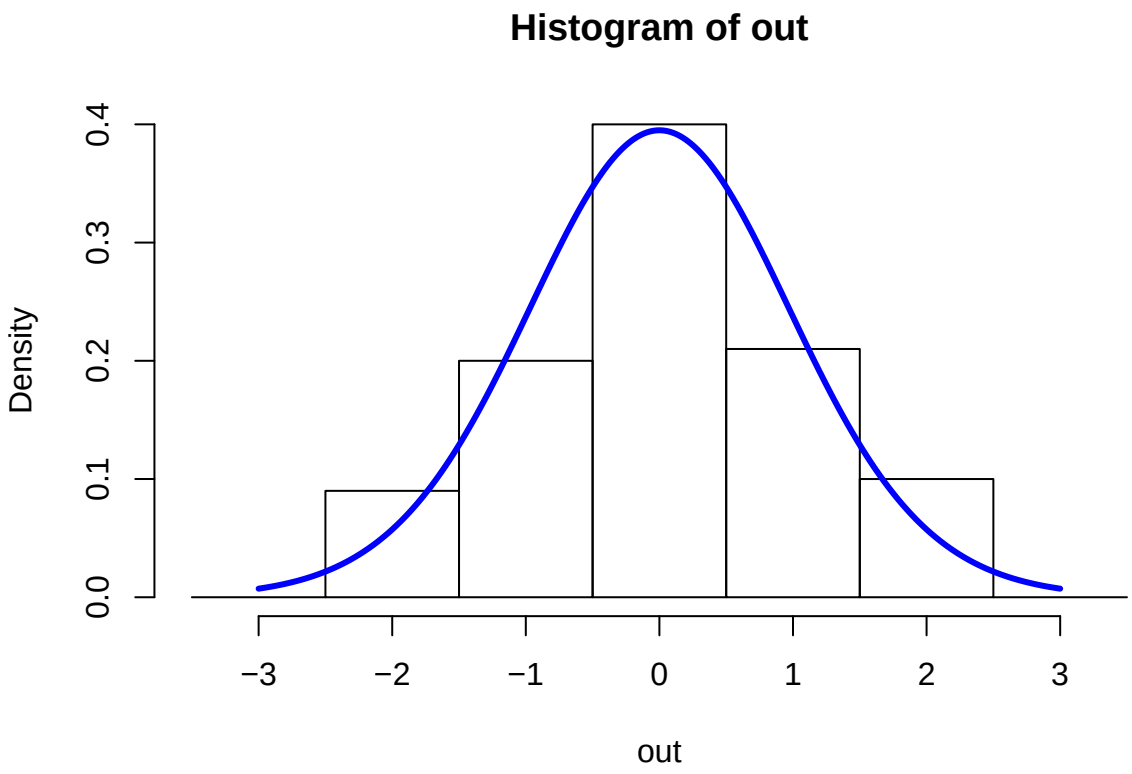
```
trans1 <- transt(2,100)
```



```
trans2 <- transt(10,100)
```




```
trans3 <- transt(25,100)
```



Part d

When degree of freedom is 2, mcmc method result is the worst since it was not really comverge to the target t distribution when sample size is quite small (i.e. 100). When df is 10 and 25, the result of mcmc is better than the case of 2, but still not good.

Generally speaking, the transformation method can generate the best t distribution when df is 2, 10 and 25. The Accept/Reject method can also generate a comparable outcome when df is 2, but for df is 10 or 25, the result is not as good as that from transformation method. Another drawback of Accept/Reject is that we have a probability to reject some outcome so that with sample size set to be 100 we can only generate 85, 74 and 83 Student's t distribution random variable for df is 2, 10 and 25.