Inverse-CDF and Acceptance-Rejection

Sreenand Sasikumar

13/02/2020

Question 1: Cluster sampling

1.1

Importing the population data into R.

1.2

```
## The function that selects 1 city from the whole list by the probability scheme randomly is :
```

```
##      ex_city_nm ex_city_pop
## [1,] "Sigtuna"  "39219"
```

1.3

```
## The function that selects 20 city from the whole list by the probability scheme randomly is :
```

```
##       ex_city_nm    ex_city_pop
##  [1,] "Nacka"       "88085"
##  [2,] "Linköping"   "144690"
##  [3,] "Borlänge"    "48681"
##  [4,] "Halmstad"    "91087"
##  [5,] "Oxelösund"   "11126"
##  [6,] "Skövde"      "50984"
##  [7,] "Ronneby"     "28416"
##  [8,] "Klippan"     "16382"
##  [9,] "Kalmar"      "62388"
## [10,] "Hudiksvall"  "36848"
## [11,] "Göteborg"    "507330"
## [12,] "Lidingö"     "43445"
## [13,] "Trollhättan" "54873"
## [14,] "Stockholm"   "829417"
## [15,] "Kalix"       "16926"
## [16,] "Ängelholm"   "39083"
## [17,] "Ekerö"       "25095"
## [18,] "Ragunda"     "5609"
## [19,] "Umeå"        "114075"
## [20,] "Katrineholm" "32303"
```

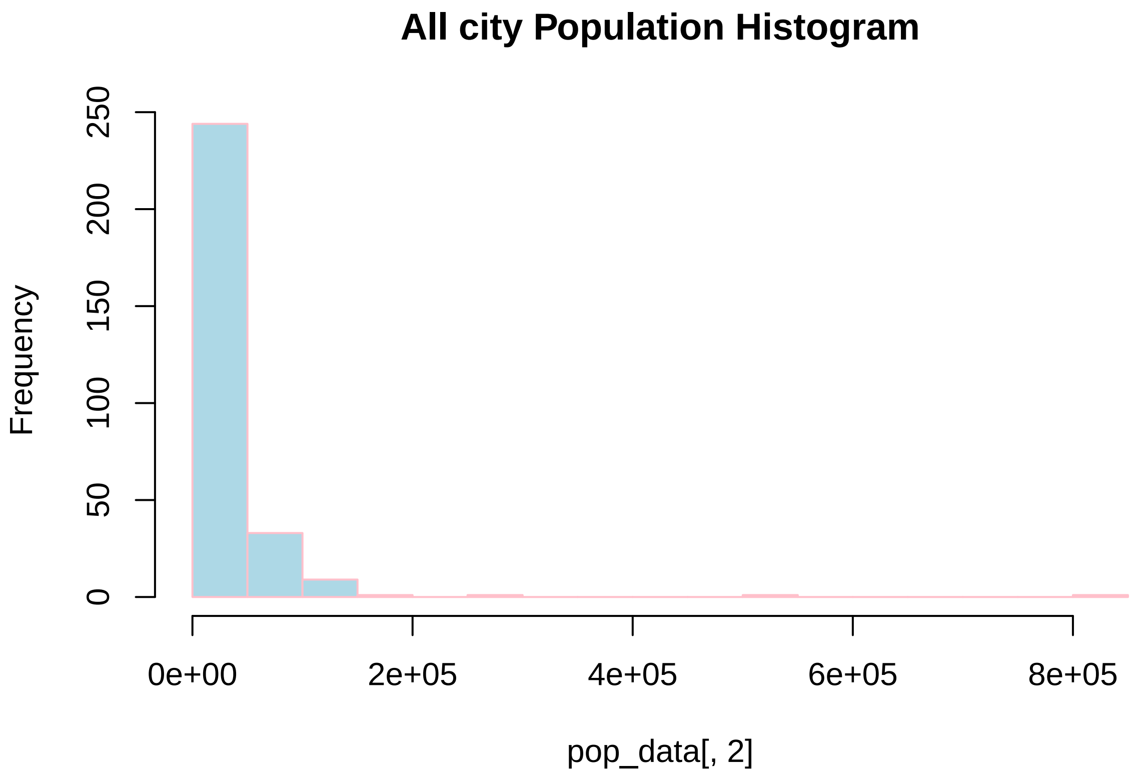
1.4

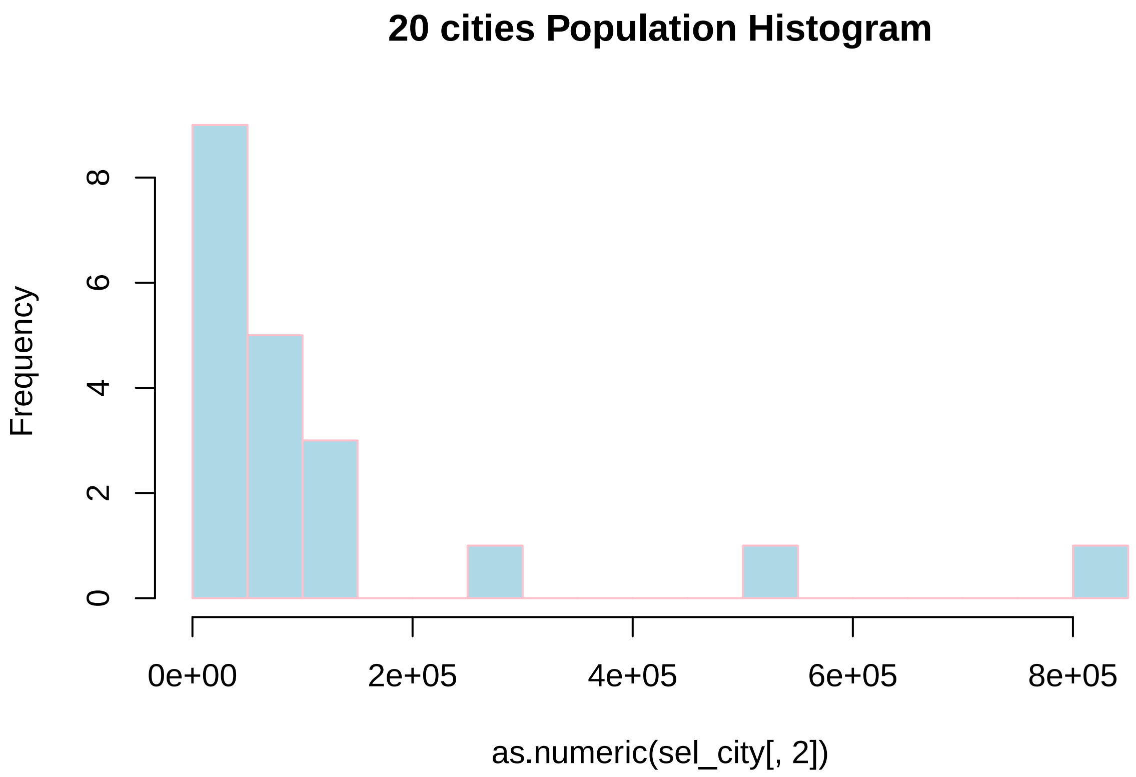
```
## The range of the Population of cities is :
```

```
## [1]   2500 829417
## Number of cities whose population is greater than 15000

## [1] 149

## Count of number of cities having population greater than 15000 when the function is ran 10 times

##  [1] 15 17 17 17 13 15 16 17 17 13
```

As we can see, the range of population of the cities is between 2500 and 829417 and number of cities which has population greater than 15000 is 149 which is almost 50% of the overall cities in the data set. After running the functions multiple times to pick 20 cities, almost most of the time more than 15 cities are being picked and the reason could be that, the probability of population is high and hence given more preference in picking these cities.

1.5


All city Population Histogram
Frequency
0
50
100
150
200
250
0e+00
2e+05
4e+05
6e+05
8e+05
pop_data[, 2]


20 cities Population Histogram
Frequency
0
2
4
6
8
0e+00
2e+05
4e+05
6e+05
8e+05
as.numeric(sel_city[, 2])

From the above histograms, it is evident that in both the histograms, highly populated cities are present and plotted. A random generator to pick a city is totally depend on population of that particular city, ence higher the population probability of picking the city.

Question 2: Different distributions

2.1

The Probability Density function of a Double exponential function with

$$\mu = 0, \alpha = 1$$

is given by,

$$f_X(x \mid \mu = 0, \alpha = 1) = \frac{1}{2} exp(-x) \; \forall x \geq \mu$$

$$f_X(x \mid \mu = 0, \alpha = 1) = \frac{1}{2} exp(x) if x < 0$$

In order to find the Inverse CDF function to generate random numbers, we need Cumulative Distribution Function of the Probability Density Function, hence the CDF of

$$f_X(x \mid \mu = 0, \alpha = 1)$$

is given by,

$$F_X(x) = \left(\int_{-\infty}^{x} \frac{1}{2} exp(x) \, dx \right) \forall x < 0$$

and

$$F_X(x) = \left(\int_{-\infty}^{0} \frac{1}{2} exp(-x) dx + \int_{0}^{x} \frac{1}{2} exp(-x) dx \right) \; \forall x \geq 0$$

$$F_X(x) = \frac{1}{2} exp(x) \;\; \forall x < 0$$

and

$$F_X(x) = 1 - \frac{1}{2} exp(-x) \;\; \forall x \geq 0$$

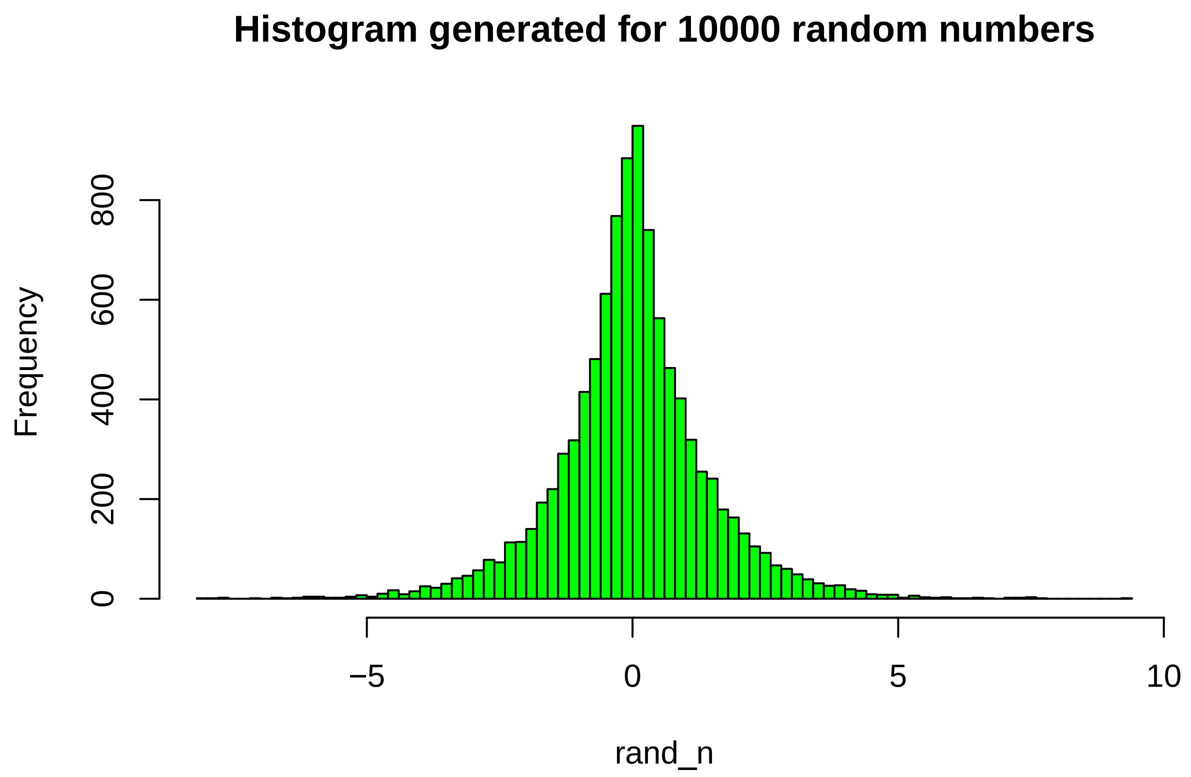
Now that we have got Cumulative Distribution Function of a Double Exponential Distribution, we can find Inverse functions of the CDF's, The Inverse CDF of Double Exponential function is given by,

$$F_X(y) = \log(2x) \; \forall \; 0 < y < \frac{1}{2}$$

and

$$F_X(y) = -\log(2 - 2x) \; \forall \; \frac{1}{2} < y < 1$$

Using the above Inverse CDF functions, the 10000 random numbers will be generated from Unif(0,1)



From the above histogram it is evident that the result looks reasonable since it does resemble a regular Normal distribution curve. It can also be noted that this histogram has wider tails than regular normal distribution tail with mean 0.

2.2

According to Acceptance/Rejection method, in order to generate Normal Random numbers with

$$\mu = 0, \sigma = 1$$

, the following formula and steps are followed:

Check the random number U generated from Unif(0,1) holds good the below condition,

$$U \leq \frac{f_X(Y)}{cf_Y(Y)}$$

where

$$f_X(Y) = Target\ Density$$

,

$$f_Y(Y) = Majorizing\ Density$$

,

$$c = Majorizing\ Constant$$

In order to check for the above condition, the value of c has to be calculated. c can be defined as the maximum value of the ratio of

$$\frac{f_X(Y)}{f_Y(Y)}$$

and the output is the least value c can have.(Optimal Value for c is the least)

The

$$f_X(Y)$$

and

$$f_Y(Y)$$

are the regular Probability Density functions and in this case the Target Density

$$f_X(Y)$$

is Normal Distribution with

$$\mu = 0, \sigma = 1$$

and Majorizing Density is the Double Exponential function with

$$\mu = 0, \alpha = 1$$

. The maximum ratio can be achieved only by taking derivating of the ratio and setting it to 0. By doing so, we arrive at below formula for c,

$$f_X(Y) = \frac{1}{\sqrt{2\pi}} exp(\frac{-x^2}{2})$$

$$f_Y(Y) = \frac{1}{2} exp(-|x|)$$

On simplifying the ratio, we arrive at,

$$\frac{\sqrt{2}}{\sqrt{\pi}} exp(x - \frac{x^2}{2})$$

In order to to get maximum ratio, we have to take the partial derivatives of the ratio with respect to x and the value of x has to be altered and in this case for maximum ratio, x can have only have values -1 and 1. Hence by substituting the values, the final formula for c can be written as,

$$c = \sqrt{\frac{2}{\pi}} exp^{\frac{1}{2}}$$

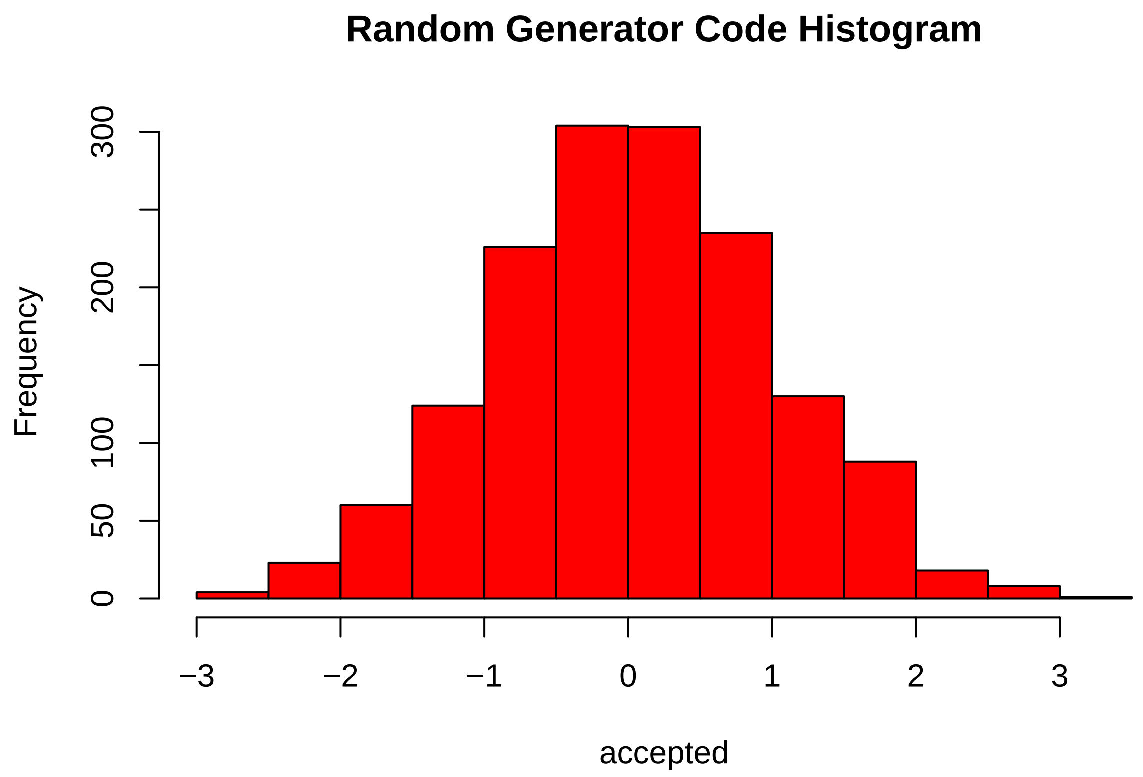
Now that the value of c has been calculated, let us substitute the value in main formula for U and random value based on the condition at each iteration for 2000 times. That is, 2000 random values has to be generated using this acceptance and rejection method.

If the condition is passed, a random number thus generated will be stores in a variable *output* and if the condition fails, the *output* variable will be stored with *999* value. To be precise, if the main condition for U succeeds, the random number is accepted and if the condition fails, the random number is rejected and stores as *999*.

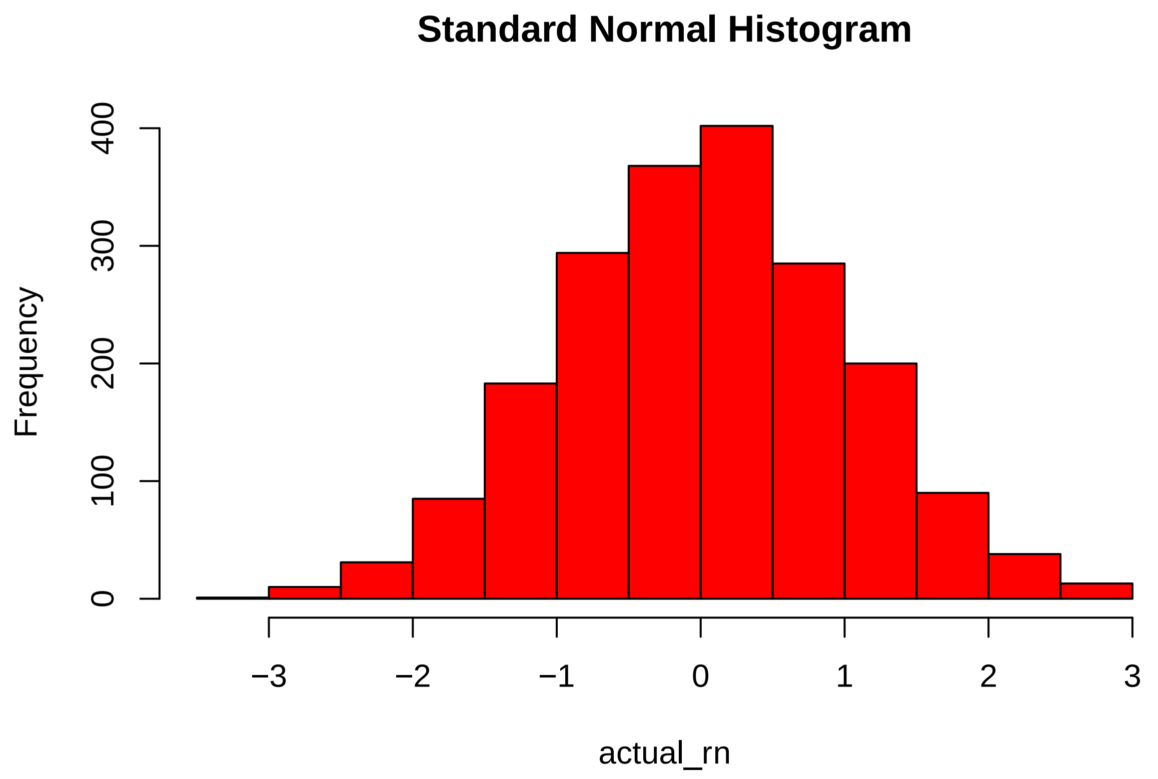
```
## Expected rejected rate is found to be :

## [1] 0.2398265

## Average rejection rate is found to be :

## [1] 0.238

## To generate 2000 random numbers from the code, it had to face 476 rejections, which means 476 random
```

It is seen that both the Expected rejection rate and Average rejection rate are almost very close to equal to each other.

Below is the Histogram that was plotted based on the Acceptance/Rejected Method to generate 2000 Normal random numbers.



Below is the Histogram that was plotted based on the Standard Normal *rnorm* function to generate 2000 Normal random numbers.



From both the plotted Histograms, it is evident that both resembles a regular Normal Distribution curve with mean 0 and sd 1. Hence the random number generated from the code is doing a good job in generating numbers which is very close to normality as compared with standard norm.

Appendix

```
knitr::opts_chunk$set(echo = TRUE,warning = FALSE, message = FALSE)
pop_data <- read.csv2(file.choose())
clus_sampling <- function(data,num_cities)
{
ex_city_pop <- numeric()
ex_city_nm <- vector()
ind <- vector()
for(i in 1:num_cities)
{
  rand_num <- runif(1)
  data[["prob"]] <- data$Population / sum(data$Population)
  data[["cumprob"]] <- cumsum(data$prob)
  ex_city_nm[i] <- as.vector(data[which(data$cumprob > rand_num)[1],1])
  ex_city_pop[i] <- as.vector(data[which(data$cumprob > rand_num)[1],2])
  ind <- which(data$cumprob > rand_num)[1]
  #cat("ind",ex_city_ind)
  #ex_city[i] <- data[ex_city_ind,1]
  #cat("city",ex_city)
  data <- data[-ind,]
  rownames(data) <- 1:nrow(data)
```

```r
}

return(cbind(ex_city_nm,ex_city_pop))
}
cat("The function that selects 1 city from the whole list by the probability scheme randomly is :")
sel_city <- clus_sampling(data = pop_data,1)
sel_city
cat("The function that selects 20 city from the whole list by the probability scheme randomly is :")
sel_city <- clus_sampling(data = pop_data,20)
sel_city
cat("The range of the Population of cities is :")
range(pop_data$Population)
cat("Number of cities whose population is greater than 15000")
length(which(pop_data$Population > 15000))
n <- vector()
for(i in seq(10))
{
  sel_city <- clus_sampling(data = pop_data,20)
  n[i] <- length(which(sel_city[,2] > 15000))
}
cat("Count of number of cities having population greater than 15000 when the function is ran 10 times")
n
hist(pop_data[,2],breaks = 20, col = "lightblue", border = "pink", main = "All city Population Histogram
sel_city <- clus_sampling(data = pop_data,20)
hist(as.numeric(sel_city[,2]),breaks = 20, col = "lightblue", border = "pink",main = "20 cities Populati
rand_n <- numeric(10000)
de <- function(n)
{
for(i in seq(n))
{
  U <- runif(1,0,1)
  if(U >= 0.5)
  {
    rand_n[i] <- -log(2- (2 * U))
  }
  else
  {
    rand_n[i] <- log(2 * U)
  }
}
  return(rand_n)
}
rand_n <- de(10000)
hist(rand_n,breaks = 100,col = "green",main = "Histogram generated for 10000 random numbers")
c <- sqrt(2 * exp(1) / pi)
output <- numeric(2000)
accep_rej_meth <- function(n1)
{
for(i in seq(n1))
{
  U1 <- runif(1,0,1)
  #print(U1)
  rand_x <- de(1)
  prop_trgt_ratio <- (exp(-(rand_x^2) / 2) / sqrt(2 * pi)) / (c * exp(-abs(rand_x)) / 2)
  #print(prop_trgt_ratio)
  if(U1 <= prop_trgt_ratio)
  {
    output[i] <- rand_x

  }
  else
  {
    output[i] <- 999
  }
}
  return(output)
}
accep_rejec <- accep_rej_meth(2000)
accepted <- accep_rejec[-which(accep_rejec == 999)]

cat("Expected rejected rate is found to be :")
expec_rej_rt <- 1 - 1/c
expec_rej_rt

cat("Average rejection rate is found to be :")
r <- (length(accep_rejec) - length(accepted)) / length(accep_rejec)
r

rejected <- length(which(accep_rejec == 999))
cat("To generate 2000 random numbers from the code, it had to face", rejected ,"rejections, which means
hist(accepted,breaks = 15,col = "red", main = "Random Generator Code Histogram")
actual_rn <- rnorm(2000)
hist(actual_rn,breaks = 20,col = "red",main = "Standard Normal Histogram")
```