Documentation: Assignment 6

Abhinav Gupta 150123001

- Q 1 Use the Box-Muller method and Marsaglia-Bray method to do the following:
 - (a) Generate a sample of 100, 500 and 10000 values from N(0, 1). Hence find the sample mean and variance.
 - (b) Draw histogram in all cases.

Code for Box-Muller method (Code for 10,000 values): R

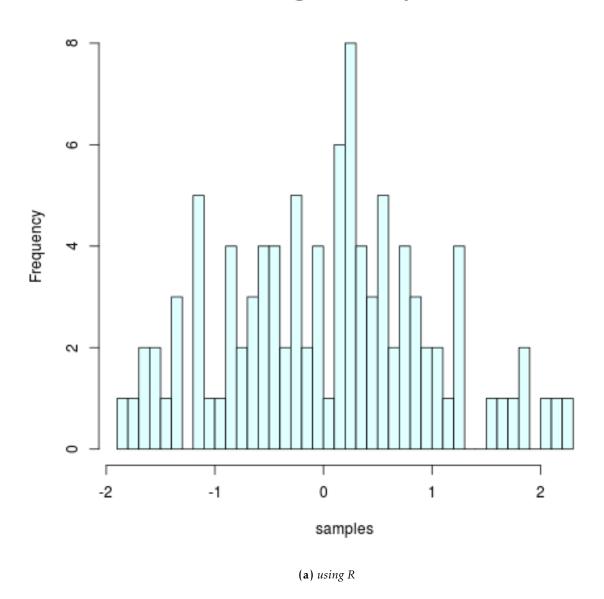
```
1 radius <- function (u) {
      return (-2*log(u))
 3 }
 4 arg <- function (u) {
 5
      return (2*pi*u)
 6
  # change count to 100, 500 and 10,000 as per requirement
  count<-10000
   samples1 < -c()
|10| samples 2 < -c()
11 for (i in 1:count/2) {
      u1<-runif(1)
12
13
      u2<-runif(1)
14
      samples1[i]<-sqrt(radius(u1))*cos(arg(u2))
15
      samples2[i]<-sqrt(radius(u1))*sin(arg(u2))
16 }
17 samples <-c (samples1, samples2)
18 png ("question1-10000-muller.png")
19 hist (samples, breaks=50, col="light cyan", plot=TRUE)
```

Output:

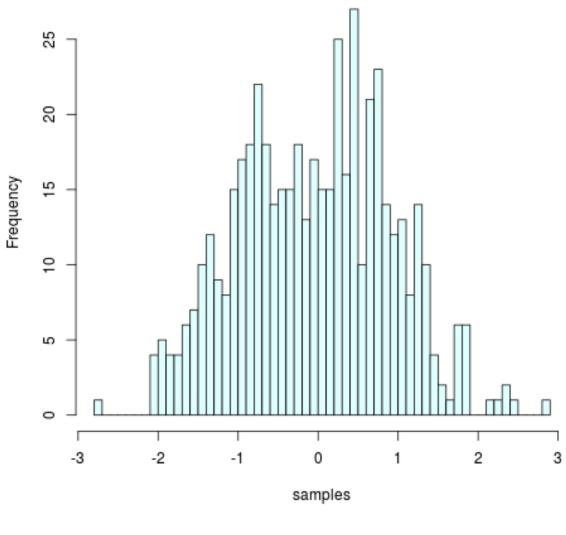
Mean: 0.002538348

Variance: 1.008891

Observation: Graph: 100 values:

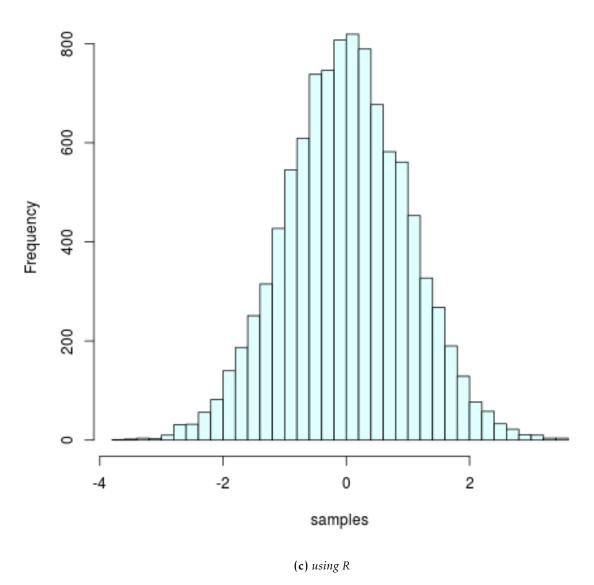


Graph: 500 values:



(b) using R

Graph: 10000 values:



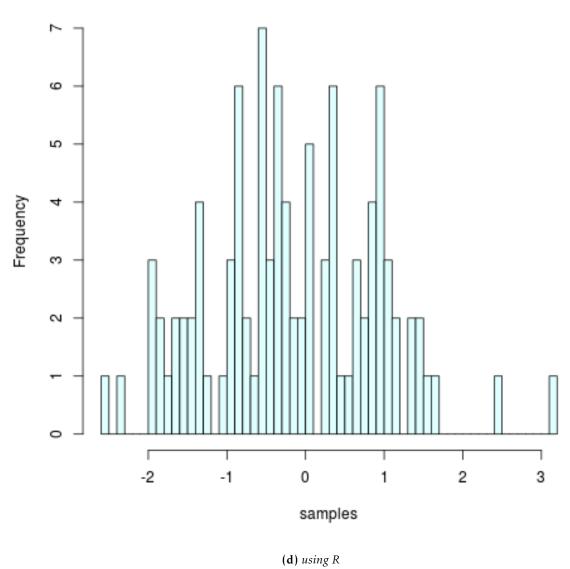
```
11 samples 2 < -c ()
12 i <-1
13 while (1) {
      u1<-runif(1)
14
15
      u2<-runif(1)
      u1 < -2 * u1 - 1
16
17
      u2 < -2 * u2 - 1
18
      x < -square(u1,u2)
19
      if(x>1){
20
          next
21
      }
      y<-intermediate(x)
22
23
      samples1[i] < -u1*y
      samples2[i] < -u2*y
24
      i < -i+1
25
      count<-count-2</pre>
26
27
      if (count==0){
          break
28
29
      }
30 }
31 samples <-c (samples1, samples2)
32 png ("question 1 - 10000 - bray.png")
33 hist(samples, breaks=50, col="light cyan", plot=TRUE)
```

Output:

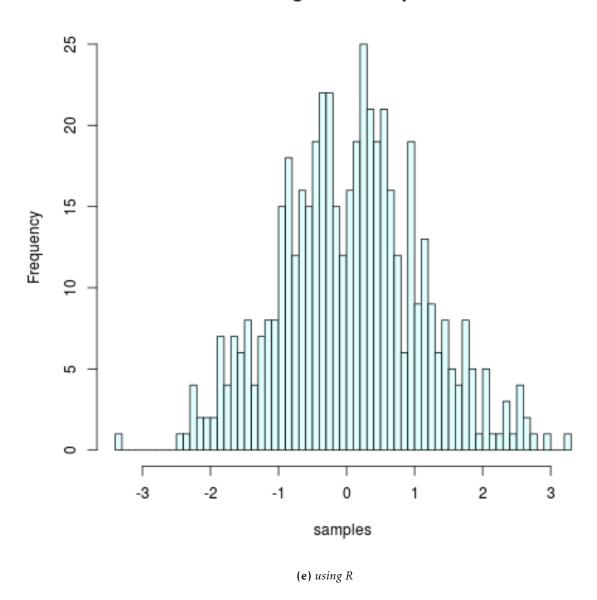
Mean: 0.009609266

Variance: 1.006028

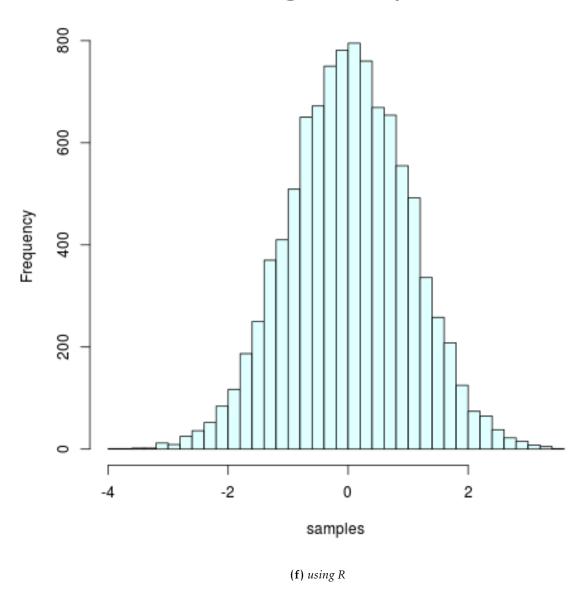
Observation: Graph:



Graph: 500 values:



Graph: 10000 values:

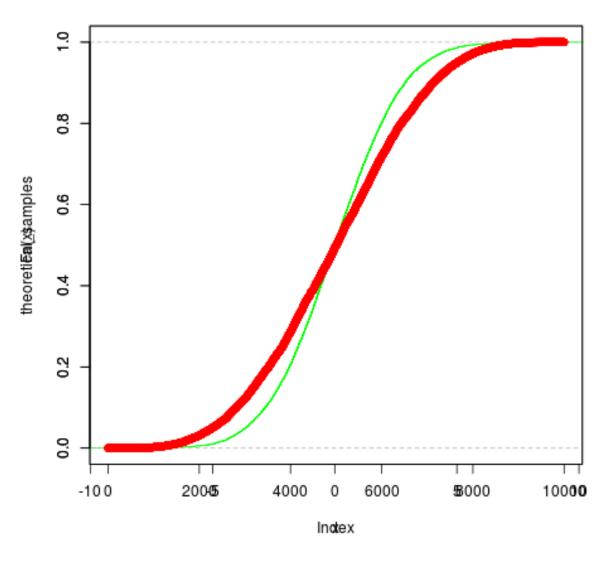


Q 2 Now use the above generated values to generated samples from N(=0, 2=5) and N(=5, 2=5). Hence plot the empirical (from sample with size 500) distribution function and theoretical distribution function in the same plot. (Use R/ you should also try making the step function in C).

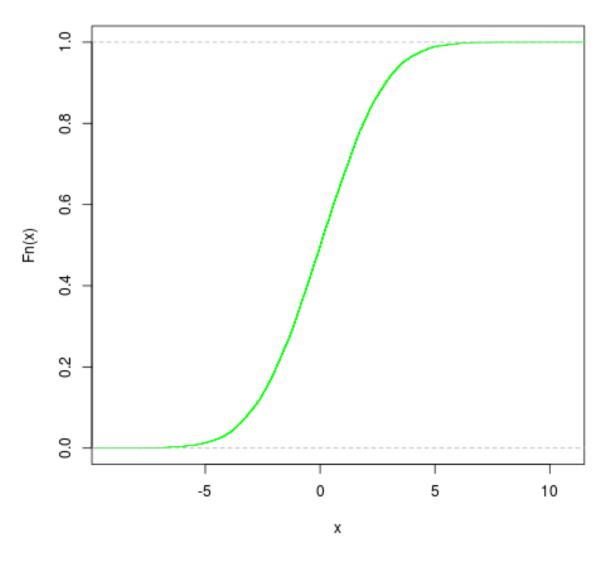
Code for Marsaglia-Bray method mean=0, variance=5: R

```
1 square<-function(u1,u2){
2    return (u1**2+u2**2)
3 }
4 intermediate<-function(x){</pre>
```

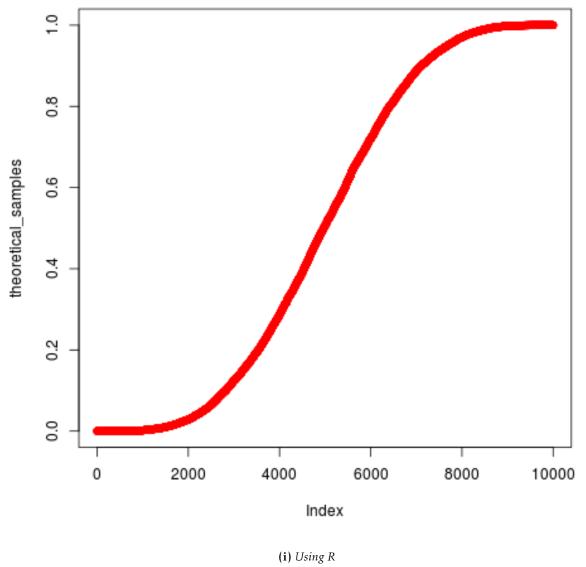
```
return (sqrt(-2*log(x)/x))
 6 }
 7 count<-10000
 8 standard_deviation <-sqrt(5)
 9 mean<-0
|10| samples 1 < -c()
11 samples 2 < -c ()
12 i <-1
13 j <−0
14 while (1) {
      u1<-runif(1)
15
16
      u2<-runif(1)
17
      u1 < -2 * u1 - 1
18
      u2 < -2 * u2 - 1
19
      j < -j + 1
20
      x < -square(u1, u2)
21
      if(x>1){
          next
22
23
      }
24
      y<-intermediate(x)
25
      samples1[i]<-u1*y*standard_deviation+mean
      samples2[i]<-u2*y*standard_deviation+mean
26
27
      i < -i+1
      count<-count-2
28
29
      if (count==0){
30
          break
31
      }
32 }
33 samples <-c (samples 1, samples 2)
34 samples <-sort (samples)
35 cat ("Mean is: ", mean (samples), "\n")
36 cat("Acceptance Probability: ",(i-1)/j,"\n")
37 empirical_samples<-ecdf(samples)
38 png("Empirical(green).png")
39 plot (empirical_samples, col="green")
40 #par (new=TRUE)
41 png("Theoretical(red).png")
42 theoretical_samples <-pnorm(samples)
43 plot (theoretical_samples, col="red")
```



(g) Using R



(h) Using R

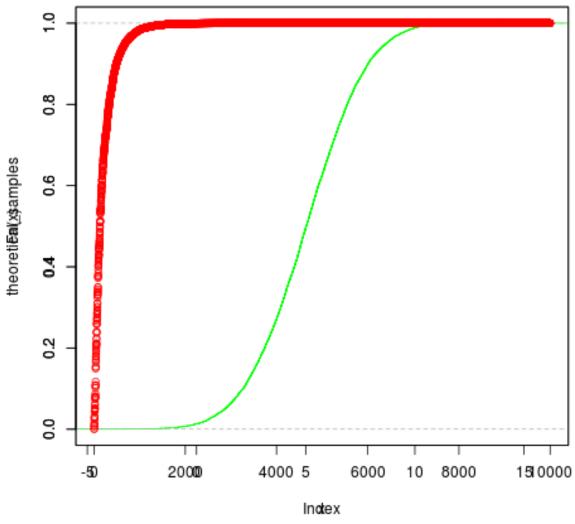


Code for Marsaglia-Bray method mean=5, variance=5 : R

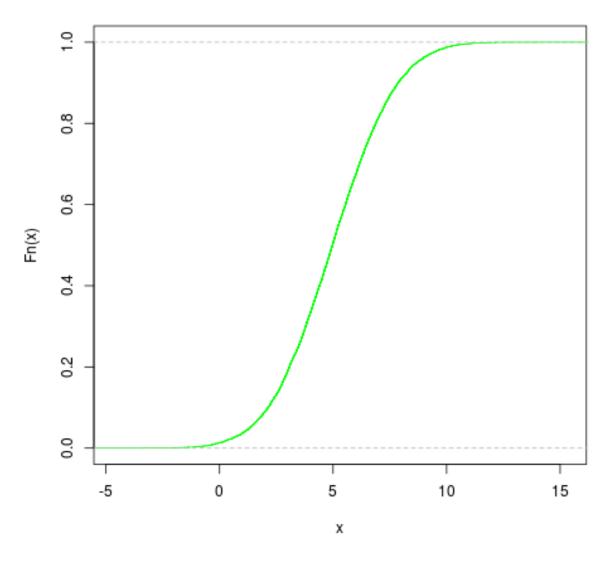
```
1  square<-function(u1,u2){
2    return (u1**2+u2**2)
3  }
4  intermediate<-function(x){
5    return (sqrt(-2*log(x)/x))
6  }
7  count<-10000
8  standard_deviation<-sqrt(5)
9  mean<-5
10  samples1<-c()</pre>
```

```
11 samples 2 < -c ()
12 i <-1
13 j < -0
14 while (1) {
15
      u1<-runif(1)
16
      u2 < -runif(1)
      u1 < -2 * u1 - 1
17
18
      u2 < -2 * u2 - 1
19
      j < -j + 1
20
      x < -square(u1, u2)
21
      if(x>1){
22
          next
23
      }
24
      y<-intermediate(x)
25
      samples1[i]<-u1*y*standard_deviation+mean
      samples2[i]<-u2*y*standard_deviation+mean
26
27
      i < -i + 1
      count<-count-2
28
      if (count==0){
29
          break
30
31
32 }
33 samples <- c (samples1, samples2)
34 samples <- sort (samples)
35 cat ("Mean is: ", mean (samples), "\n")
36 cat ("Acceptance Probability: ",(i-1)/j,"\n")
37 empirical_samples<-ecdf(samples)
38 png ("question 2 -mean = 5.png")
39 plot (empirical_samples, col="green")
40 par (new=TRUE)
41 #png ("Theoretical (red)—mean=5.png")
42 theoretical_samples <-pnorm(samples)
43 plot (theoretical_samples, col="red")
```

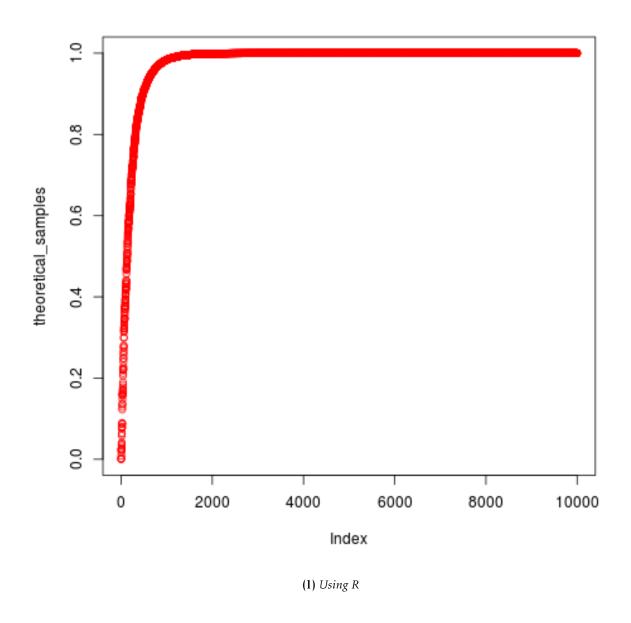
Observation: Graph:



(j) Using R



(k) Using R



Q 3 Keep a track of the computational time required for both the methods. Which method is faster?

Code for Box-Muller method: R

```
1 radius<-function(u) {
2    return (-2*log(u))
3 }
4 arg<-function(u) {
5    return (2*pi*u)
6 }</pre>
```

```
7 count<-10000
   stime<-Sys.time();</pre>
   samples1 < -c()
10 | samples 2 < -c ()
11 for (i in 1:count/2) {
12
      u1<-runif(1)
13
      u2 < -runif(1)
      samples1[i] \leftarrow sqrt(radius(u1)) * cos(arg(u2))
14
      samples2[i]<-sqrt(radius(u1))*sin(arg(u2))
15
16 }
17
18
19 samples <-c (samples1, samples2)
20 etime <-Sys. time();
21 cat ("Computation Time of Box-Muller Method: ",etime-stime,"\n");
```

Code for Marsaglia-Bray method: R

```
1 square <- function (u1, u2) {
       return (u1**2+u2**2)
 3 }
 4 intermediate <- function (x) {
 5
       return (\operatorname{sqrt}(-2*\log(x)/x))
 6 }
 7 count<-10000
 8 stime<-Sys.time();</pre>
 9 samples 1 < -c ()
10 | samples 2 < -c ()
11 i<-1
12 while (1) {
13
       u1 < -runif(1)
       u2<-runif(1)
14
       u1 < -2 * u1 - 1
15
16
       u2 < -2 * u2 - 1
17
       x < -square(u1, u2)
18
       if(x>1){
19
           next
20
       y<-intermediate(x)
21
22
       samples1[i]<-u1*y
23
       samples2[i] < -u2*y
       i < -i + 1
24
```

```
count<-count-2
if (count==0){
    break

samples<-c(samples1, samples2)

etime<-Sys.time();

cat("Computation Time of Marsaglia-Bray Method: ",etime-stime,"\n");</pre>
```

Output:

Computation Time of Box-Muller Method: 0.1807923

Computation Time of Marsaglia-Bray Method: 0.1370828

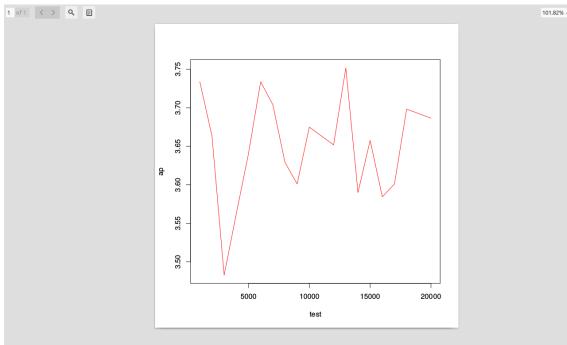
Q 4 For the Marsaglia-Bray method keep track of the proportional of values rejected. How does it compare with 1 pi/4 ?? ?

Code for Marsaglia-Bray method: R

```
1 square <- function (u1, u2) {
       return (u1**2+u2**2)
 3 }
 4 intermediate <- function (x) {
 5
       return (\operatorname{sqrt}(-2*\log(x)/x))
 6
   }
7 count<-1000
8 \operatorname{ap} < -c()
 9 for (k in 1:20) {
|10| samples 1 < -c()
11 samples 2 <- c()
12 i <-1
13 j <−0
14 while (1) {
       u1 < -runif(1)
15
16
       u2<-runif(1)
17
       u1 < -2 * u1 - 1
18
       u2 < -2 * u2 - 1
19
       j < -j + 1
20
       x < -square(u1, u2)
       if(x>1){
21
22
           next
```

```
23
24
      y<-intermediate(x)
25
      samples1[i] < -u1*y
      samples2[i]<-u2*y
26
27
      i < -i+1
      count < -count - 2
28
29
      if (count==0){
          break
30
31
32 }
33 count < -count + 1000
34 ap[k]<-(((i-1)/j)/(1-pi/4))
35 }
36 samples<-c (samples1, samples2)
37 k<-1
38 | test < -c()
39 constant <- 1000
40 for (k in 1:20) {
      test[k]<-constant*k
41
42 }
43 cat ("Acceptance Probability: ",(i-1)/j,"\n")
44 | plot (test, ap, type="l", col="red")
```

Observation: Graph:



by 1-pi divide 2.png

(m) Using R