HW1

Haoluan Chen

9/21/2021

Name: Haoluan Chen
Student number: 1003994261
Department: Staitstial Science Program: Master of Statistics
Year: 1
Email: haoluan.chen@mail.utoronto.ca

1 Generate pseudorandomm Uniform [0,1]

```
seed <- as.numeric(Sys.time())
m = 2^31 # I chosed m = 2^31 - 1 because it is a very large positive prime number
a = 68626861  # It is a large value and a multiple of 4

b = 1003994261 #It is a odd number, so my LCG is full period/
latestval<- floor(print(as.numeric(Sys.time())*1000, digits=15))
```

```
## [1] 1633718630232.87
```

```
remainder = function(n,m) {
    return( n - m * floor(n/m) )
}

nextrand = function() {
    latestval <<- remainder(a*latestval+b, m)   # (global assignment)
    return(latestval / m)
}


x = rep(0,1000)
for (i in 1:1000)
  x[i] = nextrand()
print("First and Second Moments of a sample of size 1,000:")
```

```
## [1] "First and Second Moments of a sample of size 1,000:"
```

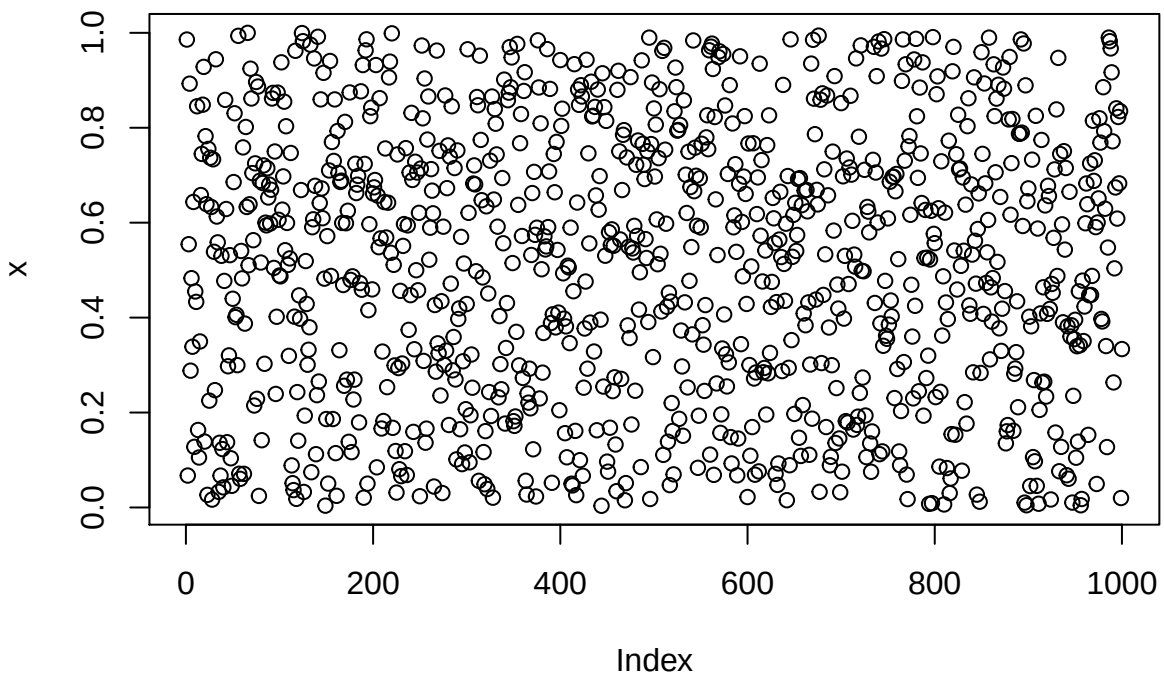
```
print( mean(x) )
```

```
## [1] 0.5104446
```

```
print( mean(x^2) )
```

```
## [1] 0.3406763
```

```
plot(x)
```




```
y = rep(0,1000)
for (i in 1:1000)
  y[i] = nextrand()
print("First and Second Moments of a sample of size 1,000:")
```

```
## [1] "First and Second Moments of a sample of size 1,000:"
```

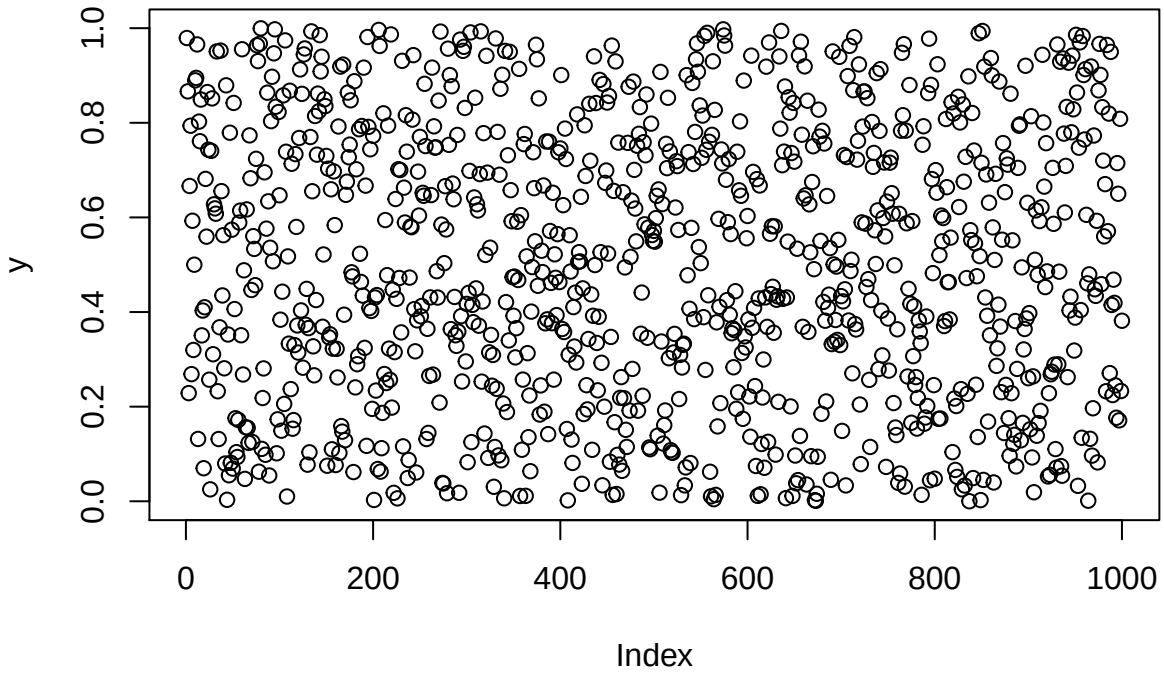
```
print( mean(y) )
```

```
## [1] 0.49787
```

```
print( mean(y^2) )
```

```
## [1] 0.3309734
```

```
plot(y)
```




```
cor(x,y)
```

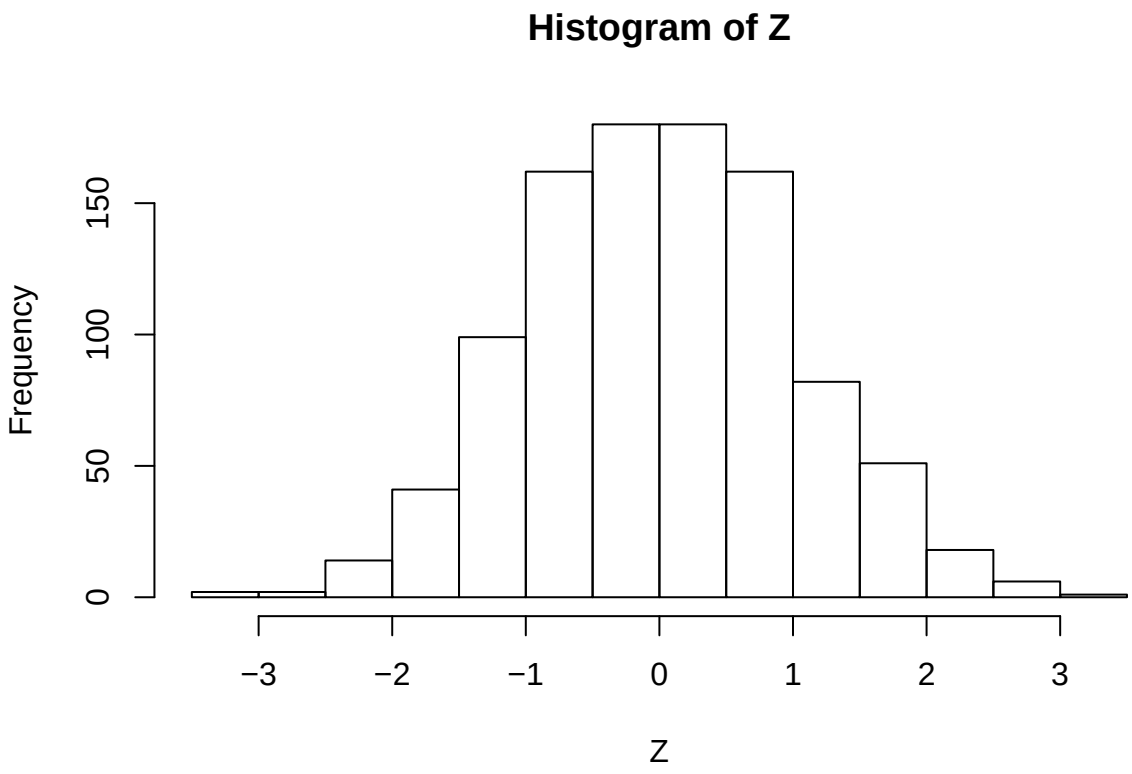
```
## [1] 0.04137135
```

I think my generator seems to be pretty good. First, I ran it twice and found the mean to be close to 0.5 and variance close to 0.33. This is similar to results in Rrng file. Second, two plot shows my points are uniformly distributed between 0 and 1. Third, the correlation between the two set of data that I generated are close to 0.

2)

```
u1= rep(0,1000)
for (i in 1:1000)
  u1[i] = nextrand()
# generate normal[0,1]
Z = sqrt(2*log(1/x))* cos(2*pi*u1)
# generate Exp(3)
Y = -log(y)/3

# checking if my normal(0,1) is good
hist(Z)
```




```
mean(Z)
```

```
## [1] 0.01643404
```

```
mean(Z^2)
```

```
## [1] 0.9958038
```

```
# checking if my Exp(3) is good
mean(Y) # should be 1/3
```

```
## [1] 0.3412465
```

```
mean(Y^2) # should be 1/3^2
```

```
## [1] 0.2437101
```

Normal looks good, but for exp(3) slightly larger variance compared to theoretical variance.

First attempt:

```
mean(abs(Y^2 * Z^5 * sin(Y^3*Z^2)))
```

```
## [1] 0.7144915
```

```
se = sd(abs(Y^2 * Z^5 * sin(Y^3*Z^2)))/ 1000
se
```

```
## [1] 0.00441032
```

```
result = rep(0, 20)
```

```
q2 <- function(){
x = rep(0,1000)
y = rep(0,1000)
u1= rep(0,1000)
for (i in 1:1000){
  x[i] = nextrand()
  y[i] = nextrand()
  u1[i] = nextrand()
  }

Z = sqrt(2*log(1/x))* cos(2*pi*u1)
Y = -log(y)/3

est <- mean(abs(Y^2 * Z^5 * sin(Y^3*Z^2)))
est
se = sd(abs(Y^2 * Z^5 * sin(Y^3*Z^2)))/ 1000
print(paste("mean =" ,est ,"and se =", se))
}


for (i in 1:20){
  each <- q2()
  result[i] = as.numeric(substr(each, 8, 20))
}
```

```
## [1] "mean = 0.841379408214832 and se = 0.00612335155552169"
## [1] "mean = 0.813573595860695 and se = 0.00667746672642425"
## [1] "mean = 0.723666274920127 and se = 0.00513978922857547"
## [1] "mean = 0.387860172345146 and se = 0.00285826014370122"
## [1] "mean = 0.470745428164911 and se = 0.00464788304383669"
## [1] "mean = 0.536217039867703 and se = 0.00518739230725002"
## [1] "mean = 0.562403656390422 and se = 0.00567798687167452"
## [1] "mean = 0.637515874012331 and se = 0.00491227353022193"
## [1] "mean = 1.11042741897974 and se = 0.0114615235702178"
## [1] "mean = 0.885360527046843 and se = 0.00898445311227278"
## [1] "mean = 0.651718850230217 and se = 0.00647603019341789"
## [1] "mean = 0.562132644523543 and se = 0.00384599579738137"
## [1] "mean = 1.11038367951472 and se = 0.0114514607939083"
## [1] "mean = 0.864069368978021 and se = 0.0089741853803954"
## [1] "mean = 0.667556195391633 and se = 0.00648592647412979"
## [1] "mean = 0.564542848316474 and se = 0.00386122332167508"
## [1] "mean = 1.20445737999525 and se = 0.0119108169793083"
## [1] "mean = 0.751885243926115 and se = 0.00833660258459282"
## [1] "mean = 0.702511509976582 and se = 0.00652648411483594"
## [1] "mean = 0.544858688152598 and se = 0.00381331527808679"
```

```
print(paste("final estiamte:", mean(result)))
```

```
## [1] "final estiamte: 0.729663290236"
```

I don't think my estimate is very accurate because my estimate seems to vary by a lot. For example from some of my run, I saw the lowest estimate is 0.45 and my highest estimate is 0.85, which is almost 2 times the lowest estimate. Secondly, the variance for the exp(3) random variable are not very close to the theoretical variance.

3)

My student ID is 1003994261, so A = 4 , B = 2, C = 6, D = 1.

```
set.seed(1)
A = 4
B = 2
C = 6
D = 1

g <- function(x1, x2, x3, x4, x5){
  x1^{A+6}*2^{x2+3} * (1 + cos(x1 + 2*x2 + 3*x3 + 4*x4 + (B + 3)*x5))*
    exp((C-12)*x4^2)*exp(-(D+2)*(x4-3*x5)^2)
}

h <- function(x1, x2, x3, x4, x5){
  (x1 + x2^2) / (2 + x3*x4 + x5)
}

q3 <- function(){
M = 10^6

x1list = runif(M)
x2list = runif(M)
x3list = runif(M)
x4list = runif(M)
x5list = runif(M)

numlist = g(x1list, x2list, x3list, x4list, x5list) *
  h(x1list, x2list, x3list, x4list, x5list)

denomlist = g(x1list, x2list, x3list, x4list, x5list)


se = M^{-1/2}*sd(h(x1list, x2list, x3list, x4list, x5list))
se

print(paste("My estimate is ", mean(numlist)/mean(denomlist), "with se =", se))
}


q3()
```

```
## [1] "My estimate is  0.591228487460249 with se = 0.00016043617770047"
```

```
q3()
```

```
## [1] "My estimate is  0.589846510031752 with se = 0.000160300960119795"
```

```
q3()
```

```
## [1] "My estimate is  0.590007672627745 with se = 0.000160291563466063"
```

```
q3()
```

```
## [1] "My estimate is  0.590848821375935 with se = 0.000160214187620242"
```

```
q3()
```

```
## [1] "My estimate is  0.589140457666465 with se = 0.000160460461364194"
```

I chose all of my x1, x2, x3, x4, x5 to be uniform(0,1) distribution, because it is easy to sample from and it reduces the computational complex of my calculation.

```
# final esitmate (taking means of one of my runs)
(0.590788435863751 + 0.590569032282988 + 0.590206922048098
 + 0.591317519358796 + 0.590511405015528)/5
```

```
## [1] 0.5906787
```

I think my program does work well, because my estimates are very consistent (around 0.59). My final estimate is 0.5906787. I think my estimate is pretty accurate because it has consistent and low estimated variance. Also it is close to my estimate in question 4.

4

I want to find a 5-dimensional function that is greater than g(x1, x2, x3, x4, x5) for all xi in (0,1)

my function g is:

$$x_1^{10} 2^{x_2+3} (1 + cos[x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5]) e^{-6x_4^2} e^{-3(x_4 - 3x_5)^2}$$

$(1 + cos[x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5])$ The maximum of this part of the function is equal to 2.

2^{x_2+3} Since x_2 is between 0 to 1, so the maximum of this part of the function is 2^4

$e^{-6x_4^2} < 1$ (since $0 < x_4, x_5 < 1$)

$e^{-3(x_4-3x_5)^2} < 1$ (since $0 < x_4, x_5 < 1$)

Therefore Let $K = 2^4 * 2 $

Let

$$f(x1) = 1_{0<x_1<1}$$

$$f(x_2) = 1_{0<x_2<1}$$
$$f(x_3) = 1_{0<x_3<1}$$

$$f(x4) = 1_{0<x_4<1}$$
$$f(x5) = 1_{0<x_5<1}$$

```
k = 2^4 * 2
f <- function(x1,x2, x3, x4, x5) {1*x1*x2*x3*x4*x5}
M = 100000


q4<- function(){
hlist = rep(NA, M)
numsamples = 0
x1list= rep(NA, M)
x2list= rep(NA, M)
```

```
x3list= rep(NA, M)
x4list= rep(NA, M)
x5list= rep(NA, M)


for (i in 1:M) {
    x1 = runif(1)
    x2 = runif(1)
    x3 = runif(1)
    x4 = runif(1)
    x5 = runif(1)
    X = f(x1,x2, x3, x4, x5)  # sample from f
    U = runif(1)  # for accept/reject
    alpha = g(x1, x2, x3, x4, x5) / (k * 1)  # for accept/reject
    if (U < alpha) {
        x1list[i] = x1  # keep X value if accepted
          x2list[i] = x2  # keep X value if accepted
          x3list[i] = x3  # keep X value if accepted
          x4list[i] = x4  # keep X value if accepted
          x5list[i] = x5  # keep X value if accepted
          hlist[i] = h(x1,x2,x3,x4,x5) # keep h(X) value if accepted
        numsamples = numsamples + 1
    }

}
print(paste(numsamples, mean(hlist, na.rm=TRUE)))
}

for(i in 1:10){
  q4()
}
```

```
## [1] "331 0.58712528596649"
## [1] "331 0.585261907096158"
## [1] "342 0.582592897558771"
## [1] "289 0.589129111869184"
## [1] "342 0.590889931925265"
## [1] "351 0.582426814805924"
## [1] "344 0.583003213887997"
## [1] "324 0.588724870451217"
## [1] "367 0.597622678085176"
## [1] "319 0.59032978185728"
```

```
#final estimate (taking means of one of my runs)
(0.58712528596649 + 0.585261907096158 + 0.582592897558771 + 0.589129111869184 + 0.590889931925265
 + 0.582426814805924 + 0.583003213887997 + 0.588724870451217 + 0.597622678085176 + 0.59032978185728)/10
```

```
## [1] 0.5877106
```

I think my algorithm does work well, because the number of accepted samples is high (around 300) and my estimate varies only by a bit. Also my estimate is very close to my estimate from q3 around 0.59