Week 2 Assignment

John Navarro

Part 1. Generate uniformly distributed random numbers

1.1 Use runif()

```
##set specific seed, use runif() to create a sample of 0s and 1s with 1000
##datapoints
set.seed(15)
Sample<-runif(1000,0,1)
```

1.2 Simulate Uniform Random Sample on [0,1] Using Random.org

```
##install random package, set nFlips to 1000, use randomNumbers() function to
##create data set dataFromRandom, which is a series of 0s and 1s. print the
##first 6 items in the dataset
library(random)
nFlips<-1000
dataFromRandom<-randomNumbers(n=nFlips, min=0, max=1, col=1, base=2, check=TRUE)
head(dataFromRandom)
```

```
##      V1
## [1,]  1
## [2,]  1
## [3,]  1
## [4,]  0
## [5,]  1
## [6,]  1
```

Learn how to turn your sequence of {0,1} into uniform random numbers on [0,1]

```
##install compositions package
suppressMessages(library(compositions))
##create a function called bin2dec that takes the input Bin.Seq, removes the
##"," and uses function unbinary() to turn the string into a number
bin2dec<-function(Bin.Seq){
  unbinary(paste(Bin.Seq,collapse=""))
}
bin2dec(c(1,1,1,1,1,0))
```

```
## [1] 62
```

Turn the sequence of zeros and ones dataFromRandom of length 1000 into a matrix with 10 columns and 100 rows

```
##Create a matrix of 10 col and 100 rows
Binary.matrix<-matrix(dataFromRandom,ncol=10)
head(Binary.matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    0    0    1    1    0    1    0    0     0
## [2,]    1    1    1    0    1    0    1    0    1     1
## [3,]    1    1    1    1    1    0    1    1    1     0
## [4,]    0    0    0    1    1    0    0    0    1     0
## [5,]    1    0    1    0    1    1    0    0    0     0
## [6,]    1    0    1    0    1    0    0    0    1     1
```

Transform each row of the matrix into decimal format using bin2dec() and divide the numbers by 2^10 to make real numbers in [0,1]

```
##Use bin2dec() on each row and divide the values by 2^100 to get 100 decimal
##values
Decimal.Sample<-apply(Binary.matrix,1,bin2dec)/2^10
Decimal.Sample
```

```
##   [1] 0.60156250 0.91699219 0.98242188 0.09570312 0.67187500 0.65917969
##   [7] 0.67187500 0.20605469 0.27636719 0.27441406 0.18847656 0.46875000
##  [13] 0.40917969 0.03710938 0.06738281 0.93261719 0.31152344 0.31738281
##  [19] 0.14941406 0.90136719 0.78222656 0.79785156 0.79687500 0.15722656
##  [25] 0.21289062 0.20117188 0.79296875 0.20800781 0.21191406 0.33593750
##  [31] 0.33886719 0.25390625 0.52539062 0.07910156 0.99707031 0.54687500
##  [37] 0.24414062 0.94140625 0.10839844 0.71972656 0.59375000 0.89257812
##  [43] 0.80273438 0.91406250 0.75390625 0.76367188 0.79199219 0.05566406
##  [49] 0.62695312 0.58691406 0.66796875 0.44335938 0.66503906 0.09667969
##  [55] 0.45605469 0.32031250 0.72656250 0.73144531 0.90820312 0.51171875
##  [61] 0.66992188 0.38867188 0.71679688 0.60156250 0.05078125 0.26464844
##  [67] 0.16601562 0.90234375 0.97851562 0.00781250 0.37597656 0.81250000
##  [73] 0.73730469 0.27148438 0.29394531 0.63769531 0.68750000 0.43066406
##  [79] 0.24414062 0.25195312 0.65917969 0.08691406 0.61718750 0.38281250
##  [85] 0.55957031 0.69042969 0.39746094 0.42871094 0.63769531 0.35546875
##  [91] 0.91308594 0.33300781 0.17089844 0.19531250 0.93261719 0.73730469
##  [97] 0.41796875 0.75683594 0.34472656 0.52148438
```

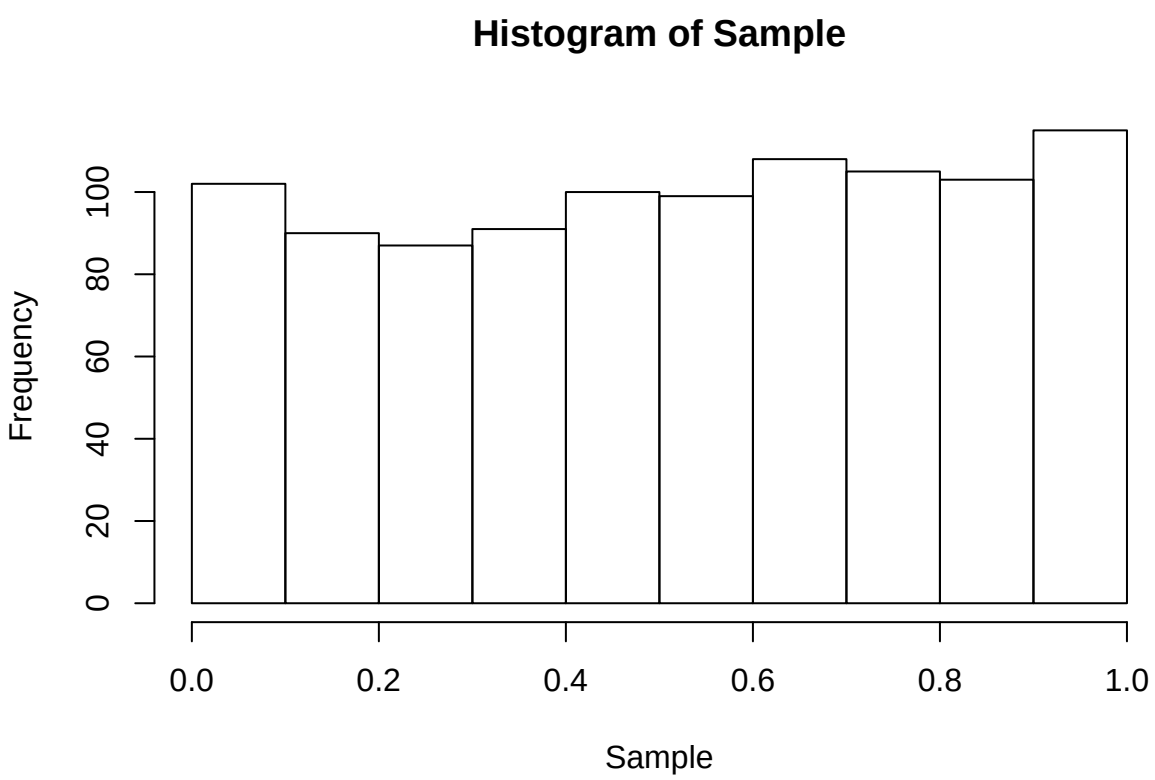
Part 2. Test random number generator

2.1 Test uniformity of distribution of both random number generators

2.1.1. Sample obtained by runif()

Analyze what was simulated by first looking at the histogram.

```
##plot a histogram of Sample, the data from runif()
Sample.histogram<-hist(Sample)
```




```
Sample.histogram
```

```
## $breaks
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
##
## $counts
##  [1] 102  90  87  91 100  99 108 105 103 115
##
## $density
##  [1] 1.02 0.90 0.87 0.91 1.00 0.99 1.08 1.05 1.03 1.15
##
## $mids
##  [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95
##
## $xname
## [1] "Sample"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

What does the histogram tell you about the distribution? Is it consistent with the goal of simulation? This histogram is a uniform distribution. Yes, it is consistent with the goal of the simulation. Which was to create a random uniform set of 1000 points all between 0 and 1.

Estimate mean and standard deviation of Sample.histogram$density.

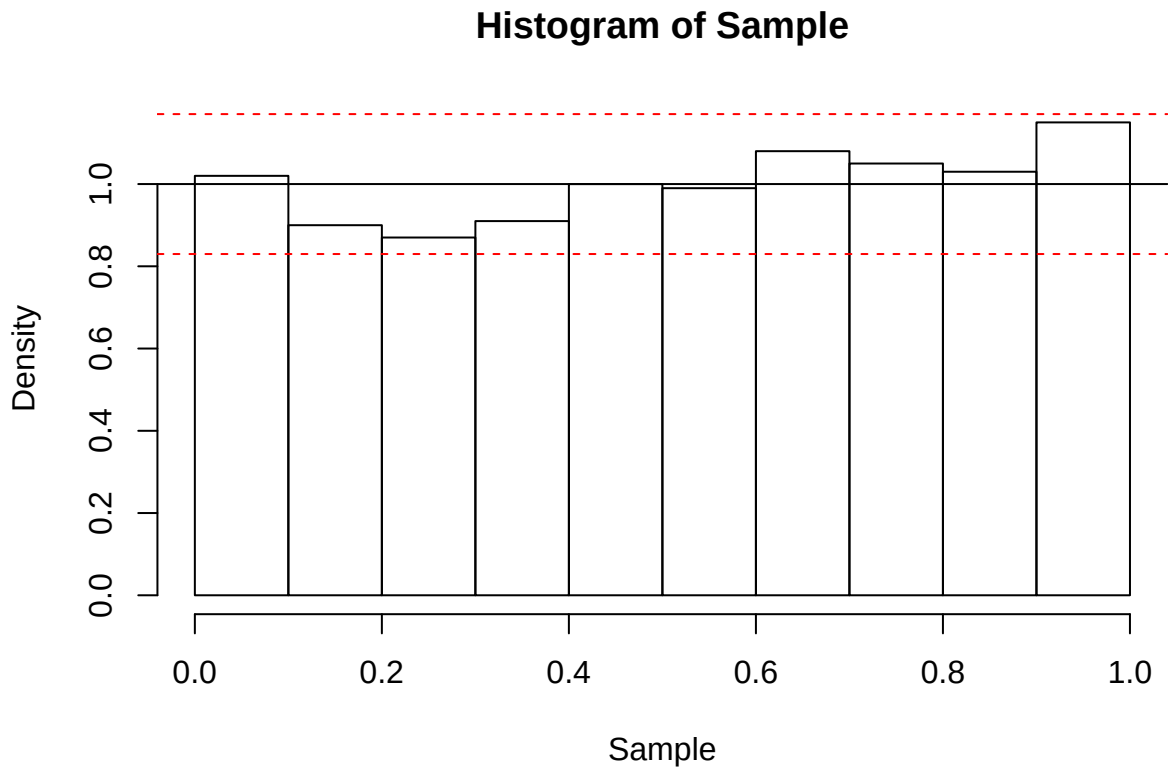
```
##find the mean and sd of the sample$density
(Sample.histogram.mean<-mean(Sample.histogram$density))
```

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

```
(Sample.histogram.sd<-sd(Sample.histogram$density))
```

```
## [1] 0.08679478
```

```
##plot the histogram showing density not frequency
plot(Sample.histogram,freq=FALSE)
##draw the lines for the mean and the 95% Conf Intv
abline(h=Sample.histogram.mean)
abline(h=Sample.histogram.mean+1.96*Sample.histogram.sd,col="red",lty=2)
abline(h=Sample.histogram.mean-1.96*Sample.histogram.sd,col="red",lty=2)
```



What does the graph tell you about the observed distribution?

It looks like a uniform distribution with mean of 1 and a standard deviation of 0.0868

```
##Find the mean and variance of the Sample
(Sample.mean<-mean(Sample))
```

```
## [1] 0.5161848
```

```
(Sample.variance<-var(Sample))
```

```
## [1] 0.08413663
```

What do you conclude about the estimated distribution from the moments?

The mean of Sample is 0.516 and the variance is 0.0841 so most of the data is centered around 0.5 with a narrow variance.

Check the summary of the simulated sample.

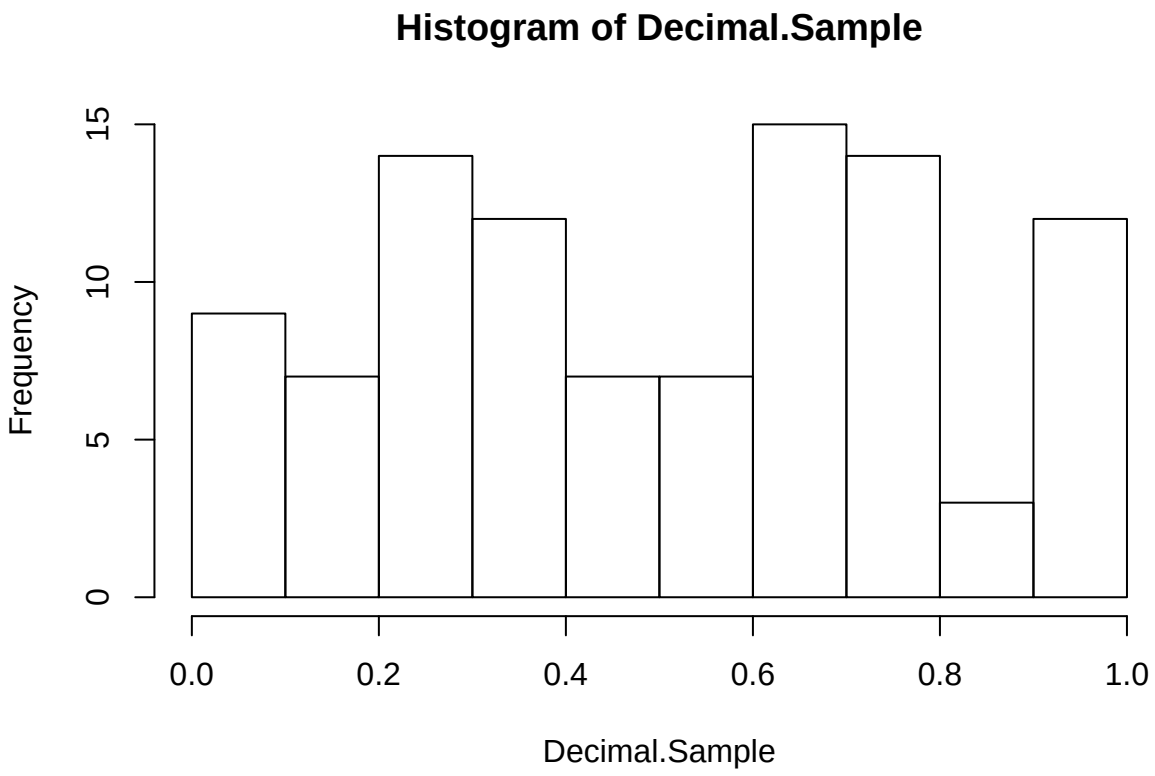
```
#run the summary of the sample
summary(Sample)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## 0.0000127 0.2678000 0.5210000 0.5162000 0.7620000 0.9980000
```

What do you think is the best way of estimating uniform distribution over unknown interval? The best way to estimate uniform distribution is by analyzing the moments.

2.1.2. Repeat the same steps to test uniformity of the sample created from Random.org data.

```
##plot a histogram using the decimal values from bin2dec() function
D.Sample.histogram<-hist(Decimal.Sample)
```



```
D.Sample.histogram
```

```
## $breaks
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
##
## $counts
##  [1]  9  7 14 12  7  7 15 14  3 12
##
## $density
##  [1] 0.9 0.7 1.4 1.2 0.7 0.7 1.5 1.4 0.3 1.2
##
## $mids
##  [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95
##
## $xname
## [1] "Decimal.Sample"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

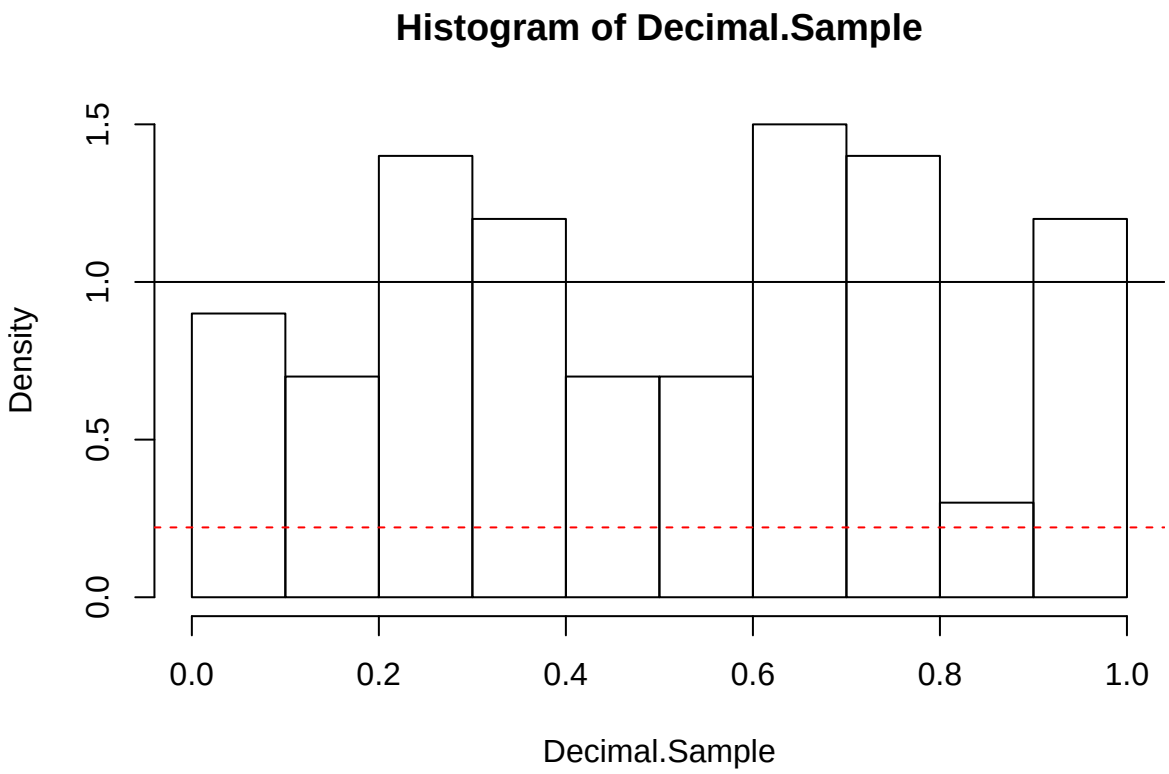
```
#Calculate the mean and sd of histogram$density
(D.Sample.histogram.mean<-mean(D.Sample.histogram$density))
```

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

```
(D.Sample.histogram.sd<-sd(D.Sample.histogram$density))
```

```
## [1] 0.3972125
```

```
##plot the histogram with density, not frequency
plot(D.Sample.histogram,freq=FALSE)
##plot the mean and Conf Intv
abline(h=D.Sample.histogram.mean)
abline(h=D.Sample.histogram.mean+1.96*D.Sample.histogram.sd,col="red",lty=2)
abline(h=D.Sample.histogram.mean-1.96*D.Sample.histogram.sd,col="red",lty=2)
```




```
##Calculate the mean, variance and summary of the sample
(D.Sample.mean<-mean(Decimal.Sample))
```

```
## [1] 0.5033008
```

```
(D.Sample.variance<-var(Decimal.Sample))
```

```
## [1] 0.07781878
```

```
summary(Decimal.Sample)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.007812 0.262000 0.516600 0.503300 0.732900 0.997100
```

What does the histogram tell you about the distribution? Is it consistent with the goal of simulation? That the distribution is not uniform, It looks as if the mode is around 0.2. It does not seem consistent with the goal of the simulation. *What does the graph tell you about the observed distribution?* It says that the density has a mean of 1, but a wide confidence interval, due to the large standard deviation. *What do you conclude about the estimated distribution from the moments?* This distribution has a similar mean, but a larger variance. It still gives a uniform distribution. *What do you think is the best way of estimating uniform distribution over unknown interval?*

The best way to estimate uniform distribution is by analyzing the moments.

2.2. Test independence of the sequence of zeros and ones

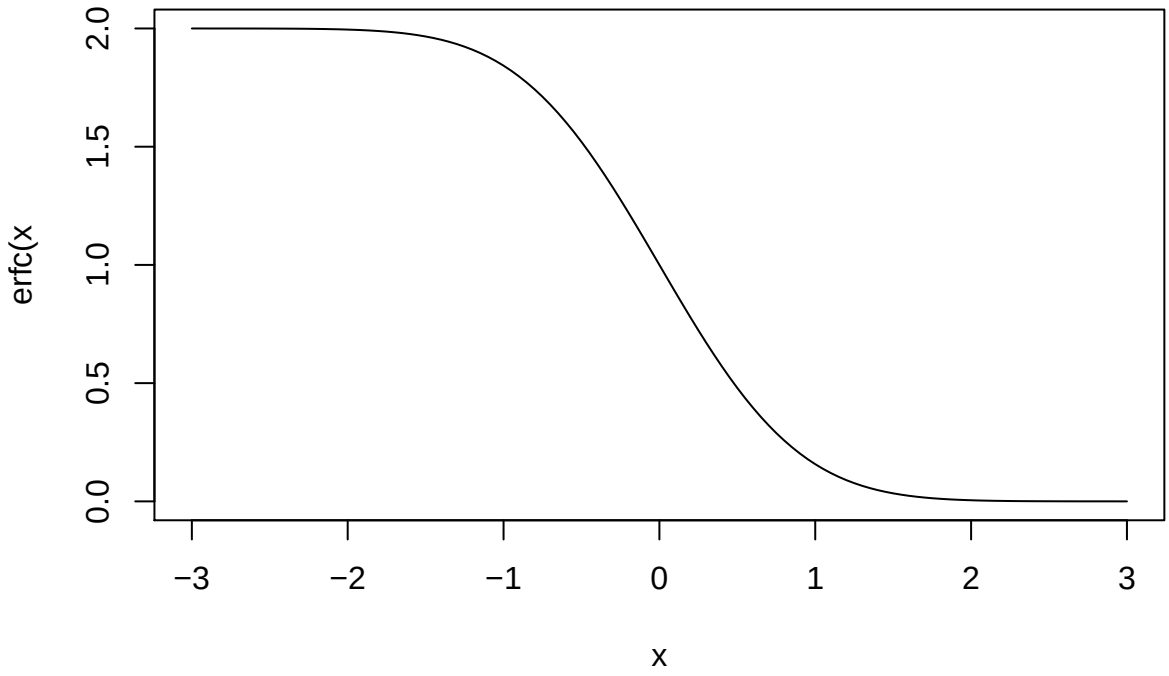
2.2.1. Turning point test

```
##install randtests package, run turning.point.test() on decimal.sample
library(randtests)
turning.point.test(Decimal.Sample)
```

```
##
##  Turning Point Test
##
## data:  Decimal.Sample
## statistic = 2.0744, n = 100, p-value = 0.03805
## alternative hypothesis: non randomness
```

2.3. Test frequency by Monobit test

```
##transorm data of {0,1} to {-1,1}
dataFromRandom.plusminus1<-(dataFromRandom-.5)*2
##create 2 complimentary functions erf and erfc, that
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
##plot the erfc
plot(seq(from=-3,to=3,by=.05),erfc(seq(from=-3,to=3,by=.05)),type="l",xlab="x",ylab="erfc(x")
```



To test the sequence RiRi check the value erfc(S).

If the P-value or erfc(S) is less or equal than 0.01 the sequence fails the test.

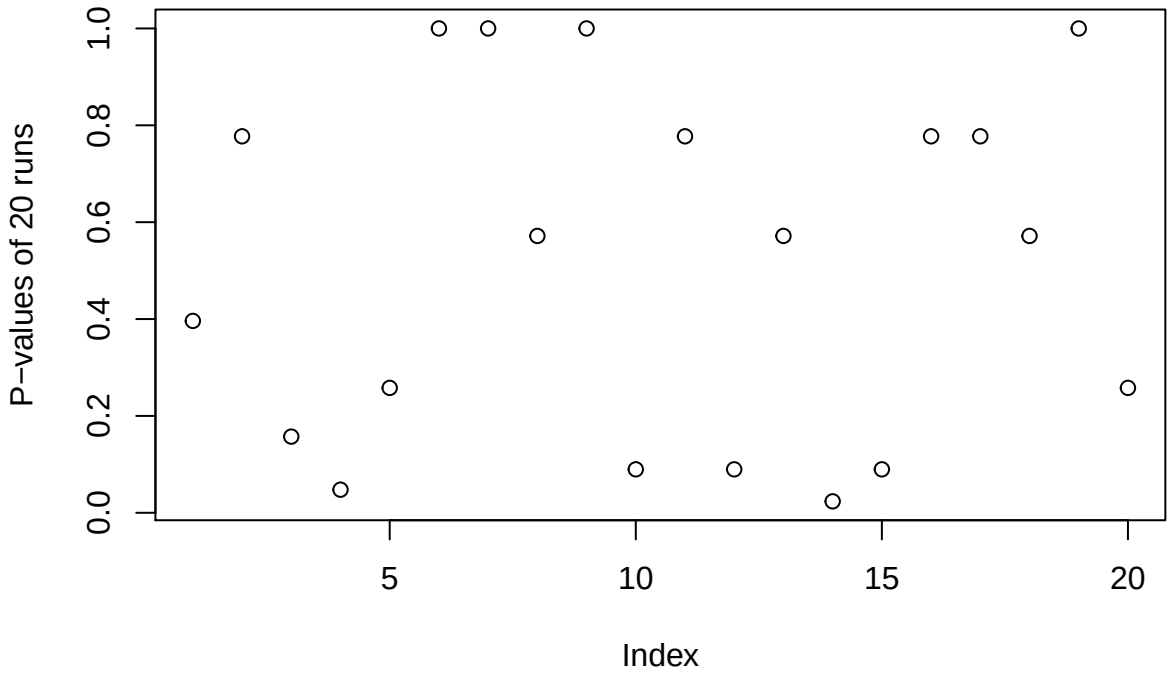
```
##Sum the rows, and take absolute value and divide by 100 to get decimal
##Use erfc, to get the pvalues of the sub-sequences
##see which ones are less than 0.01 and sum the true ones
erfc(abs(sum(dataFromRandom.plusminus1)/sqrt(2*nFlips)))
```

```
## [1] 0.2294931
```

The test shows that our sequence passes.

Now check each of the sub-sequences created earlier:

```
plot(erfc(abs(apply(matrix(dataFromRandom.plusminus1,ncol=50),1,sum))/sqrt(2*50)),ylab="P-values of 20 r
```



How many runs out of 20 fail the test?

```
sum(erfc(abs(apply(matrix(dataFromRandom.plusminus1,ncol=50),1,sum))/sqrt(2*50))<=.01)
```

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

None of them fail, there are none less than 0.01

Part 3. Invent a random number generator

Think about possible sources of true or pseudo-random sequences of {0,1} and choose one or two of them. Conduct the tests described in the previous section.

3.1 Description of your random number generator

My sequence is taken from a MLB pitcher (Jake Arrieta) career statistics for walks and strikeouts per game. For each game I tabulated a ratio of walks divided by strikeouts, and removed the NA values(for when he had zero strikeouts) as well as the values greater than 1 for when he had more walks than strikeouts. We are left with 128 datapoints between 0 and 1.

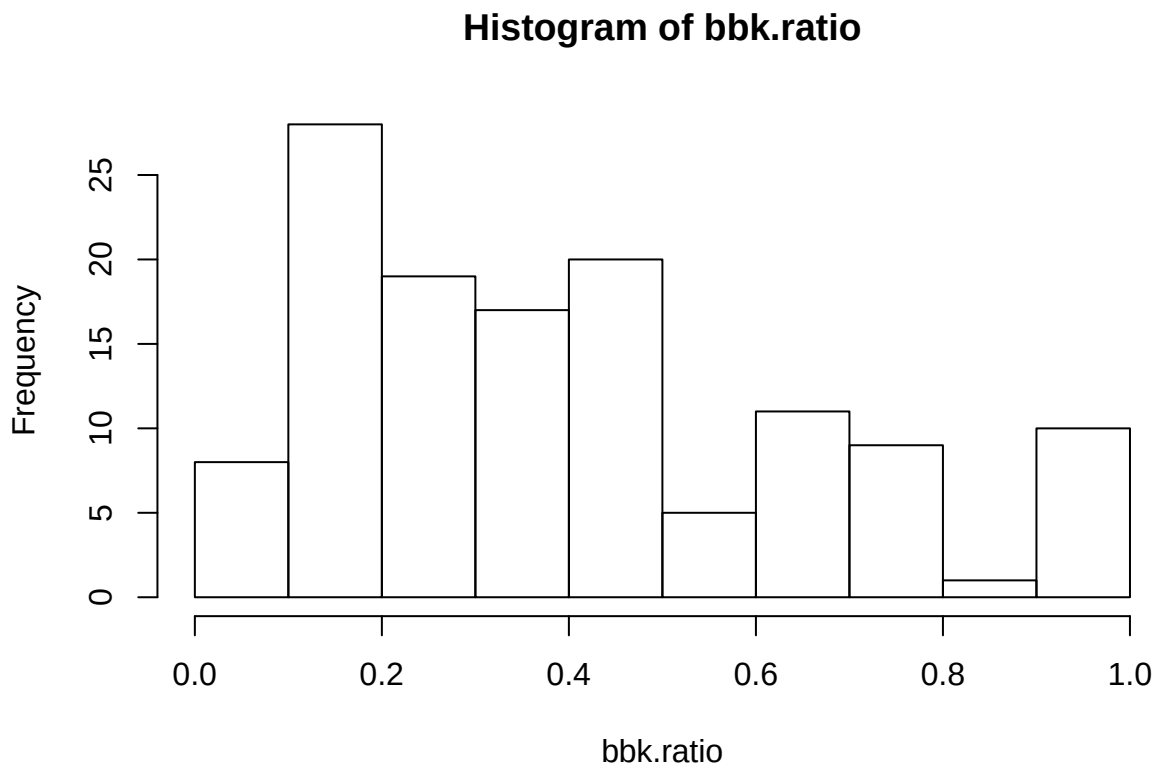
3.2 Generated sequence

```
##read in the data
setwd("C:/Users/JohntheGreat/Documents/MSCA/StatisticalAnalysis/Week2/Assignments")
ratio <- read.csv(file = "ArrietaRatio.csv", header = F)
bbk.ratio <- ratio$V3
head(bbk.ratio)
```

```
## [1] 0.07692308 0.08333333 0.08333333 0.09090909 0.09090909 0.10000000
```

3.3 Uniformity test

```
## plot a histogram of bbk,ratio
bbk.ratio.histogram<-hist(bbk.ratio)
```




```
bbk.ratio.histogram
```

```
## $breaks
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
##
## $counts
##  [1]  8 28 19 17 20  5 11  9  1 10
##
## $density
##  [1] 0.625000 2.187500 1.484375 1.328125 1.562500 0.390625 0.859375
##  [8] 0.703125 0.078125 0.781250
##
## $mids
##  [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95
##
## $xname
## [1] "bbk.ratio"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

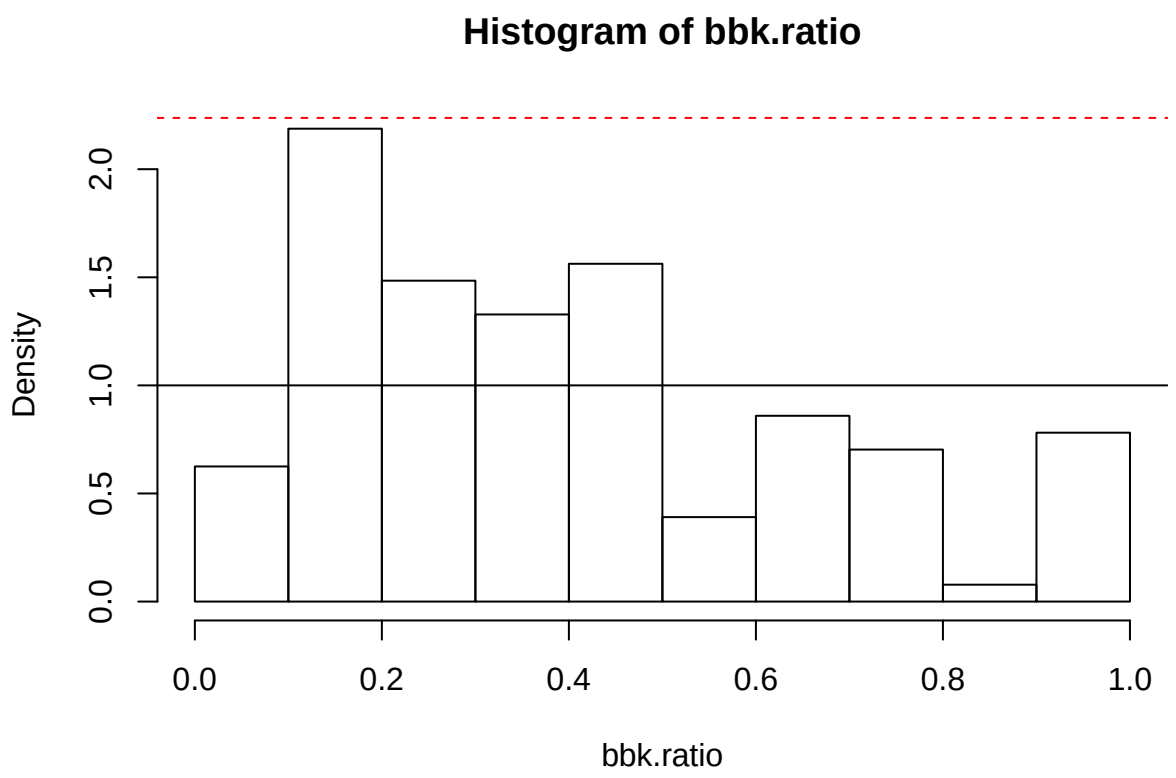
```
##calc the mean and sd of the density
(bbk.ratio.histogram.mean<-mean(bbk.ratio.histogram$density))
```

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

```
(bbk.ratio.histogram.sd<-sd(bbk.ratio.histogram$density))
```

```
## [1] 0.631262
```

```
##plot the histogram and the mean and CI
plot(bbk.ratio.histogram,freq=FALSE)
abline(h=bbk.ratio.histogram.mean)
abline(h=bbk.ratio.histogram.mean+1.96*bbk.ratio.histogram.sd,col="red",lty=2)
abline(h=bbk.ratio.histogram.mean-1.96*bbk.ratio.histogram.sd,col="red",lty=2)
```




```
##Calc the mean, variance and summary of bbk.ratio
(bbk.ratio.mean<-mean(bbk.ratio))
```

```
## [1] 0.4237362
```

```
(bbk.ratio.variance<-var(bbk.ratio))
```

```
## [1] 0.06918158
```

```r
summary(bbk.ratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07692 0.20000 0.37500 0.42370 0.60000 1.00000
```

What does the histogram tell you about the distribution? Is it consistent with the goal of simulation? The distribution is not uniform, it looks like the frequencies are high for low values, and low for the values between 0.5 and 1.

What does the graph tell you about the observed distribution? It tells me that the variance of the densities are large, especially none of the data falls withing the confidence interval.

What do you conclude about the estimated distribution from the moments? It tells me that that mean is at 0.42, and not close to 0.50. Most of the data is under 0.50

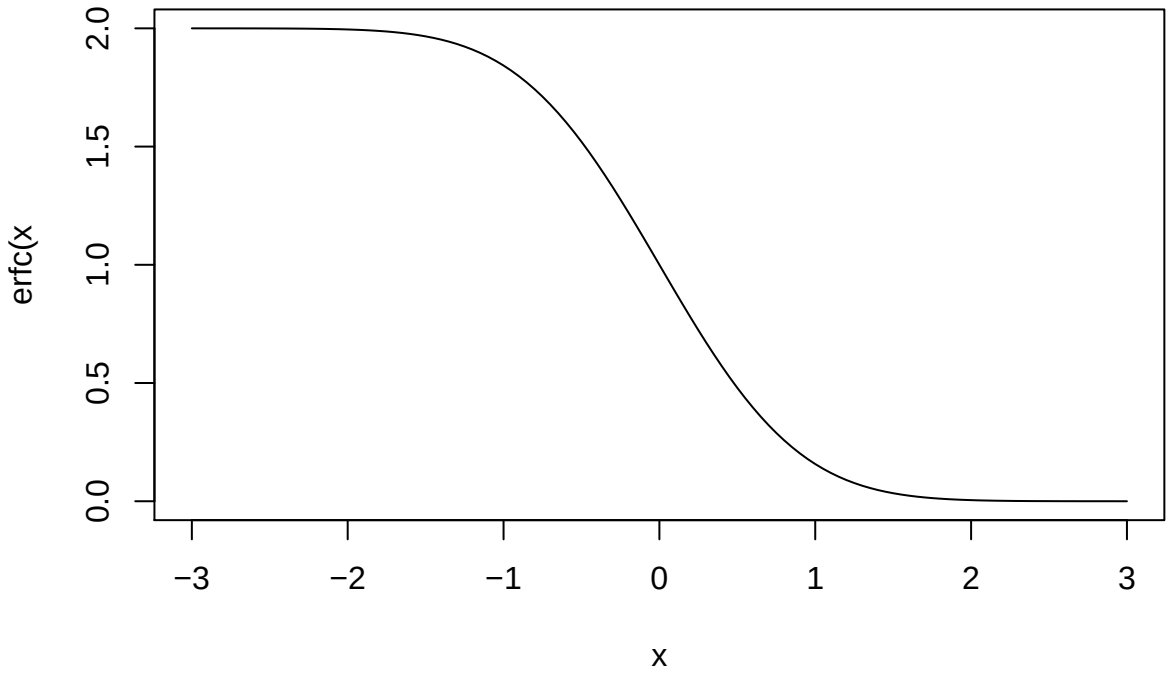
What do you think is the best way of estimating uniform distribution over unknown interval? The best way to estimate uniform distribution is by analyzing the moments.

3.4 frequency test

```r
library(compositions)
##use binary to convert bbk.ratio to binary values
bbk.bin <- binary(bbk.ratio * 100)
##paste all values to form a string of 0s and 1s
##split all the characters and turn them into numerics
bbk.sample <- as.numeric(unlist(strsplit(paste(bbk.bin, collapse = ""), split="")))
##turn the 0s and 1s to +/-1s
bbk.bin.plusminus1 <- (bbk.sample - .5)*2

erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)

plot(seq(from=-3,to=3,by=.05),erfc(seq(from=-3,to=3,by=.05)),type="l",xlab="x",ylab="erfc(x")
```

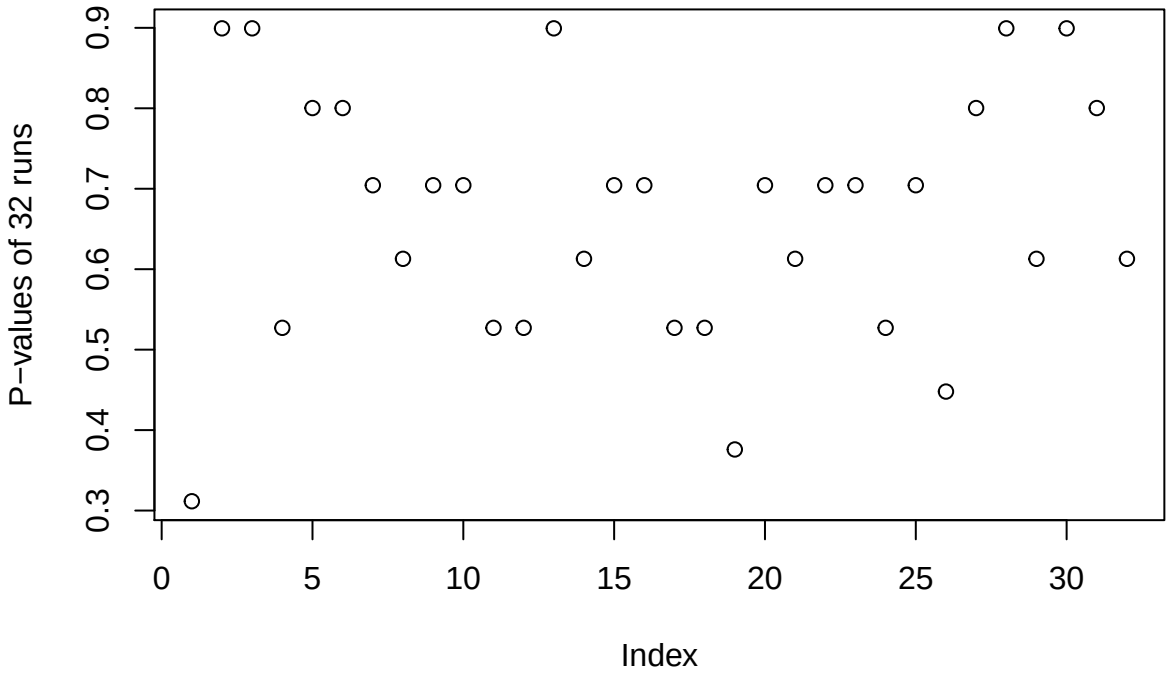



```
erfc(abs(sum(bbk.bin.plusminus1)/sqrt(2*nFlips)))
```

```
## [1] 2.214615e-12
```

Since the Pvalue is less than 0.01, by a signifcant amount, this sequence fails the test and is not random.

```
##I only have 896 data points, so I used a matrix with 28 columns and 32 rows.
plot(erfc(abs(apply(matrix(bbk.bin.plusminus1,ncol=28),1,sum))/sqrt(2*250)),ylab="P-values of 32 runs")
```




```
##Sum the rows, and take absolute value and divide by 100 to get decimal
##Use erfc, to get the pvalues of the sub-sequences
##see which ones are less than 0.01 and sum the true ones
sum(erfc(abs(apply(matrix(bbk.bin.plusminus1,ncol=28),1,sum))/sqrt(2*50))<=.01)
```

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

None of the sub sequences are less than 0.01

3.5 Turning point test

```
##install randtests package, run turning.point.test() on decimal.sample
library(randtests)
turning.point.test(bbk.ratio)
```

```
##
##  Turning Point Test
##
## data:  bbk.ratio
## statistic = -8.1875, n = 29, p-value = 2.668e-16
## alternative hypothesis: non randomness
```

Part 4. Monte Carlo Method

4.1. Scratch off quote of the day: function download

```
##load the file
datapath<-"C:/Users/JohntheGreat/Downloads/documents%2FMScA Statistical Analysis 31007%2FMScA 31007 Lec
#load(file=paste(datapath,'ScratchOffMonteCarlo.rda',sep='/'))

##load(file = datapath)
load("C:/Users/JohntheGreat/Downloads/documents%2FMScA Statistical Analysis 31007%2FMScA 31007 Lecture
```

4.2. Simulate pseudo-random poins [x,y][x,y] on [0,100]×[0,100]

Select a number o points nSample.

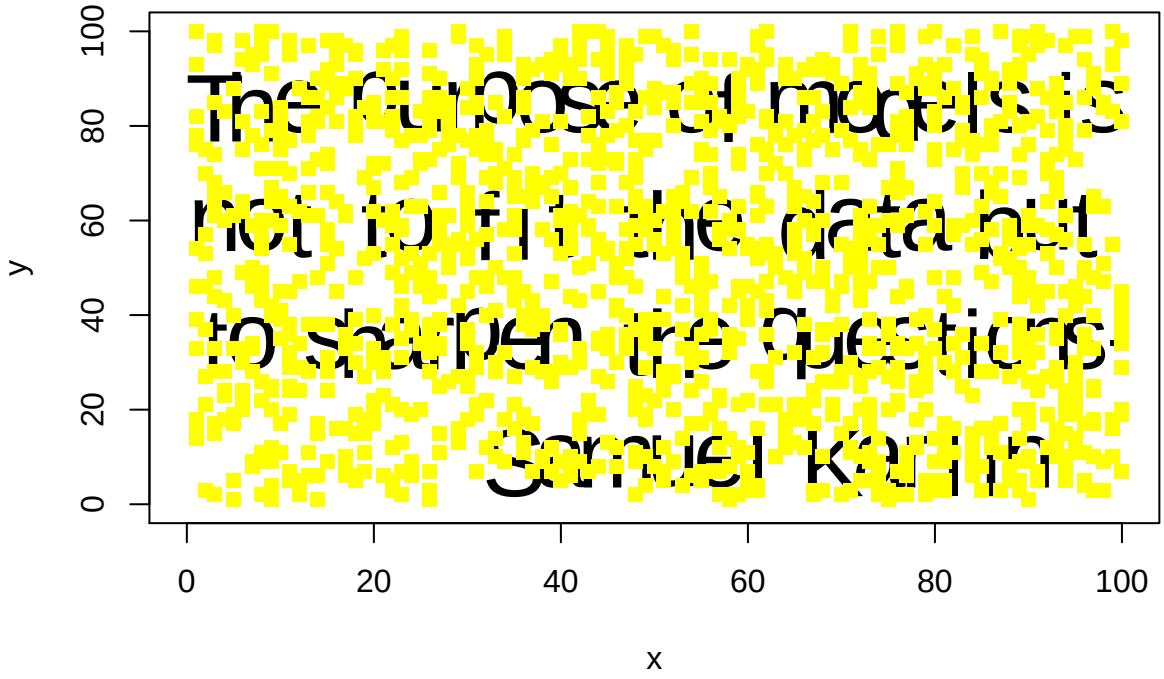
Simulate a sample of length 2*nSample from uniform distribution on [0,100] and turn it into a (nSample×2)(nSample×2) matrix. Use a seed of your choice my.seed.

```
##Set sample and seed, use runif to create random values from 0 to 100 and put
##into a matrix with two columns
nSample <- 20000
my.seed <- 5555
set.seed(my.seed)
xy<-runif(2*nSample,0,100)
xy<-matrix(xy,ncol=2)
head(xy)
```

```
##           [,1]     [,2]
## [1,] 22.089697 54.15111
## [2,] 12.923022 91.32639
## [3,] 32.718565 83.45741
## [4,] 16.986400 20.32621
## [5,]  8.011574 27.21023
## [6,]  6.727130 28.40253
```

Throw nSample simulated points on square [0,100]×[0,100][0,100]×[0,100] to scratch off some of yellow paint.

```
##run scratchoff function using the matrix xy
ScratchOffMonteCarlo(xy)
```




```
## [1] "Size = 20000"    "Open (%)= 86.64"
```

Take a note of the percentage scratched off returned by ScratchOffMonteCarlo(xy)

By changing nSample and my.seed try to make the quote of the day readable with minimum sample size. What percent you needed to scratch off to make the quote readable?

I needed over 86% to make the quote readable, which was nSample <- 20000

4.3. Simulate quasi-random poins [x,y][x,y] on [0,100]×[0,100]

```
##install randtoolbox package
library(randtoolbox)
```

```
## Loading required package: rngWELL
```

```
## This is randtoolbox. For overview, type 'help("randtoolbox")'.
```

```
##
## Attaching package: 'randtoolbox'
```

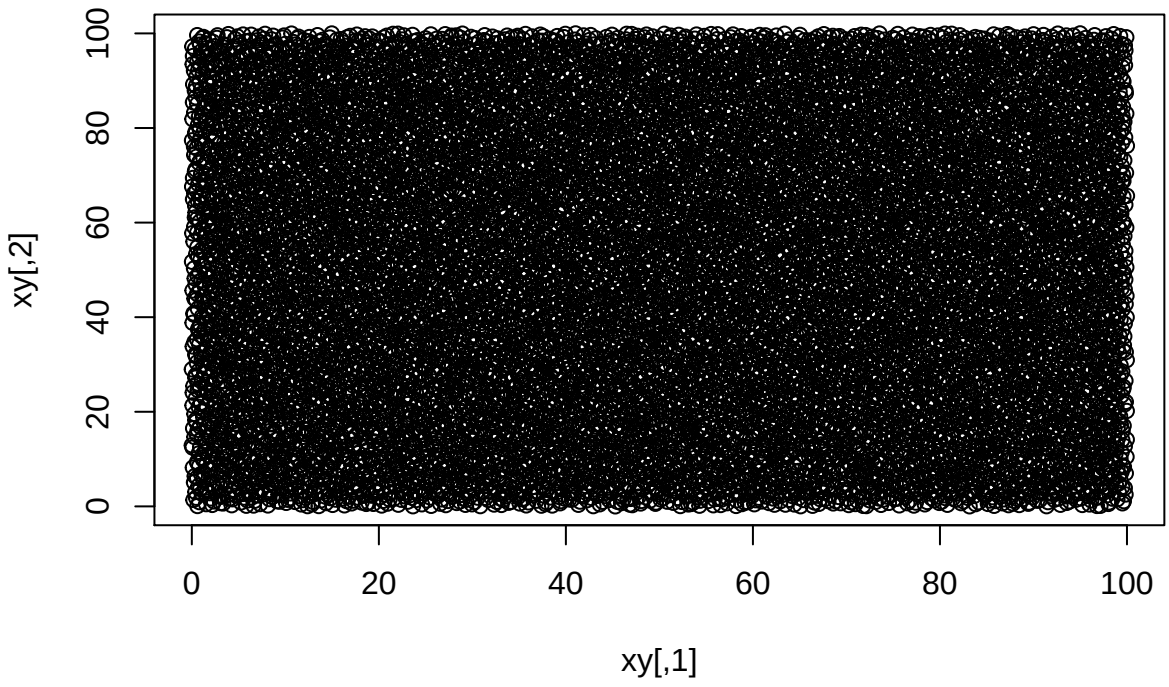
```
## The following object is masked from 'package:randtests':
##
##     permut
```

Run sobol() first time with the default set for parameter init=T.

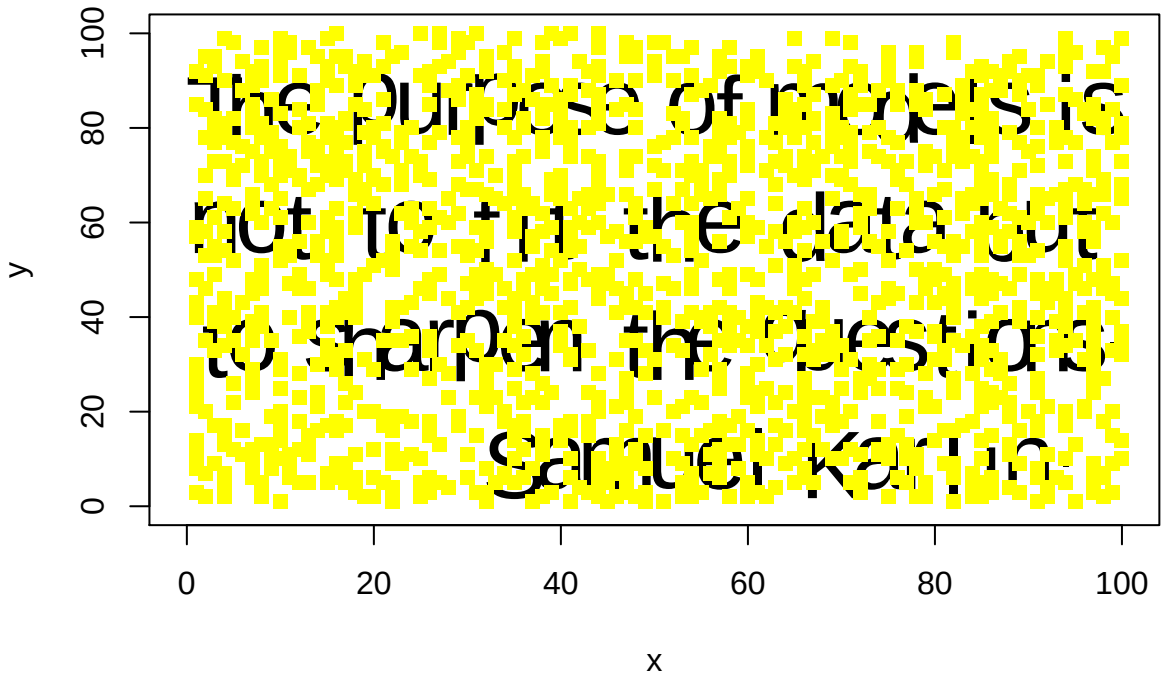
```
##set seed and nsample
my.seed<-6565
set.seed(my.seed)
nSample<-10
## use sobol function to create a 2 column matrix with 10 rows
xy<-sobol(nSample,dim=2,init=T)*100
```

Then make init=F if you want to generate different sample every time or keep it equal to T if you want repeated samples.

```
##run using a different seed and sample
my.seed<-123
nSample<-13000
xy<-sobol(nSample,dim=2,init=T,scrambling = T,seed=my.seed)*100
##Plot the data pairs and run ScratchOffMonteCarlo
plot(xy)
```




```
ScratchOffMonteCarlo(xy)
```




```
## [1] "Size = 13000"    "Open (%)= 86.31"
```

Again, by changing nSample and my.seed try to make the quote of the day readable with minimum sample size. What percent you needed to scratch off to make the quote readable? Again, I needed about 86% open to make it readable

Which of the Monte Carlo methods makes the quote readable sooner? The Sobol method was able to reach a readable percentage with a smaller nSample size

Which parameters nSample and my.seed gave you the best result, what percent of the yellow paing you were able to scratch off by each method? For runif(), I used my.seed <- 5555 with nSample <- 20000 For Sobol, I used my.seed <- 123 with nSample <- 13000 Both got the percent open to around 86% *Changing which of the two parameters plays more significant role?* Changing nSample played a more significant role.