DATA 604 - Homework #3

J. Hamski

September 26, 2016

```
library(magrittr)
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.2.5
```

1.

```
LCG.1 <- function(seed, a, m){
  X.i <- (seed * a) %% m
  return(X.i)
}

rand.list.1 = 1
a = 11
m = 16

cnt = 0
while (cnt < 3) {
  rand.list.1 <- c(rand.list.1, LCG.1(rand.list.1[cnt], a, m))
  cnt <- cnt + 1
}

ifelse(rand.list.1[1:2] == rand.list.1[(cnt - 1):cnt], break, rand.list.1 <- c(rand.list.1, LCG.1(rand.l
```

```
## [1]  1 11
```

```
rand.list.1
```

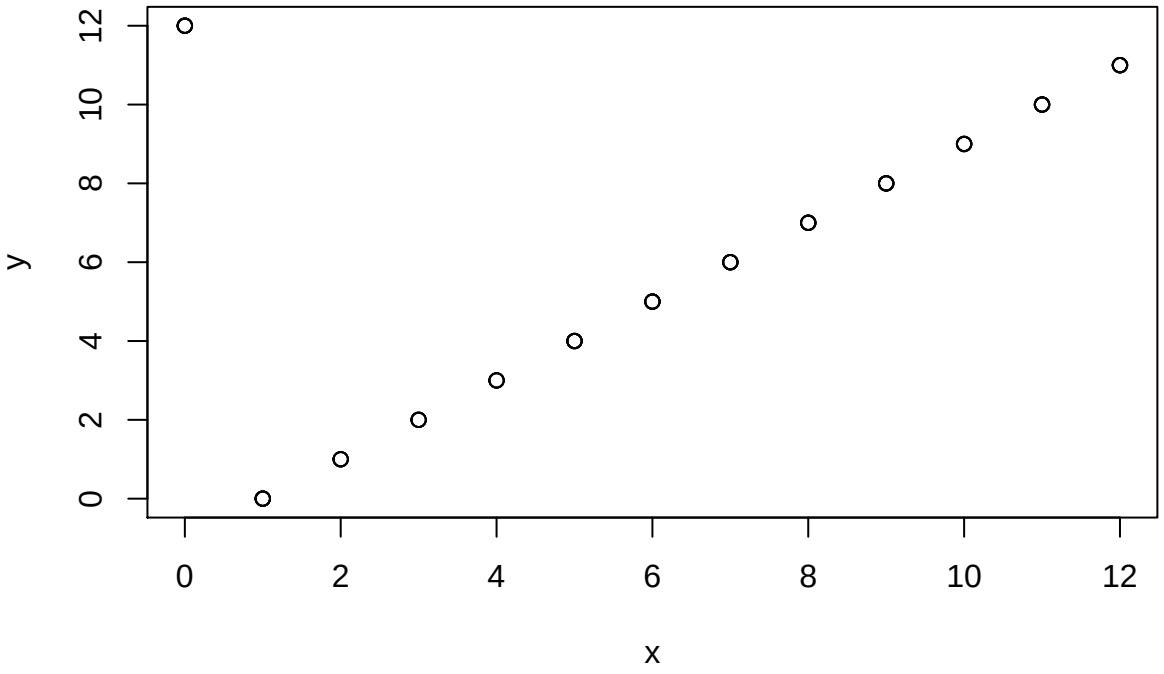
```
## [1]  1 11  9  3
```

2.

```
mod.function <- function(X.i1, a, c, m){return((a*X.i1 + c) %% m)}

a = 1
c = 12
m = 13
X.0 = 1
gen.size = 50
rand.list.2 = X.0

for (i in 1:gen.size){
  rand.list.2[i+1] <- mod.function(rand.list.2[i], a, c, m)
}

rand.list.2
```

```
##  [1]  1  0 12 11 10  9  8  7  6  5  4  3  2  1  0 12 11 10  9  8  7  6  5
## [24]  4  3  2  1  0 12 11 10  9  8  7  6  5  4  3  2  1  0 12 11 10  9  8
## [47]  7  6  5  4  3
```

```
x <- rand.list.2[1:gen.size-1]
y <- rand.list.2[2:gen.size]
plot(x, y)
```



3. Referenced used: http://www1.maths.leeds.ac.uk/~voss/projects/2012-RNG/Burgoine.pdf however my code was developed independently.

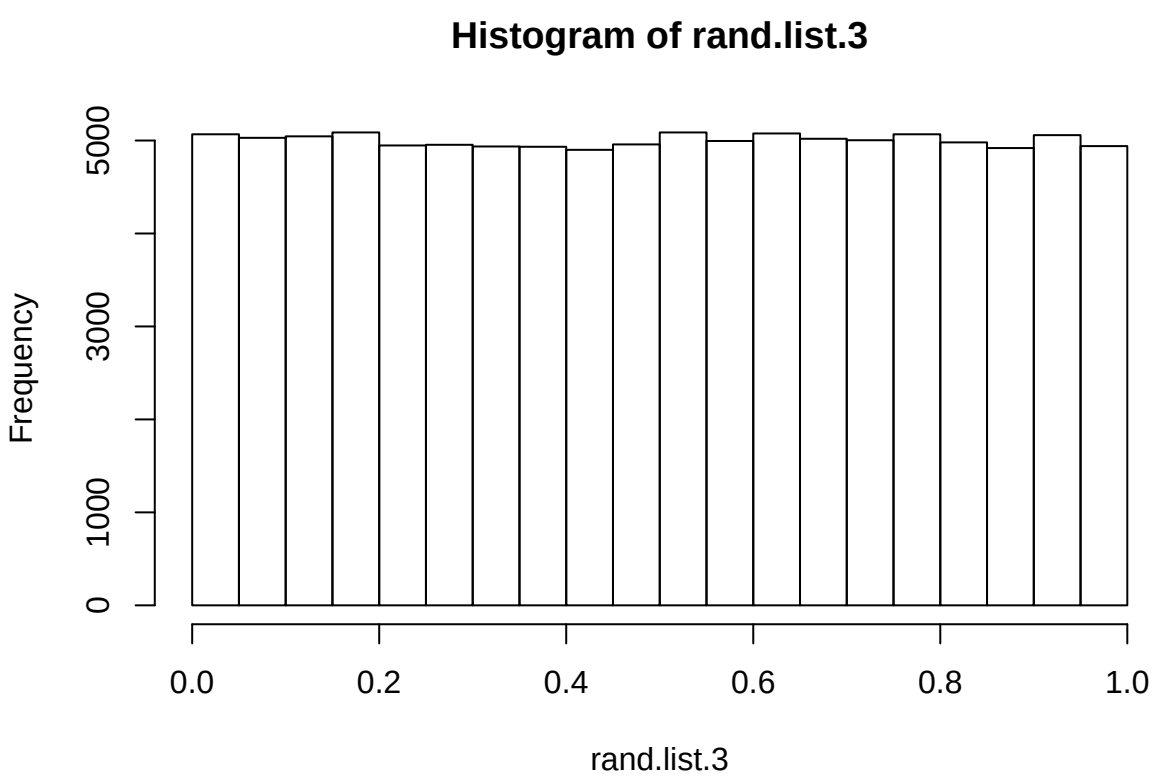
```
a = 16807
c = 0
m = (2 ** 31) - 1
X.0 = 1234567
obs = 100000
rand.list.3 = X.0

for (i in 1:(obs-1)){
  rand.list.3[i+1] <- mod.function(rand.list.3[i], a, c, m)
}

rand.list.3 <- rand.list.3 / m
mean(rand.list.3)
```

```
## [1] 0.4995559
```

```
var(rand.list.3)
```

```
## [1] 0.08346409
```

```r
hist(rand.list.3)
```




```r
binned.random.numbers <- by(rand.list.3, cut(rand.list.3,20),FUN=sum)
```

4. Inverse transform

```r
normrandit <- function(){
  R <- runif(1)
  normrand <- ((R ** 0.135) - (1 - R ** 0.135)) / 0.1975
  return(normrand)
}

itstats <- function(N){
  normrandit.samples <- replicate(N, normrandit())
  return(c(mean(normrandit.samples), sd(normrandit.samples)))
}
```

Box-Muller algorithm

6.

```r
inside.circle <- function(x){
  ifelse(x[1]^2 + x[2]^2 < 1, return(1), return(0))}
```

```r
estimate.pi <- function(N){
```

```
  x <- runif(N)
  y <- runif(N)
  rand.samples <- matrix(x, y, ncol = 2, nrow = N)

  results.list <- apply(rand.samples, 1, function(x) inside.circle(x))

  pi.estimate <- sum(results.list)/length(results.list) * 4

  se <- sd(results.list) / sqrt(length(results.list)) * 4
  conf.int.up <- pi.estimate + 1.96 * se
  conf.int.low <- pi.estimate - 1.96 * se

  output <- c(N, pi.estimate, se, conf.int.up, conf.int.low)
  return(output)
}
```

```
test.sample.sizes <- seq(from = 1000, to = 10000, by = 500)

#this would be better with an apply function, but using a loop for now.
Ncnt <- NULL
estimate <- NULL
st.error <- NULL
upper.conf <- NULL
lower.conf <- NULL

for(i in 1:length(test.sample.sizes)){

  size.results <- estimate.pi(test.sample.sizes[i])

  Ncnt[i] <- size.results[1]
  estimate[i] <- size.results[2]
  st.error[i] <- size.results[3]
  upper.conf[i] <- size.results[4]
  lower.conf[i] <- size.results[5]
}
sample.size.results <- cbind(Ncnt, estimate, st.error, upper.conf, lower.conf)
```

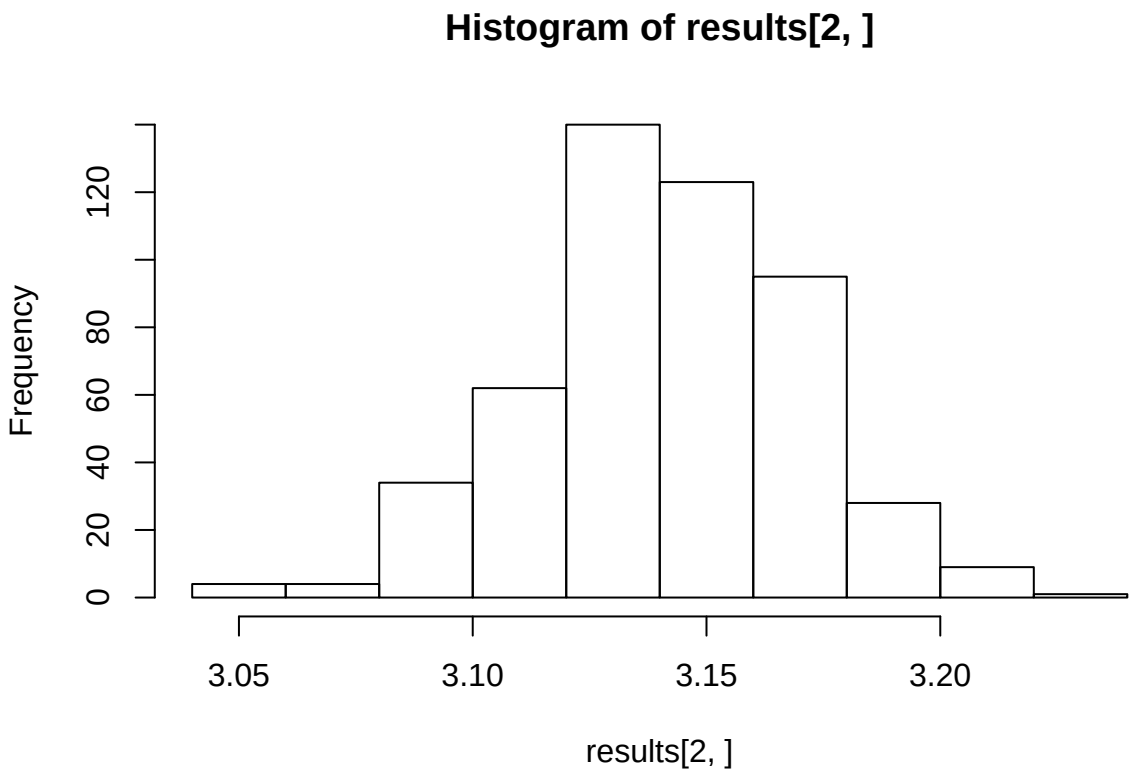
```
colnames(sample.size.results) <- c("N", "estimate", "st.error", "upper.conf", "lower.conf")
kable(sample.size.results)
```

N	estimate	st.error	upper.conf	lower.conf
1000	3.152000	0.0517259	3.253383	3.050617
1500	3.077333	0.0435220	3.162637	2.992030
2000	3.136000	0.0368162	3.208160	3.063840
2500	3.177600	0.0323376	3.240982	3.114218
3000	3.109333	0.0303881	3.168894	3.049773
3500	3.106286	0.0281675	3.161494	3.051077
4000	3.156000	0.0258086	3.206585	3.105415
4500	3.137778	0.0245224	3.185842	3.089714
5000	3.153600	0.0231073	3.198890	3.108310
5500	3.179636	0.0217796	3.222324	3.136948
6000	3.141333	0.0212046	3.182894	3.099772

N	estimate	st.error	upper.conf	lower.conf
6500	3.186462	0.0199719	3.225607	3.147317
7000	3.115429	0.0198430	3.154321	3.076536
7500	3.114667	0.0191760	3.152252	3.077082
8000	3.145000	0.0183348	3.180936	3.109064
8500	3.148235	0.0177627	3.183050	3.113420
9000	3.148444	0.0172607	3.182275	3.114614
9500	3.141895	0.0168472	3.174915	3.108874
10000	3.138400	0.0164448	3.170632	3.106168
It takes	7500 sampl	es before we	can be confi	dent we're within 0.01 of the actual value of pi.

```
results <- replicate(500, estimate.pi(7500))
```

```
hist(results[2,])
```



This resembles the normal distribution because of the central limit theorem.

```
(se <- sd(results[2,]) / sqrt(500))
```

```
## [1] 0.001260669
```

```
length(which(results[2,] < 3.168006 & results[2,] > 3.093327)) / 500 * 100
```

```
## [1] 80.4
```

Approximately 81% of my results are within the 95% confidence interval.