Stat 550 Jan Domingo

Final : Lab Take-home May 16, 2019 Section 2 TTH 1600-1715

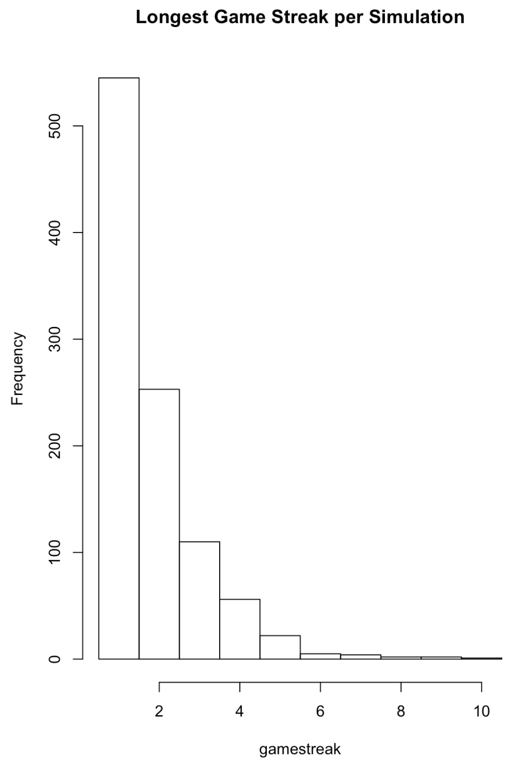
Your solutions should contain clearly identified numerical answers and well labeled tables/figures, where appropriate. Make sure to include clear and concise interpretations (that is, clear, full sentences) of any values, figures, and responses to queries posed. Place all code in an appendix, at the end of your solutions document.

1. Roulette wheel simulation: A roulette wheel has 38 slots of which 18 are red, 18 are black, and 2 are green. If a ball spun on to the wheel stops on the color a player bets, the player wins. Consider a player betting on red. Game streaks follow a Geometric(p=20/38) distribution derivation of the Geometric distribution from the Bernoulli distribution to simulate the game. Namely, generate Bernoulli (p=20/38) random variates until a black or green occurs. Run 1000 simulations and report the following:

a) What was the longest game streak until a loss?

The longest streak was 10. *See attached appendix*

b) Plot the histogram of game streak lengths



c) Average of the game streaks

The average of the game streaks was 1.817. *See attached appendix*

d) Variance of the game streaks

The variance of the game streaks was 1.45897. *See attached appendix*

e) Compare (c) and (d) to the true mean and variance from a Geometric distribution with p = 20/38

The true mean is $E[X] = \frac{1}{p} = \frac{1}{\frac{20}{38}} = 1.9$

The true variance is $Var[x] = \frac{1-p}{p^2} = \frac{1-\left(\frac{20}{38}\right)}{\left(\frac{20}{38}\right)^2} = 1.71$

Simulated mean: 1.817

Simulated Variance: 1.45897

The simulated mean had an absolute difference of .083 from the true mean

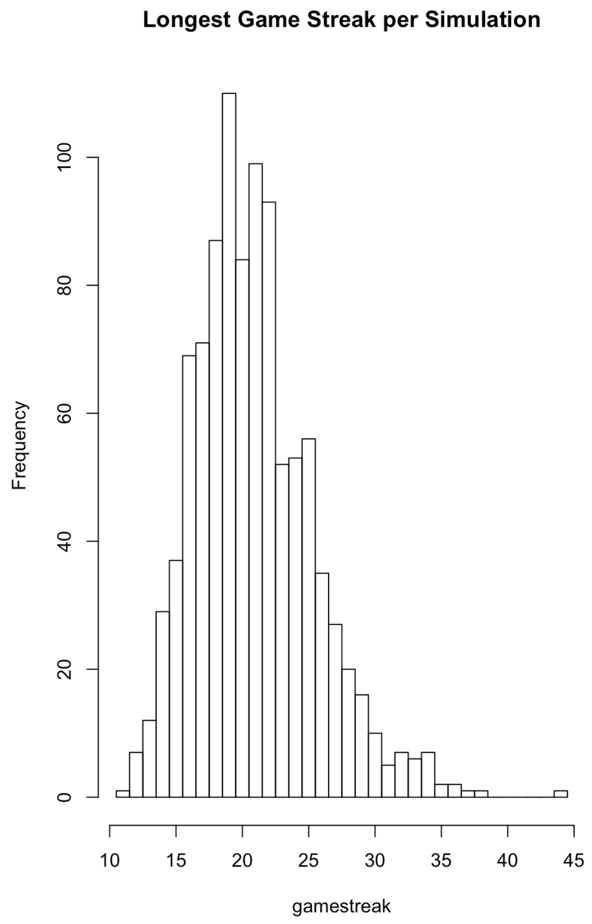
The simulated variance had an absolute difference of .25103 from the true variance

2. Simulating negative binomial distributions: Recall that a negative binomial random variable NegBin(r,p) is the sum of r Geometric(p) random variables. Suppose the player from the first problem decides to play until the player loses 10 times (i.e., until we get 10 black or green spins). Modify the algorithm in problem 1 to simulate 1000 NegBin(10, 20/38) random variates and report the following:

a) What was the longest game streak until 10 black or green spins?

The longest streak was 17. *See attached appendix*

b) Plot the histogram of the game streak lengths



c) Average of the game streaks

The average of the game streaks was 20.891. *See attached appendix*

d) Variance of the game streaks

The variance of the game streaks was 20.00913. *See attached appendix*

e) Compare (c) and (d) to the true mean and variance from a Negative Binomial distribution with r = 10 and p = 20/38.

True mean: $E[X] = \frac{r}{p} = \frac{10}{\frac{20}{38}} = 19$

True variance: $Var[X] = \frac{r(1-p)}{p^2} = \frac{10\left(1-\left(\frac{20}{38}\right)\right)}{(\frac{20}{38})^2} = 17.1$

Simulated mean: 20.891

Simulated variance: 20.00913

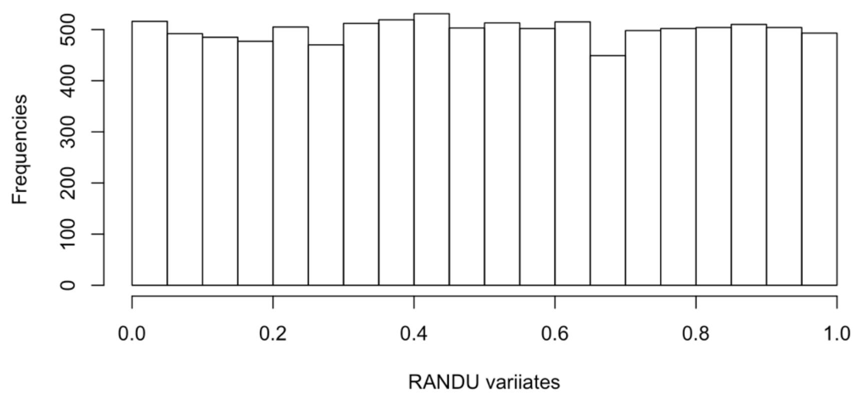
The simulated mean had an absolute difference of 1.891 from the true mean. The simulated variance had an absolute difference of 2.90913 from the true variance.

3. Recall that the Linear congruential generator is defined as

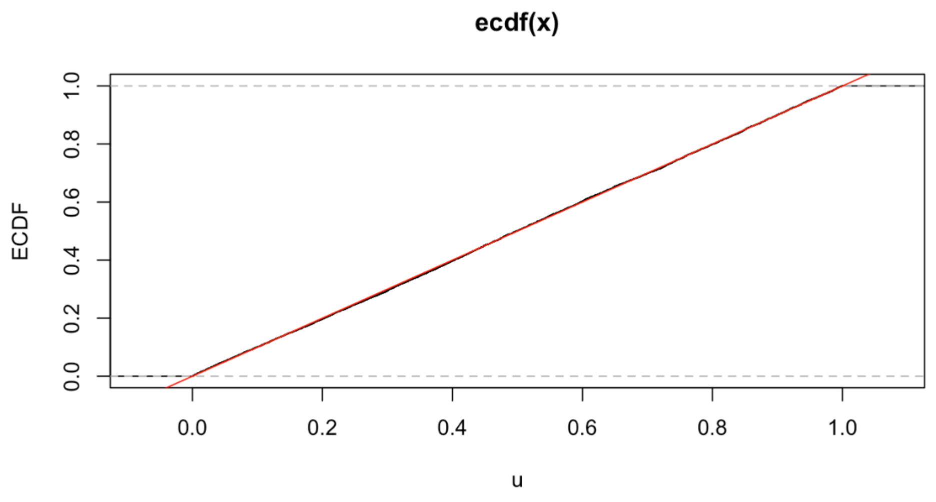
$$X_{n+1} = (aX_n + b) \mod m$$

Where a is our multiplier, b is our increment value, and m is our modulus value and determines the length of our sequence. Then we take this generated sequence of numbers and divide them by m to get a set of variates between 0 and 1. For this problem, pick your own values of a, b, and m and generate 10,000 variates using your random number generator. Then do the following:

a) Plot a histogram of your variates

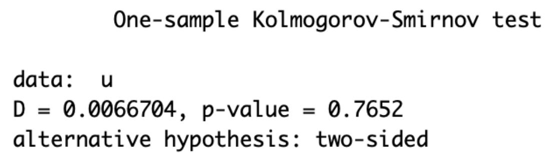


b) Draw the empirical CDF of your variates against the true CDF of a uniform distribution



c) Perform your Kolmogorov-Smirnov test to validate your random number generator. Do your pseudorandom numbers seem to follow a uniform distribution? Base your conclusions from the test at an $\alpha = 0.05$

The pseudo random numbers I chose does seem to follow a uniform distribution. With a p value .7652 and is greater than the value of alpha (0.05) therefore we fail to reject the null hypothesis and the result is statistically nonsignificant.



```
                One-sample Kolmogorov-Smirnov test

data:  u
D = 0.0066704, p-value = 0.7652
alternative hypothesis: two-sided
```

4. Simulating Normal probabilities. Finding areas under the normal curve is analytically quite difficult to the form of the pdf. We can use Monte Carlo estimates via simulation. One such method is the Box-Müller algorithm. This simulator applys a transformation method to convert uniform variates into normal variates by generating two random variates $Z_1$ and $Z_2$ from a N(0,1) distribution. The Box-Müller steps to simulate a normal distribution with mean $\mu$ and variance $\sigma^2$ are as follows:

i. Generate $U_1, U_2 \sim U(0,1)$, two uniform random variates

*See attached appendix*

ii. Compute $Z_1 = \sqrt{-2\ln(U_1)}\cos(2\pi U_2)$ and $Z_2 = \sqrt{-2\ln(U_1)}\sin(2\pi U_2)$

*See attached appendix*

iii. Compute$X_1 = \sigma Z_1 + \mu$and$X_2 = \sigma Z_2 + \mu$.

*See attached appendix*

Use the Box-Müller algorithm to simulate 1000 random variates of $X_1$ and 1000 random variates of $X_2$ that follow a normal distribution with $\mu = 2$ and $\sigma^2 = 2$, i.e. $X_1 \sim N(2,2)$. Then report the following:

a) The mean, variance, and median of the combined variates of $X_1$ and $X_2$

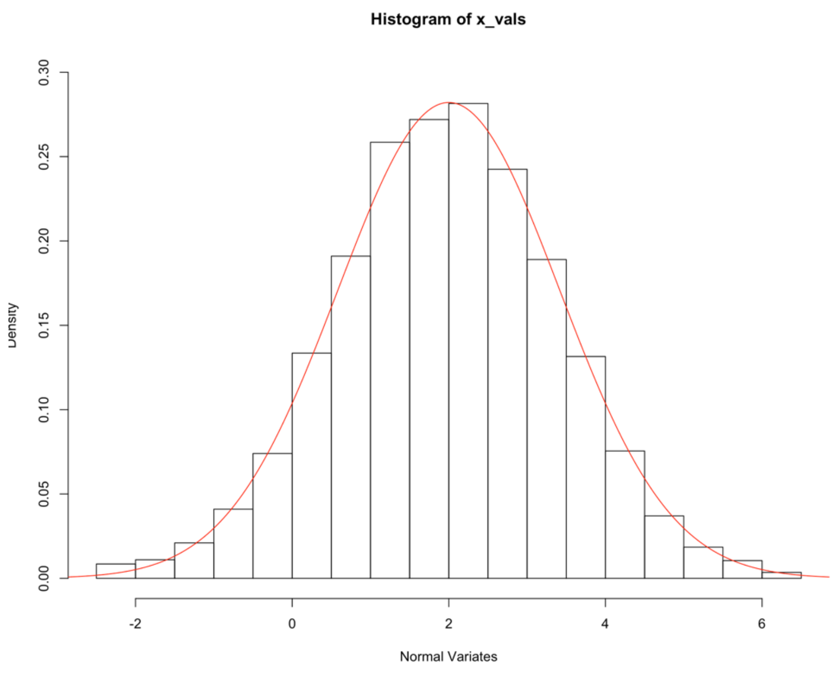
mean: 1.975361

variance: 1.985998

median: 1.982107

*See attached appendix*

b) Histogram of these combined variates and an overlay with the true density



c) Find the proportion of simulated values of these combined variates that lie within one, two, and three standard deviations from the mean (i.e., they are within $\mu \pm \sigma$, $\mu \pm 2\sigma$, and $\mu \pm 3\sigma$). These percentages are estimates of $P(u - i\sigma < X < \mu + i\sigma)$ for $i = 1, 2, 3$.

The proportion of the simulated value that lie within one, two and three standard deviations from the mean are 0.6845, 0.952, and 0.9975 respectively.
*See attached appendix*

d) The true values of the three probabilities in c are 0.68, 0.95, and .997, respectively. This is known as the Empirical rule or "68-95-99.7" rule. Compare your estimates from (c) to these true values

| | |
|---|---|
| True values: | 0.68, 0.95, .997 |
| Simulated values: | 0.6845, 0.952, 0.9975 |

These values are very close to each other with an absolute value difference of only .0045, .002, and .0005 respectively and with our values being slightly higher.

```r
# APPENDIX
# Jan Domingo
# May 16, 2019
# Final: Lab Take-home

# -------------------------------------------------------------------------------------------------------------

#1
#(a) What was the longest game streak until a loss?
p = 20/38
n = 1000 #Amount of simulations
gamestreak = vector()
longestStreak = 0

for (i in 1:n) {
  streak = 0
  lose = 0
  while (lose == 0)
  {
    u = runif(1)  #runif(1) generates a random number between 0 and 1
    if (u < p)  #If the random number is less than the probability of winning
      lose = 1  #Stops the simulation if lose = 1
    streak = streak + 1
    if (streak > longestStreak)
      longestStreak = streak
  }
  streak  #Returns the amount of plays until lose
  longestStreak #Returns the longest streak in the simulation

  gamestreak = c(gamestreak, streak)
}

#(b) Plot the histogram of game streak lengths
hist(gamestreak, br=seq(min(gamestreak)-.5, max(gamestreak)+.5), main = "Longest Game Streak per Simulation")

#(c) Average of the game streaks
mean(gamestreak)

#(d) Variance of the game streaks
var(gamestreak)

# -------------------------------------------------------------------------------------------------------------

#2
#(a) What was the longest game streak until 10 black or green spins?
p = 20/38
n = 1000
gamestreak = vector()
longestStreak = 0

for (i in 1:n) {
  streak = 0
  lose = 0
  while (lose <= 10) #This line is different from #1a. Only stops simulation when lose reaches 10
  {
    u = runif(1)  #Generates a random number between 0 and 1
    if (u < p)    #If the random number is less than the proabability of winning
      lose = lose + 1 #Then increase the lose count
    streak = streak + 1
    if (streak > longestStreak)
      longestStreak = streak
  }
  streak #Returns the longest streak until 10 black or green spins
  longestStreak #Returns the longest streak in the simulation
  gamestreak = c(gamestreak, streak)
}

#(b) Plot a histogram of the game streak lengths
hist(gamestreak, br=seq(min(gamestreak)-.5, max(gamestreak)+.5), main = "Longest Game Streak per Simulation")

#(c) Average of the game streaks
mean(gamestreak)

#(d) Variance of the game streaks
var(gamestreak)

# -------------------------------------------------------------------------------------------------------------
```

```
#3
n = 10000 #number of variates
x = 1     #seed value

for (i in 1:n) {
  x = c(x, ((31425*x[i]+100)%%(2^29))) #Linear Congruential Method with custom a(31425), b(100), and m(2^29)
values
}

x = x[2:(n+1)]  #Disregards seed value, keeps the other 10000 from for loop
x

u = x/(2^29)  #Transform uniform variates between 0 and 1

#(a) Plot a histogram of your variates
par(mfrow = c(2,1)) #2 rows, 1 column for the graph matrix
hist(u, main ="", xlab="RANDU variiates", ylab="Frequencies") #Histogram of the n RANDU variates


#(b) Draw the empirical CDF of oyur variates against the true CDF of a uniform distribution
plot.ecdf(u, verticals = TRUE, do.p = FALSE, xlab = "u", ylab = "ECDF")
abline(0,1, col="red")

#(c)
ks.test(u, "punif", 0, 1) #Komolgorov-smirnov test of RANDU variates against U(0,1)

# ------------------------------------------------------------------------------------------------------------

#4

x1_v = vector()
x2_v = vector()
n = 1000
x_vals = vector()
m = 2 #mean
v = 2 #variance
s = sqrt(v) #standard deviation

#The following i, ii, and iii are the steps for the Box-Muller Method
#i Generates two uniform variates
for (i in 1:n) {
  u1 = runif(1)
  u2 = runif(2)

#ii Convert Variates to z scores
z1 = sqrt(-2*log(u1))*cos(2*pi*u2)
z2 = sqrt(-2*log(u1))*sin(2*pi*u2)

#iii Inverse normal transformation to x variates
x1 = s * z1 + m
x2 = s * z2 + m

x1_v = c(x1_v, x1)
x2_v = c(x2_v, x2)
}

x_vals = c(x1_v, x2_v)

#(a) The mean, variance, and median of the combined variates of x1_v and x2_v
mean(x_vals)
var(x_vals)

median(x_vals)

#(b) Histogram of these combined variates and an overly with the true density
x <- seq(-4, 8, by=0.001) #Generate a sequence of numbers from -4 to 8
y <- dnorm(x, m, s) #Normalize the values with mean=2 and pop sd=sqrt(2)
hist(x_vals, freq = FALSE, xlab="Normal Variates", ylim=c(0,0.30))      #Creates histogram with density values
lines(x,y,col = "red") #Overlay the true density curve over the graph

#(c) Calculating the empirical rule probabilities
i = 1
length(x_vals[(m-i*s) <= x_vals & x_vals <= (m+i*s)]) / length(x_vals)
i = 2
length(x_vals[(m-i*s) <= x_vals & x_vals <= (m+i*s)]) / length(x_vals)
i = 3
length(x_vals[(m-i*s) <= x_vals & x_vals <= (m+i*s)]) / length(x_vals)
```