Homework 5

Zachary Lazerick

24 February 2023

1) Convolutions of random variables – For each situation below, generate 7500 realizations of the random variable and then compare the mean, variance, and pdf of your simulated values with the theoretical values. For the pdf, superimpose the true density curve over a histogram of your simulated values.

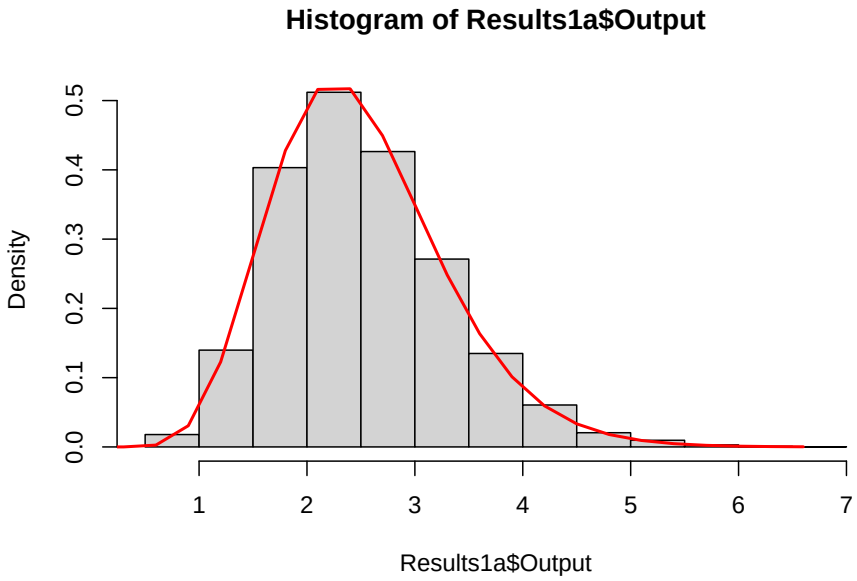
a) A gamma distribution with parameters $\alpha = 10$ and $\beta = 4$. Use inverse CDF sampling in conjunction with the fact that the sum of n independent Exp(β) random variables has a gamma($\alpha = n, \beta$) distribution. Note: The R command for gamma density function is: dgamma(x, shape=a, scale=b)

```
GammaConv <- function(iterations, alpha, beta) {
  X <- c(); counter <- 0
  while (length(X) < iterations) {
    Y <- rexp(alpha, rate = beta)
    X <- append(X, sum(Y)); counter <- counter + 1
  }
  mylist <- list("Output" = X, "Proposals" = counter)
  return(mylist)
}

set.seed(255)
Results1a <- GammaConv(7500, 10, 4)

## Theoretical Mean and Variance for a Gamma(10, 4):
## Mean = (10)(4) = 40; Variance = (10)(4)^2 = 160
mean(Results1a$Output) > 40; var(Results1a$Output) > 160
```

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

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

```
hist(Results1a$Output, freq = FALSE)
X <- seq(0, max(Results1a$Output), .3); Gamma <- dgamma(X, 10, 4)
lines(X, Gamma, col = 'red', lwd = 2)
```



b) The negative binomial distribution is used to describe the probability distribution for the number of trials until the rth success, so a negative binomial random variable can be simulated as the sum of r geometric(p) random variables. Use $r = 5$ and $p = 0.6$. Notes:

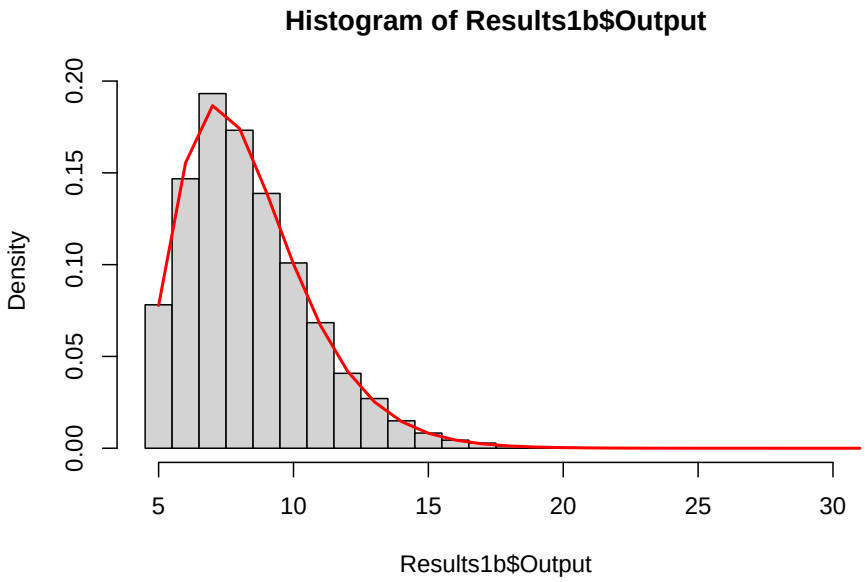
- The R Command for negative binomial density function is dnbinom(x, size = r, prob = p) However, this function simulates the number of failures until the rth succes and not the number of trials until the rth success, so the two random variables are defined differently. To account for this, the "x" in the dnbimon() function should be a sequence of integers from zero to the maximum
- The density function will not match your histogram unless: a. You change the default bin size of the histogram. Use breaks=seq(4.5, max(X)+1, 1) b. You use "lines(x+r, Y)", where "x" is the input and "Y" the output of the dbinom() function shown above.

```
NegBinomConv <- function(iterations, r, p) {
  X <- c(); counter <- 0
  while (length(X) < iterations) {
    Y <- rgeom(r, prob = p); Z <- sum(Y) + r
    X <- append(X, Z); counter <- counter + 1
  }
  mylist <- list("Output" = X, "Proposals" = counter)
  return(mylist)
}

set.seed(127)
Results1b <- NegBinomConv(7500, 5, .6)

## Theoretical Mean and Variance for a NegBinom(5, .6):

## Mean = (5)/(.6) = 8.33; Variance = (5)/(.6^)^2 = 500/36
mean(Results1b$Output) > 3.33; var(Results1b$Output) > 5.55
```

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

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

```
hist(Results1b$Output, breaks = seq(4.5, 30.5, by = 1), freq = FALSE)
X <- seq(0, max(Results1b$Output), 1); NegBi <- dnbinom(X, size = 5, prob = .6)
lines(X+5, NegBi, col = 'red', lwd = 2)
```



c) A χ^2 (v) random variable can be written as the sum of v independent χ^2 random variables, each of which can be simulated as a squared normal random variable. Use v=8 for this simulation. Notes:

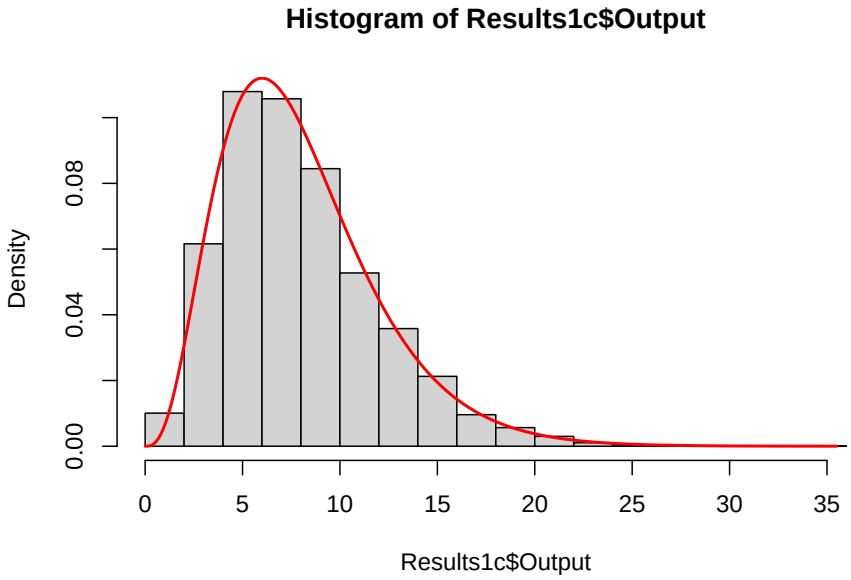
- You can use the built-in normal random number generator rnorm() to simulate your normal random variables
- The R command for the χ^2 density function is: dchisq(x, df)

```
ChisqConv <- function(iterations, v) {
  X <- c(); counter <- 0
  while (length(X) < iterations) {
    Y <- rnorm(v); Z <- Y^2
    X <- append(X, sum(Z)); counter <- counter + 1
  }
  mylist <- list("Output" = X, "Proposals" = counter)
  return(mylist)
}

set.seed(329)
Results1c <- ChisqConv(7500, 8)
## Theoretical Mean and Variance for a Chisq(8):
## Mean = 8; Variance = 2(8) = 16
mean(Results1c$Output) > 8; var(Results1c$Output) > 16
```

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

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

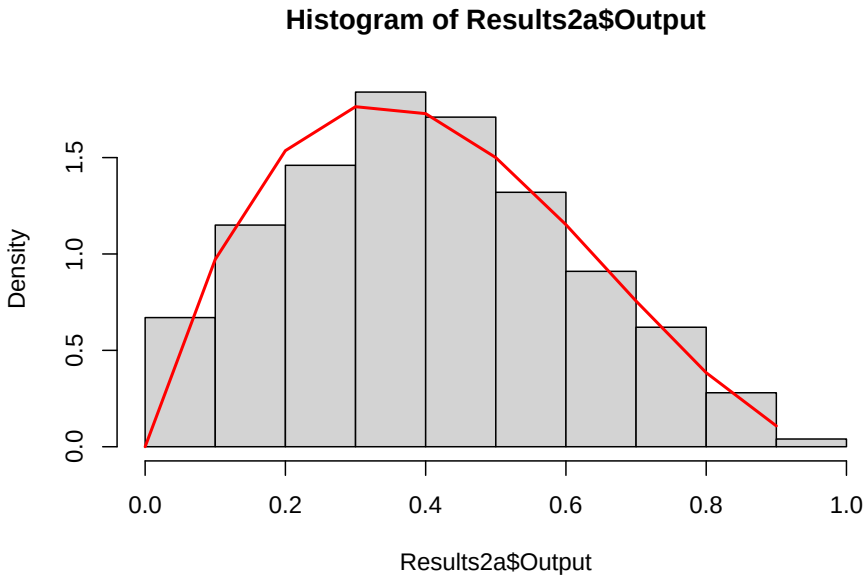
```
hist(Results1c$Output, freq = FALSE)
X <- seq(0, max(Results1c$Output), .1); Chisq <- dchisq(X, 8)
lines(X, Chisq, col = 'red', lwd = 2)
```



2) In class, we talked about how you can simulate a Beta random variable by a transformation of a pair of gamma random variables.

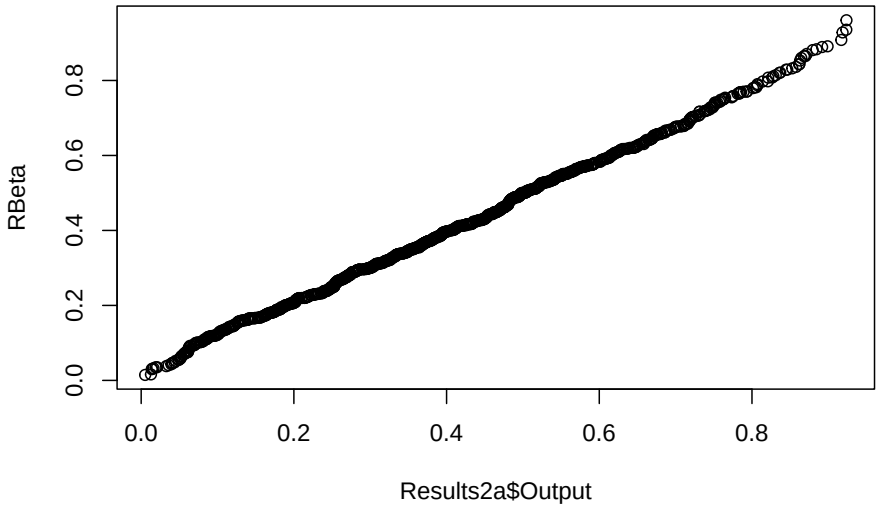
a) Simulate 1000 realizations of a Beta(2,3) random variable in this manner and then make a histogram of your values with the beta(2,3) density function superimposed.

```
GammaSim <- function(iterations, r, s) {
  X <- c(); counter <- 0
  while (length(X) < iterations) {
    U <- rgamma(1, r, rate = 1); V <- rgamma(1, s, rate = 1)
    X <- append(X, (U/(U+V))); counter <- counter + 1
  }
  mylist <- list("Output" = X, "Proposals" = counter)
  return(mylist)
}

Results2a <- GammaSim(1000, 2, 3)

hist(Results2a$Output, freq = F)
X <- seq(0, max(Results2a$Output), .1); Beta <- dbeta(X, 2, 3)
lines(X, Beta, col = 'red', lwd = 2)
```



b) Use the qqplot command in R to compare this approach to the build-in beta random variable generator. In other words, generate 1000 realizations of the beta(2,3) distribution using your simulator from part (a) and the R function rbeta, then run qqplot(sample1, sample2). If the distributions 'match' then this plot should follow a straight line.

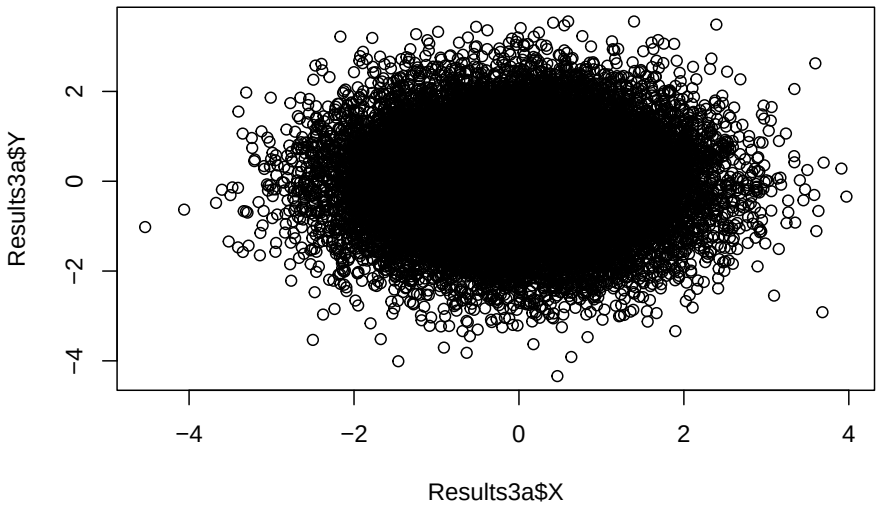
```
RBeta <- rbeta(1000, 2, 3)
qqplot(Results2a$Output, RBeta)
```


RBeta
0.0
0.2
0.4
0.6
0.8
0.0
0.2
0.4
0.6
0.8
Results2a$Output

3) Bivariate Normal Probabilities (Polar Method) – Use the polar method to simulate 25000 pairs (X,Y) of independent, standard normal random realizations.

a) Create a scatterplot of X versus Y and calculate the correlation to verify that the random variables are indeed independent.

```
BiNormPolar <- function(iterations) {
  X <- c(); Y <- c(); counter <- 0
  while (length(X) < iterations) {
    U_1 <- runif(1); U_2 <- runif(1)
    V_1 <- 2*U_1 - 1; V_2 <- 2*U_2 - 1
    if (V_1^2 + V_2^2 <= 1) {
      S <- V_1^2 + V_2^2
      X <- append(X, V_1*sqrt((-2*log(S))/S)); Y <- append(Y, V_2*sqrt((-2*log(S))/S))
      counter <- counter + 1
    }
    else {
      counter <- counter + 1
    }
  }
  mylist <- list("X" = X, "Y" = Y, "Proposals" = counter)
  return(mylist)
}

set.seed(404)
Results3a <- BiNormPolar(25000)

plot(Results3a$X, Results3a$Y)
```




```
## independence implies r close to 0
cor(Results3a$X, Results3a$Y)
```

```
## [1] 0.0008184757
```

b) Use these realizations to find the value of k that satisfies

$$P(\sqrt{X^2 + Y^2} < k) = \frac{1}{2}$$

```
## The value of k that satisfies the above equation is k = 1.172

count <- 0
for (i in 1:length(Results3a$X)) {
  if (sqrt((Results3a$X[i])^2 + (Results3a$Y[i])^2) < 1.172) {
    count <- count + 1
  }
}
count/25000
```

```
## [1] 0.50096
```

c) We said in class that you can estimate the value of π as 4 divided by the average number of iterations. See how close you are!

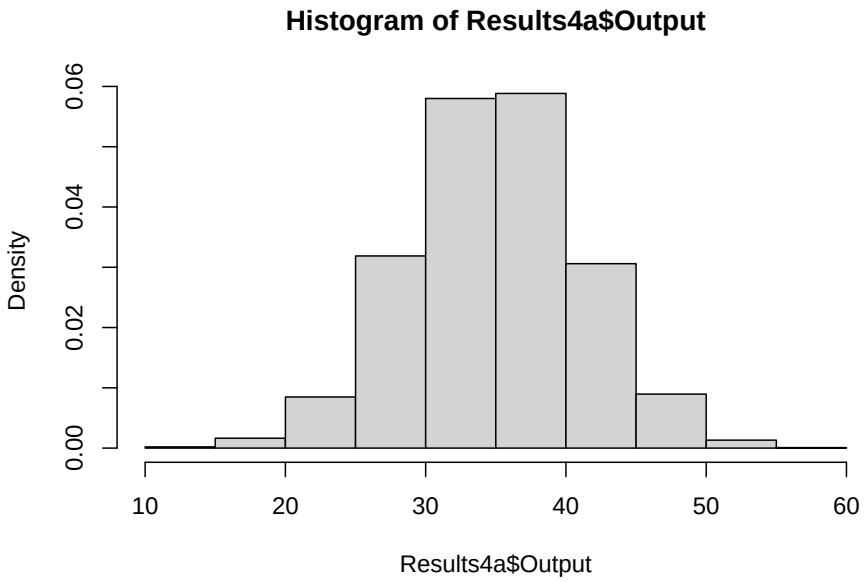
```
E.pi <- 4 / (Results3a$Proposals / 25000)
E.pi
```

```
## [1] 3.149011
```

4) Suppose that $X_1 \sim N(5,2), X_2 \sim N(10,3)$, and $X_3 \sim N(20,5)$ are all normally distributed random variables [Notation: $X_1 \sim N(5,2)$ means that X_1 has a normal distribution with mean 5 and standard deviation 2].

a) Let $Y = X_1 + X_2 + X_3$ be a linear combination of these X's (so you need to generate all three values of X to get a single value for Y). Simulate 5000 realizations of the random variable Y and make a histogram of their values.

```
NormSim <- function(iterations) {
  X <- c(); counter <- 0
  while (length(X) < iterations) {
    X_1 <- rnorm(1, 5, 2); X_2 <- rnorm(1, 10, 3); X_3 <- rnorm(1, 20, 5)
    X <- append(X, X_1 + X_2 + X_3)
  }
  mylist <- list("Output" = X, "Proposals" = counter)
  return(mylist)
}

set.seed(717)
Results4a <- NormSim(5000)
hist(Results4a$Output, freq = F)
```

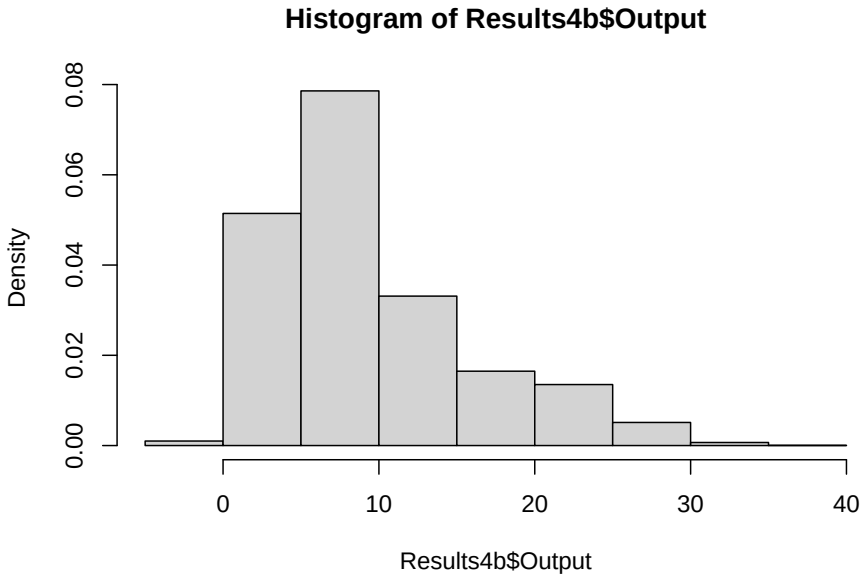


b) This time, suppose that y is a mixture of these three distribution instead (so you randomly choose a density function and then sample a value from that one function, not all three). In other words:

$$f(y) = 0.5f_1(x_1) + 0.3f_2(x_2) + 0.2f_3(x_3)$$

Simulate 5000 realizations of the random variable Y and make a histogram of their values. How do the two histograms compare to each other? To answer this question, make sure to use the same bin size for each set of values and then either (i) make overlapping histograms (see homework #2); or (ii) create two separate histograms whose x- and y-axes are labeled in the same way.

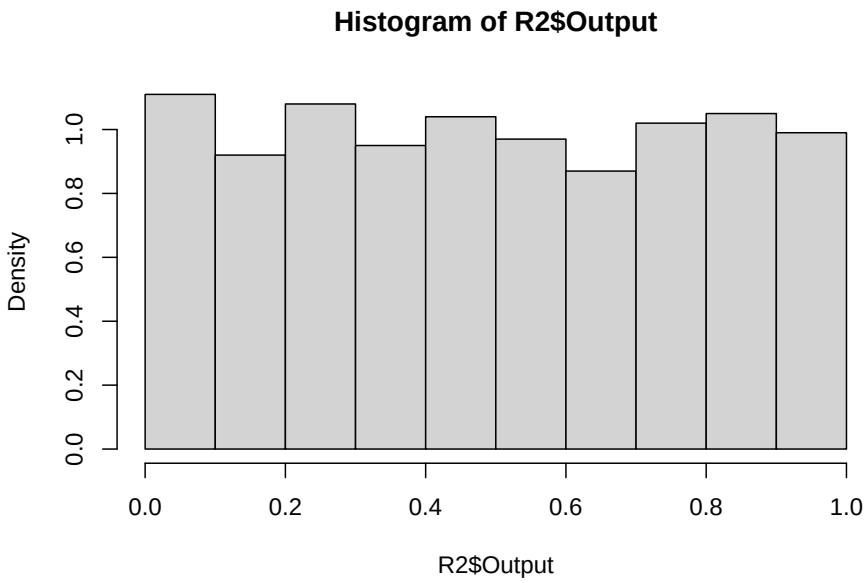
```r
NormSimMix <- function(iterations) {
  X <- c(); counter <- 0
  while (length(X) < iterations) {
    U <- runif(1)
    if (U < .5) {
      X_1 <- rnorm(1, 5, 2)
      X <- append(X, X_1)
      counter <- counter + 1
    }
    else if ((U >= .5) & (U < .8)) {
      X_2 <- rnorm(1, 10, 3)
      X <- append(X, X_2)
      counter <- counter + 1
    }
    else if ((U >= .8) & (U < 1)) {
      X_3 <- rnorm(1, 20, 5)
      X <- append(X, X_3)
      counter <- counter + 1
    }
    else {
      counter <- counter + 1
    }
  }
  mylist <- list("Output" = X, "Proposals" = counter)
  return(mylist)
}

set.seed(1917)
Results4b <- NormSimMix(5000)
hist(Results4b$Output, freq = F)
```



5) Let (X,Y) be uniformly distributed in a circle of radius 1. Show that if R is the distance from the center of the circle to (X,Y), then R^2 is uniform on (0,1).Steps:

- Use the rejection sampler to randomly generate 1000 values in the unit circle
- Create a histogram of the 1000 sampled values and verify that all values of R^2 appear to be equally likely.

```
UnifTest <- function(iterations) {
  X <- c(); Y <- c(); counter <- 0
  while (length(X) < iterations) {
    U_1 <- runif(1); U_2 <- runif(1)
    V_1 <- 2*U_1 - 1; V_2 <- 2*U_2 - 1
    if (V_1^2 + V_2^2 <= 1) {
      R2 <- V_1^2 + V_2^2
      X <- append(X, R2); counter <- counter + 1
    }
    else {
      counter <- counter + 1
    }
  }
  mylist <- list("Output" = X, "Proposals" = counter)
  return(mylist)
}

set.seed(726)
R2 <- UnifTest(1000)

## All Values appear equally likely
hist(R2$Output, freq = F)
```



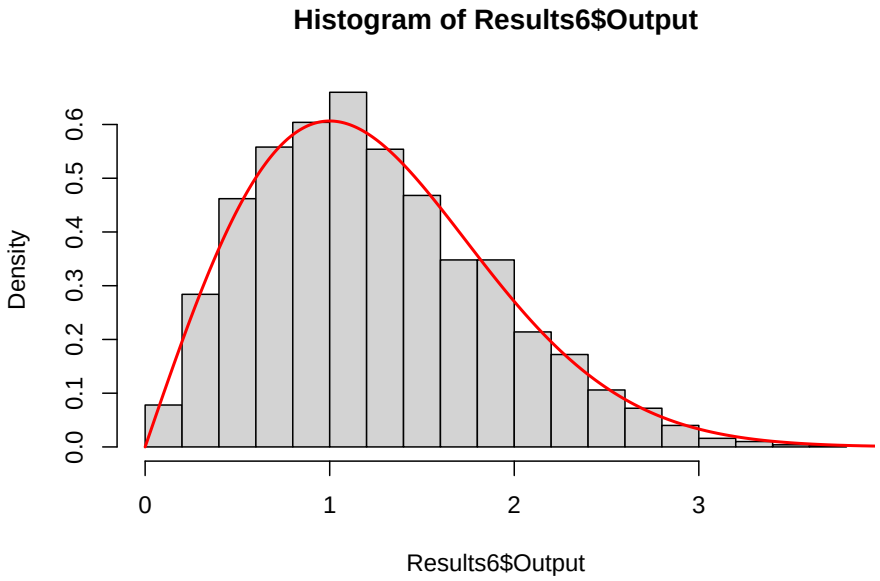
6) Rizzo Book. [Question 3.4 in textbook] The Rayleigh density is

$$f(x) = \frac{x}{\sigma^2} e^{-x^2} 2\sigma^2 x \geq 0, \qquad \sigma > 0$$

Develop an algorithm to generate 2500 random samples from a Rayleigh (σ) distribution. Generate Rayleigh(σ) samples for several choices of $\sigma > 0$ and check that the mode of the generated samples is close to the theoretical mode σ [i.e. check the histogram to verify your algorithm is correct]. The Markdown file you turn in need only show one of these graphs, but it would be good to check several different values for sigma. Hints:

- The Rayleigh density function can be obtained as a simple transformation of another type of RV that, conveniently, is very simple to sample from.
- You can generate the density function yourself using: $x = seq(0, 20, 0.01), y = x/sig^2 * exp(-x^2/(2 * sig^2))$. Or by loading the package vgam and using the drayleigh function.

```
## Rayleigh is chi with v = 2
## Generate 2 Z^2 and sum for chi^2 w/ v = 2
## Take sqrt of chi^2 for chi value
RayleighSamp <- function(iterations) {
  X <- c(); counter <- 0
  while (length(X) < iterations) {
    Y <- rnorm(2); Z <- sum(Y^2)
    X <- append(X, sqrt(Z)); counter <- counter + 1
  }
  mylist = list("Output" = X, "Proposals" = 2)
  return(mylist)
}

Results6 <- RayleighSamp(2500)

hist(Results6$Output, freq = F)
X <- seq(0, 20, 0.01); Rayleigh <- drayleigh(X, scale = 1)
lines(X, Rayleigh, col = 'red', lwd = 2)
```

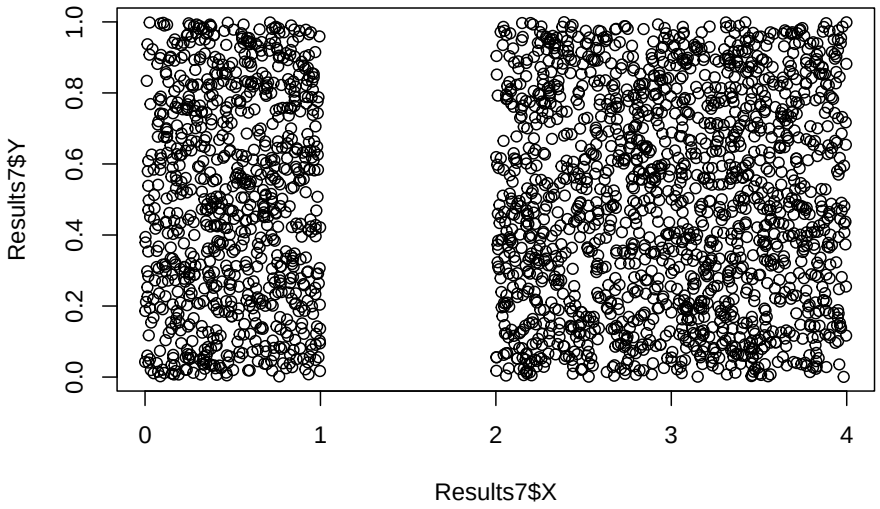

Histogram of Results6$Output
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1
2
3
Results6$Output

7) Voss Textbook:[Question E.1.11 in textbook] Without using rejection sampling, propose a method to sample from the uniform distribution on the set:

$$A = ([0,1]x[0,1]) \cup ([2,4]x[0,1])$$

Write a program implementing your method. Make a scatterplot of 2500 sampled values.

```
## A = [0, 1] X [0, 1]; B = [2, 4] X [0, 1]
## A has Area = 1; B has Area = 2;
## Sample from A 1/3 of time; from B 2/3 of time
RectangleSamp <- function(iterations) {
  X <- c(); Y <- c(); counter <- 0
  while (length(X) < iterations) {
    U <- runif(1)
    if (U <= (1/3)) {
      X_cord <- runif(1, 0, 1); Y_cord <- runif(1, 0, 1)
      X <- append(X, X_cord); Y <- append(Y, Y_cord)
      counter <- counter + 1
    }
    else if (U > (1/3) & (U < 1)) {
      X_cord <- runif(1, 2, 4); Y_cord <- runif(1, 0, 1)
      X <- append(X, X_cord); Y <- append(Y, Y_cord)
      counter <- counter + 1
    }
  }
  mylist <- list("X" = X, "Y" = Y, "Proposals" = counter)
  return(mylist)
}

Results7 <- RectangleSamp(2500)

plot(Results7$X, Results7$Y)
```

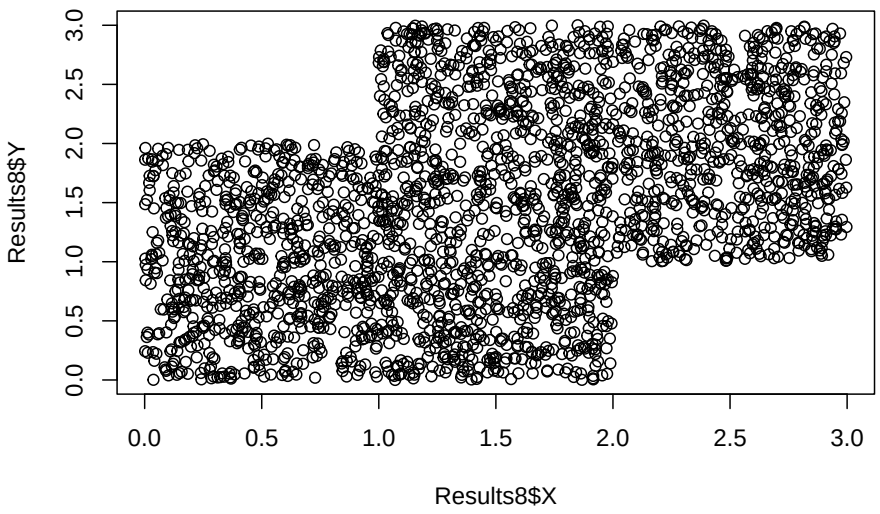


8) [Question E.1.12 in textbook] Without using rejection sampling, propose a method to sample from the uniform distribution on the set:

$$B = ([0,2]x[0,2]) \cup ([1,3]x[1,3])$$

Write a program implementing your method. Make a scatterplot of 2500 sampled values.

```
## A = [0, 1] X [0, 2]; B = [1, 2] X [0, 3]; C = [2, 3] x [1, 3]
## A has Area = 2; B has Area = 3; C has Area = 2
## Sample from A 2/7 of time; from B 3/7 of time; C 2/7 of time
RectangleSamp2 <- function(iterations) {
  X <- c(); Y <- c(); counter <- 0
  while (length(X) < iterations) {
    U <- runif(1)
    if (U <= (2/7)) {
      X_cord <- runif(1, 0, 1); Y_cord <- runif(1, 0, 2)
      X <- append(X, X_cord); Y <- append(Y, Y_cord)
      counter <- counter + 1
    }
    else if (U > (2/7) & (U <= 5/7)) {
      X_cord <- runif(1, 1, 2); Y_cord <- runif(1, 0, 3)
      X <- append(X, X_cord); Y <- append(Y, Y_cord)
      counter <- counter + 1
    }
    else if (U > (5/7) & (U < 1)) {
      X_cord <- runif(1, 2, 3); Y_cord <- runif(1, 1, 3)
      X <- append(X, X_cord); Y <- append(Y, Y_cord)
      counter <- counter + 1
    }
  }
  mylist <- list("X" = X, "Y" = Y, "Proposals" = counter)
  return(mylist)
}

set.seed(135)
Results8 <- RectangleSamp2(2500)

## Scatterplot of Samples
plot(Results8$X, Results8$Y)
```

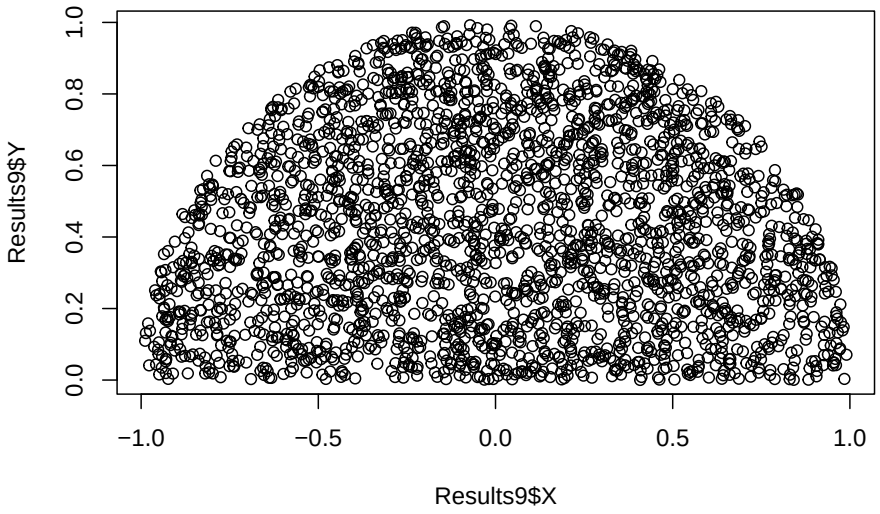

Results8$Y
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Results8$X
0.0
0.5
1.0
1.5
2.0
2.5
3.0

9) [Question E.1.14 in textbook] Propose a rejection method to sample from the uniform distribution on the semicircle:

$$\{(x, y) \in \mathbb{R} | x^2 + y^2 \leq 1, y \geq 0\}$$

Which has an acceptance probability of greater than 80%. Implement your method, make a scatterplot of 2500 sampled values, and calculate your acceptance probability. Note: Using a rejection sampler with $U_n \sim Unif([-1, 1]x[0, 1])$ has an acceptance probability of 78.5%

```
## Sample Uniformly from 2 Rectangles
## Let A = [-1, 1] x [0, .6]; B = [-.8, .8] x [.6, 1]
## Area of A = 2 * .6 = 1.2; Area of B = 1.6 * .4 = .64
## Sample from A = 1.2/1.84 of Time; From B = .64/1.84 of Time
SemiReject <- function(iterations) {
  X <- c(); Y <- c(); counter <- 0
  while (length(X) < iterations) {
    U <- runif(1)
    if (U < 1.2/1.84) {
      U_1 <- runif(1, -1, 1); U_2 <- runif(1, 0, .6)
      if (U_1^2 + U_2^2 <= 1) {
        X <- append(X, U_1); Y <- append(Y, U_2)
        counter <- counter + 1
      }
      else {
        counter <- counter + 1
      }
    }
    else {
      U_1 <- runif(1, -0.8, 0.8); U_2 <- runif(1, .6, 1)
      if (U_1^2 + U_2^2 <= 1) {
        X <- append(X, U_1); Y <- append(Y, U_2)
        counter <- counter + 1
      }
      else {
        counter <- counter + 1
      }
    }
  }
  mylist <- list("X" = X, "Y" = Y, "Proposals" = counter)
  return(mylist)
}

set.seed(140)
Results9 <- SemiReject(2500)

## Scatterplot of Samples
plot(Results9$X, Results9$Y)
```




```
## Acceptance Probability of Samples
2500/Results9$Proposals
```

```
## [1] 0.8471705
```