# Stats 102C HW 3

Charles Liu (304804942)

11/3/2020



**Do not post your solutions online on a site like github. Violations will be reported to the Dean of Students.**

Modify this file with your answers and responses.

## Academic Integrity Statement

By including this statement, I, **Charles Liu**, declare that all of the work in this assignment is my own original work. At no time did I look at the code of other students nor did I search for code solutions online. I understand that plagiarism on any single part of this assignment will result in a 0 for the entire assignment and that I will be referred to the dean of students.

I did discuss ideas related to the homework with Josephine Bruin for parts 2 and 3, with John Wooden for part 2, and with Gene Block for part 5. At no point did I show another student my code, nor did I look at another student's code.

## Reading and Viewing:

- Introducing Monte Carlo Methods with R: Section 2.1, Section 2.2, and Section 2.3
- Kolmogorov-Smirnov Test on Youtube: https://www.youtube.com/watch?v=ZO2RmSkXK3c (This video covers the two-sample test, but we will conduct a one-sample test against a reference distribution)

## Problem 1 - Inverse CDF Exponential Distribution

Write a function `my_rexp(n, rate)`, that will generate `n` random values drawn from an exponential distribution with lambda = "rate" by using the inverse CDF method. Use `runif()` as your sole source of randomness.

You are not allowed to use any of the functions `dexp()`, `pexp()`, `qexp()`, or `rexp()` in your generating function.

Use your function to generate 10,000 random values from an exponential distribution with lambda = 1. **Do not** print out the 10,000 values.

Plot the theoretic CDF of the distribution. Add the empirical CDF of your data to the same plot (in a different color). You can use R's `pexp()` function when plotting the theoretic CDF.

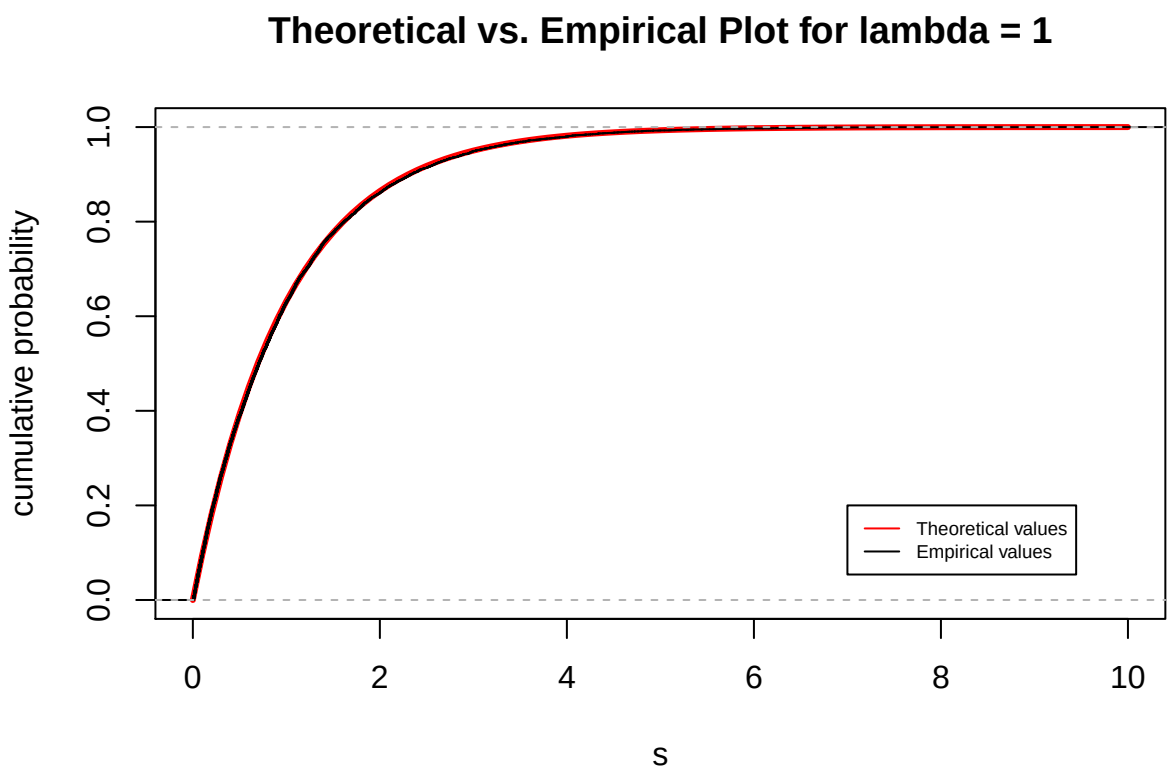
Use the Kolmogorov-Smirnov test to compare your generated samples to the theoretic exponential distribution. Be sure to print out the resulting p-value and comment on the sample produced by your function.

Once you complete the exercise for lambda = 1, repeat the exercise, this time generating 10000 values with lambda = 0.3. Plot the theoretic CDF vs empirical CDF and use the KS test.

```r
# Create my_rexp(...) function
my_rexp <- function(n, rate){
  set.seed(1)
  u <- runif(n) # random uniform
  inverse_exp_cdf <- (-1/rate) * log(u) # inverse CDF of exp(rate = lambda)
}

# Use rate = 1
x1 <- my_rexp(10^4, rate = 1)

# Plot the Theoretical vs. Empirical
s <- seq(0, 10, by = 0.1)
plot(s, pexp(s), type = "l", lwd = 3, ylab = "cumulative probability",
col = "red", main = "Theoretical vs. Empirical Plot for lambda = 1")
plot(ecdf(x1), add = TRUE)
legend(7, 0.2, legend = c("Theoretical values", "Empirical values"),
       col = c("red", "black"),
       lty = c(1, 1), cex = 0.6)
```

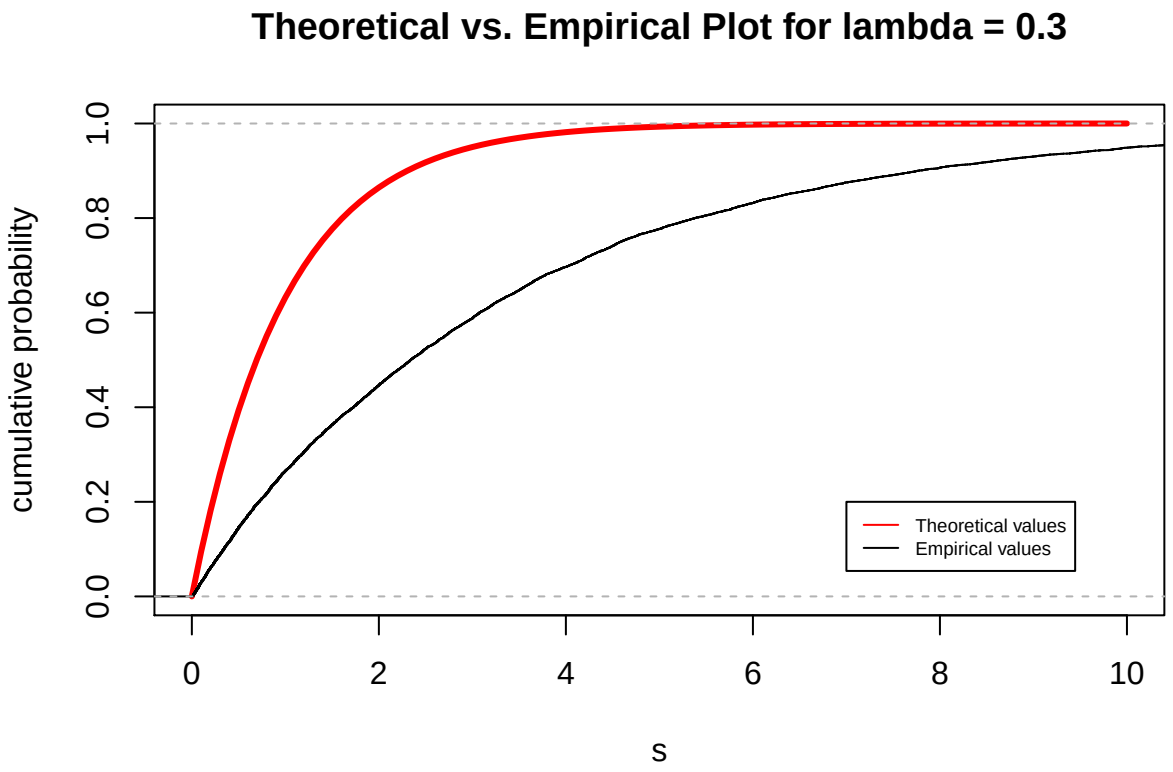



```r
# K-S Test
K1a <- ks.test(x1, pexp, rate = 1)
K1a$p.value
```

```
## [1] 0.3886046
```

```r
# Use rate = 0.3
x2 <- my_rexp(10^4, rate = 0.3)

# Plot the Theoretical vs. Empirical
plot(s, pexp(s), type = "l", lwd = 3, ylab = "cumulative probability",
col = "red", main = "Theoretical vs. Empirical Plot for lambda = 0.3")
plot(ecdf(x2), add = TRUE)
legend(7, 0.2, legend = c("Theoretical values", "Empirical values"),
       col = c("red", "black"),
       lty = c(1, 1), cex = 0.6)
```




```r
# K-S Test
K1b <- ks.test(x2, pexp, rate = 1)
K1b$p.value
```

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

**COMMENTS:** For the first sample ($\lambda = 1$), the p-value of the K-S test is greater than 0.05. THis tells us that we do *NOT* have any reason to doubt that the values came from an Exponential Distribution with $\lambda = 1$. For the second sample ($\lambda = 0.3$), the p-value of the K-S test is less than 0.05. THis tells us that we *DO* have some reason to doubt that the values came from an Exponential Distribution with $\lambda = 0.3$. One important thing to note is that the K-S test does not prove that the values came from the Exponential Distribution, but the K-S test provides no evidence to say otherwise.

## Problem 2 - Inverse CDF - Discrete Case

Write a function `my_rbinom(n, size, prob)`, that will generate `n` random values drawn from a binomial distribution with size = `size` and probability of success = `prob` by using the inverse CDF method. Use `runif()` as your sole source of randomness.

Do not use any of R's binom functions. Do not use `dbinom`, `pbinom`, `qbinom()`, or `rbinom()`

Use your function `my_rbinom()` to generate 10,000 values from a binomial distribution with n = 6, and p = 0.4.

After generating 10,000 samples, make a side-by-side barchart that shows the empirical PMF of your data and the theoretic PMF according to the binomial distribution. (See lecture 3-2, slides 25 and 26)

Use a chi-squared goodness-of-fit test to see if the generated values fit the expected probabilities. Be sure to comment on the graph and results of the test.

Use your function `my_rbinom()` again. This time generate 10,000 values from a binomial distribution with n = 12, and p = 0.55. Make a side-by-side barchart that shows the empirical PMF of your data and the theoretic PMF according to the binomial distribution. No need to run a chi-squared test for this data.

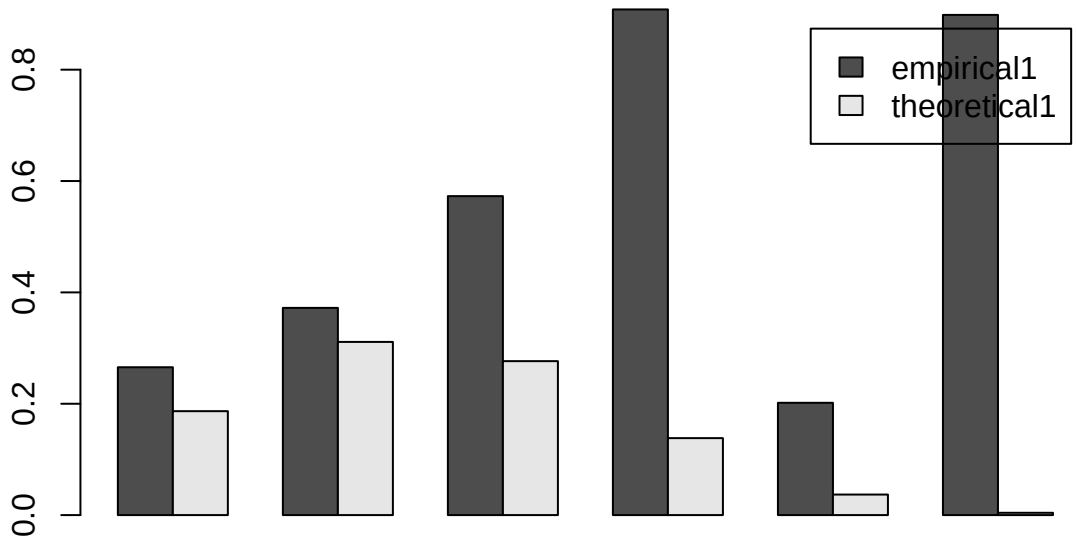
```r
# Parameters set up
set.seed(1)
n <- 10^4

# Create my_rbinom(...) function
my_rbinom <- function(size, prob){
  U <- runif(size)
  X <- rep(1:size)
  position <- 1
  if( U < sum(prob[1:position]) ) {
    position <- position + 1
  }
  return(U)
}

# Sampling
my_samp1 <- my_rbinom(size = 6, prob = 0.4)
samp1 <- replicate(n, my_samp1)
empirical1 <- my_samp1 # empirical probabilities
theoretical1 <- dbinom(1:6, size = 6, prob = 0.4) # theoretic probabilities
comparison1 <- rbind(empirical1, theoretical1)
round(comparison1, 4)
```

```
##                [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## empirical1   0.2655 0.3721 0.5729 0.9082 0.2017 0.8984
## theoretical1 0.1866 0.3110 0.2765 0.1382 0.0369 0.0041
```

```r
barplot(comparison1, beside = TRUE, legend.text = row.names(comparison1))
```

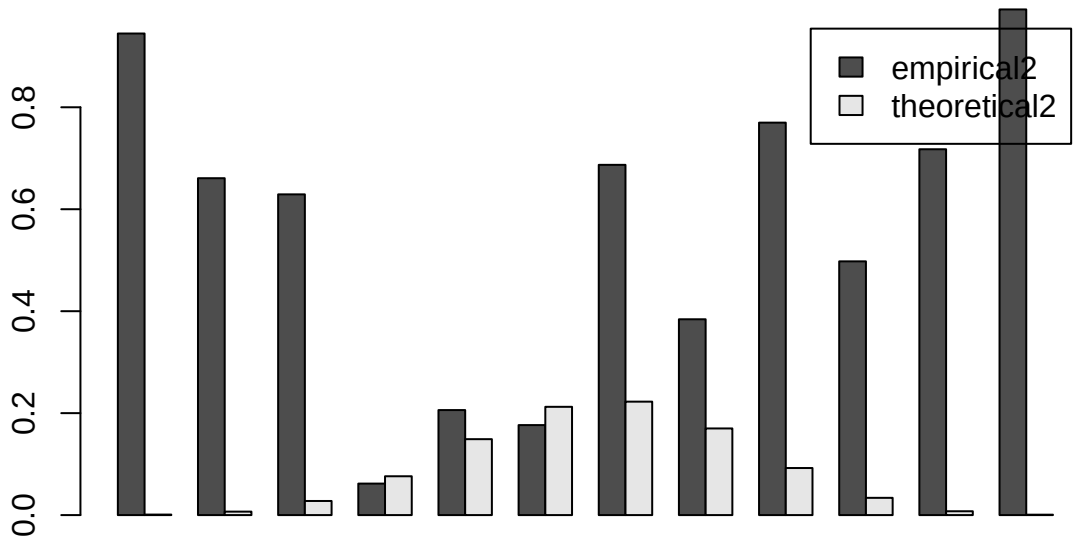



```
# chisq.test(x = table(my_samp), p = theoretical) -> probabilities must sum to 1


# Sampling
my_samp2 <- my_rbinom(size = 12, prob = 0.55)
samp2 <- replicate(n, my_samp2)
empirical2 <- my_samp2 # empirical probabilities
theoretical2 <- dbinom(1:12, size = 12, prob = 0.55) # theoretic probabilities
comparison2 <- rbind(empirical2, theoretical2)
round(comparison2, 4)
```

```
##                [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]
## empirical2   0.9447 0.6608 0.6291 0.0618 0.2060 0.1766 0.6870 0.3841 0.7698
## theoretical2 0.0010 0.0068 0.0277 0.0762 0.1489 0.2124 0.2225 0.1700 0.0923
##               [,10]  [,11]  [,12]
## empirical2   0.4977 0.7176 0.9919
## theoretical2 0.0339 0.0075 0.0008
```

```
barplot(comparison2, beside = TRUE, legend.text = row.names(comparison2))
```




## Problem 3 - RNG based on inverse CDF and convolutions

Using only `runif()` and/or `rnorm()` as sources of randomness, generate 10,000 ($10^4$) random samples from each of the following distributions. You are not allowed to use any of R's other distribution functions for the generation of random values. **Do NOT** print out the actual values in your random sample.

For each distribution:

- After generating your 10000 samples, plot a density histogram of the resulting sample (`breaks = 30, freq = FALSE`). Plot the theoretic density in another color on top of the histogram. Comment on the plot. You are allowed use R's density functions `dchisq()`, `dt()`, etc. when plotting the density function over the histogram
- Plot the theoretic CDF of the distribution. Add the empirical CDF of your data to the same plot (in a different color). You can use R's CDF functions `pchisq()`, `pt()`, etc. when plotting the CDF.
- Use the Kolmogorov-Smirnov test to compare your generated samples to the theoretic distributions. Be sure to print out the resulting p-value and comment on the sample produced by your function.

### Problem 3a:

- Beta distribution with shape parameters 4 and 2

```
# Set up parameters
n <- 10^4
a <- 4
```

```r
b <- 2

# Create the my_rbeta(...) function
my_rbeta <- function(a, b){
  U_a <- runif(a)
  X_a <- rep(0, a)
  U_b <- runif(b)
  X_b <- rep(0, b)
  inverse_exp_cdf_a <- (-1/1) * log(U_a) # rate  = 1
  inverse_exp_cdf_b <- (-1/1) * log(U_b) #  rate = 1
  X_a[U_a < inverse_exp_cdf_a] <- 1
  X_b[U_b < inverse_exp_cdf_b] <- 1
  Y <- sum(X_a)/sum(X_b)
  return(Y)
}


# my_samp3a <- my_rbeta(a = a, b = b)
# samp3a <- replicate(n, my_samp3a)

# empirical3a <- my_samp3a # empirical probabilities
# theoretical3a <- dbeta(1:n, shape1 = a, shape2 = b) # theoretic probabilities
# comparison3a <- rbind(empirical3a, theoretical3a)
# hist(comparison3a, breaks = 30, freq = FALSE)
```

**Problem 3b:**

- Chi-squared distribution with 4 degrees of freedom

```r
my_rchisq <- function(df){
  U <- rnorm(n)
}
```

**Problem 3c:**

- t-distribution with 4 degrees of freedom

```r
my_rt <- function(n, df){
  U <- rnorm(n)
}
```

**Problem 3d:**

- Gamma distribution with shape parameter 4 and rate parameter 2.

```r
my_rgamma <- function(alpha, beta){
  U_alpha <- runif(alpha)
  X_alpha <- rep(0, alpha)
  U_beta <- runif(beta)
  X_beta <- rep(0, beta)
```

```
  inverse_exp_cdf_alpha <- (-1/1) * log(U_alpha) # rate  = 1
  inverse_exp_cdf_beta <- (-1/1) * log(U_beta) #  rate = 1
  X_alpha[U_alpha < inverse_exp_cdf_alpha] <- 1
  X_beta[U_beta < inverse_exp_cdf_beta] <- 1
  Y <- sum(X_alpha)/sum(X_beta)
  return(Y)
}
```

**NOTE:** I couldn't have enough time to finish problem 3 due to me focusing more on my Kaggle Midterm for Stats 101C. It has been taking me all last week to do it. Sorry for not being able to finish these problems, but there was an attempt made on them.

## Problem 4 - Rejection Sampling

Let $f(x)$ and $g(x)$ be the target and candidate (proposal) distributions, respectively, in acceptance-rejection sampling.

$f(x) = \frac{1}{2}\sin(x)$ for $0 \leq x \leq \pi$

$g(x) = \text{Unif}(0, \pi)$

Find the optimal constant $M = \max \frac{f(x)}{g(x)}$.

Implement the rejection sampling design, using `runif(n, 0, pi)` as your source of randomness. Generate a sample of at least 10,000 accepted values.

```
set.seed(1)
f4 <- function(x){ (1/2) * sin(x) }
g4 <- function(x){ rep( (1/pi) , length(x)) } # Uniform PDF is [ 1/(b - a) ]
N <- 10^4
M <- pi # optimal constant (M)
proposed_x4 <- runif(N, 0, pi) # proposal distribution is Unif(0, pi)
r_x4 <- f4(proposed_x4)/(M*g4(proposed_x4))
U4 <- runif(N)
accepted4 <- U4 < r_x4
accepted_x4 <- proposed_x4[accepted4]
```

**ANSWER:** Our optimal constant is $M = \max \frac{f(x)}{g(x)} = \pi$ . This comes from $\frac{d}{dx}(\frac{0.5*sin(x)}{\frac{1}{(\pi-0)}}) = \frac{\pi}{2}cos(x) = 0$

What is your empirical acceptance rate?

Create a histogram of your generated (accepted) sample `(breaks = 30, freq = FALSE)`. Add the theoretic PDF to the histogram.

Plot the theoretic CDF of the distribution. Add the empirical CDF of your data to the same plot (in a different color).

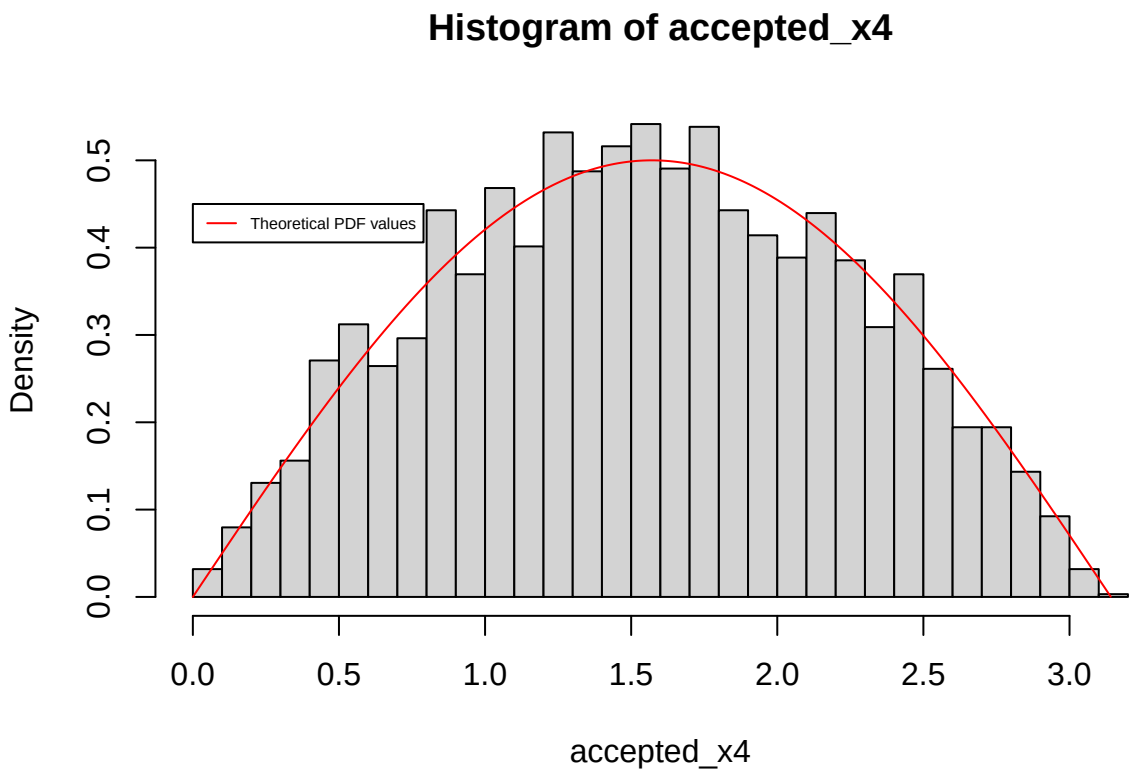
Use the Kolmogorov-Smirnov test to compare your generated samples to the theoretic distributions. Be sure to print out the resulting p-value and comment on the sample produced by your function.

I have written a vectorized PDF and CDF function for you.

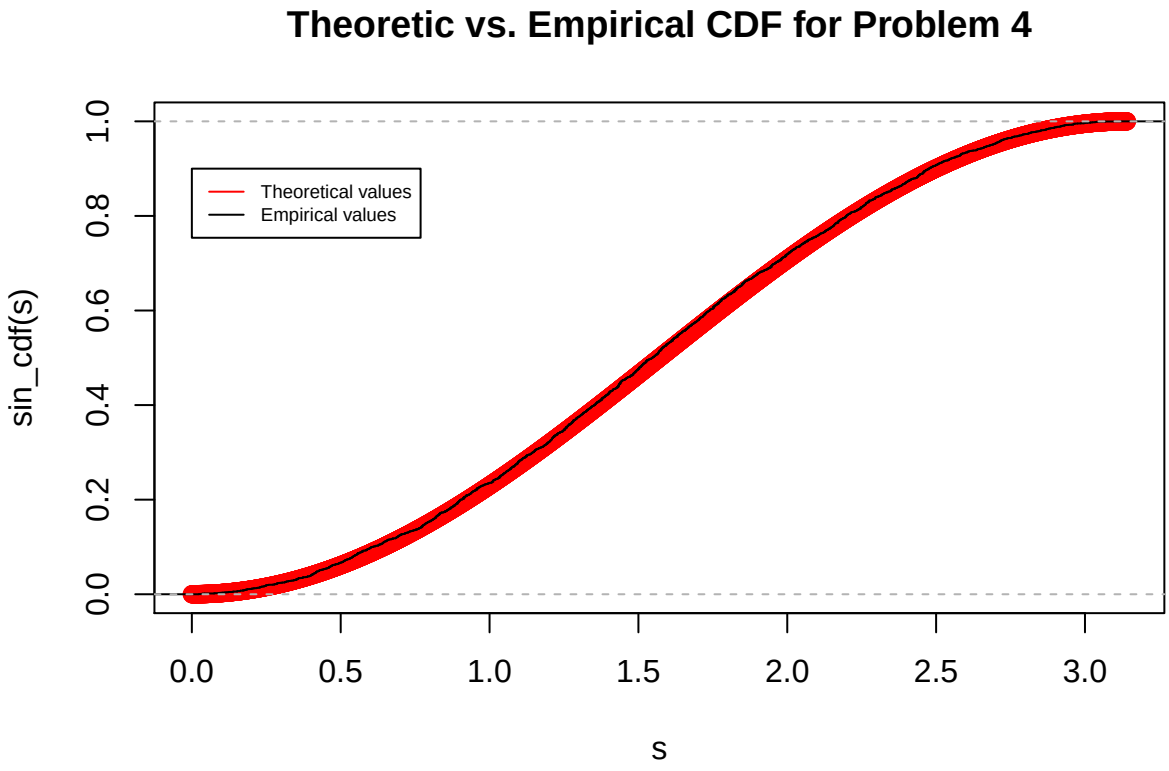
```
sin_pdf <- function(x) {
  ifelse(x > 0 & x < pi, 0.5 * sin(x), 0)
}
```

```r
sin_cdf <- function(x) {
  ifelse(x < 0, 0, ifelse(x < pi, 0.5 - 0.5 * cos(x), 1))
}
```

```r
# Create the Histogram w/ accepted samples and Theoretic PDF
s <- seq(0, pi, by = 0.001)
hist(accepted_x4, breaks = 30, freq = FALSE)
lines(s, sin_pdf(s), col = "red")
legend(0, 0.45,
       legend = "Theoretical PDF values",
       col = "red",
       lty = 1,
       cex = 0.5)
```




```r
# Plotting the Theoretic CDF & Empirical CDF
plot(s, sin_cdf(s), col = "red", lwd = 2, main = "Theoretic vs. Empirical CDF for Problem 4")
plot(ecdf(accepted_x4), add = TRUE)
legend(0, 0.9, legend = c("Theoretical values", "Empirical values"),
       col = c("red", "black"),
       lty = c(1, 1), cex = 0.6)
```




```
# K-S Test
K4 <- ks.test(accepted_x4, sin_cdf)
K4$p.value
```

```
## [1] 0.1635349
```

**COMMENTS:** For the the sample, the p-value of the K-S test is greater than 0.05. THis tells us that we do *NOT* have any reason to doubt that the values came from the distribution stated above with parameters $0 \leq x \leq \pi$.

## Problem 5 - Rejection Sampling

Use rejection sampling to generate samples from the Beta distribution with shape parameters $a = 4$ and $b = 6$.

The PDF of this Beta distribution is:

$$f(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} = 105x^3(1-x)^5$$

Use the Uniform (0,1) distribution as your trial distribution.

Use calculus to solve for $M = \max \frac{f(x)}{g(x)}$. Show your work. (The derivative is easy to find and can be easily factored to find the roots.)

Implement the rejection sampling design, using `runif(n)` as your source of randomness. Generate a sample of at least 10,000 accepted values.

Use the Kolmogorov-Smirnov test to compare your generated samples to the theoretic distributions. You may use `pbeta` for the CDF. Be sure to print out the resulting p-value and comment on the sample produced by your function. (No additional plots necessary)

```
set.seed(1)
f5 <- function(x){ 105 * (x^3) * ((1-x)^5) }
g5 <- function(x){ rep(1, length(x)) } # Uniform PDF is [ 1/(b - a) ]
N <- 10^4
M <- 1 # optimal constant (M) (work shown below)
proposed_x5 <- runif(N, 0, 1) # proposal distribution is Unif(0, pi)
r_x5 <- f5(proposed_x5)/(M*g5(proposed_x5))
U5 <- runif(N)
accepted5 <- U5 < r_x5
accepted_x5 <- proposed_x5[accepted5]

K5 <- ks.test(accepted_x5, pbeta(N, shape1 = 4, shape2 = 6))
K5$p.value
```

```
## [1] 0.0009647853
```

**ANSWER:** For showing the work for solving for the M, it is shown below... $\frac{d}{dx}[105x^3(1-x)^5] = 0$, and then we apply the Product Rule for Derivatives $f' * g + f * g'$. We have $f = x^3$, $f' = 3x^2$, $g = (1-x)^5$ and $g' = -5(1-x)^4$ for our Product Rule. We then implement all together to get $105(3x^2(1-x)^5 - 5x^3(1-x)^4 = 0$. Finally, we see that the optimal constant (M) is $M = 1$ from the choices of $x = 0, \frac{3}{8}, 1$.

**COMMENTS:** For the the sample, the p-value of the K-S test is less than 0.05. THis tells us that we *DO* have some reason to doubt that the values came from an Uniform Distribution with parameters $Unif(0, \pi)$. One important thing to note is that the K-S test does not prove that the values came from the following distribution, but the K-S test provides no evidence to say otherwise.

## Problem 6 - Empirical Supremum Rejection Sampling

One challenge of rejection sampling is finding the constant $M = \max \frac{f(x)}{g(x)}$. Empirical Supremum rejection sampling estimates the quantity $M$ with a value $\hat{c}$. The algorithm works much in the same was as rejection sampling, but it continually updates the value $\hat{c}$ if a new $x$ is produced where $\frac{f(x)}{g(x)}$ is larger than the current estimate $\hat{c}$.

Read: 6.3.3 Empirical Supremum Rejection Sampling from the following website:

https://bookdown.org/rdpeng/advstatcomp/rejection-sampling.html#empirical-supremum-rejection-sampling

(side note: Roger Peng earned his PhD in Statistics from UCLA and hosts the data-science podcast "Not so standard deviations" with Hilary Parker.)

Use Empirical Supremum Rejection Sampling to generate samples from the normal distribution.

The target distribution f(x) will be the positive half of the standard normal distribution, which will have PDF:

$$f(x) = 2 \times \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right), \text{ for } x \geq 0$$

Use an exponential distribution with lambda = 1 as your trial (proposal) distribution.

$$g(x) = e^{-x}, \text{ for } x \geq 0$$

Unlike the example from lecture, DO NOT find the optimal constant M that will maximize the acceptance rates for the rejection sampling design.

Implement Empirical Supremum Rejection Sampling. While the webpage describes a process that looks like it should be implemented with a loop, it is possible to achieve the same result in a more efficient manner without the need for any loops:

- Propose $n = 10000$ values. The accepted sample will be smaller.
- Use `runif` and inverse CDF to generate $n$ proposal values $X$ from the exponential distribution.
- Calculate the ratio for all $n$ values: $\frac{f(X)}{g(X)}$
- Estimate $\hat{c}$ as the maximum $\frac{f(X)}{g(X)}$ you encounter.
- Use `runif` to generate $n$ values of $U$ to decide whether to accept or reject the proposed $X$.
- reject all proposed X values that do not meet the rejection criteria.

Once you have generated samples from the folded normal distribution using rejection sampling, turn the accepted values into values from the standard normal distribution. Use `runif` to generate $S$ to decide the sign of the accepted $X$. The accepted $X$ values will be positive or negative with probably 0.5.

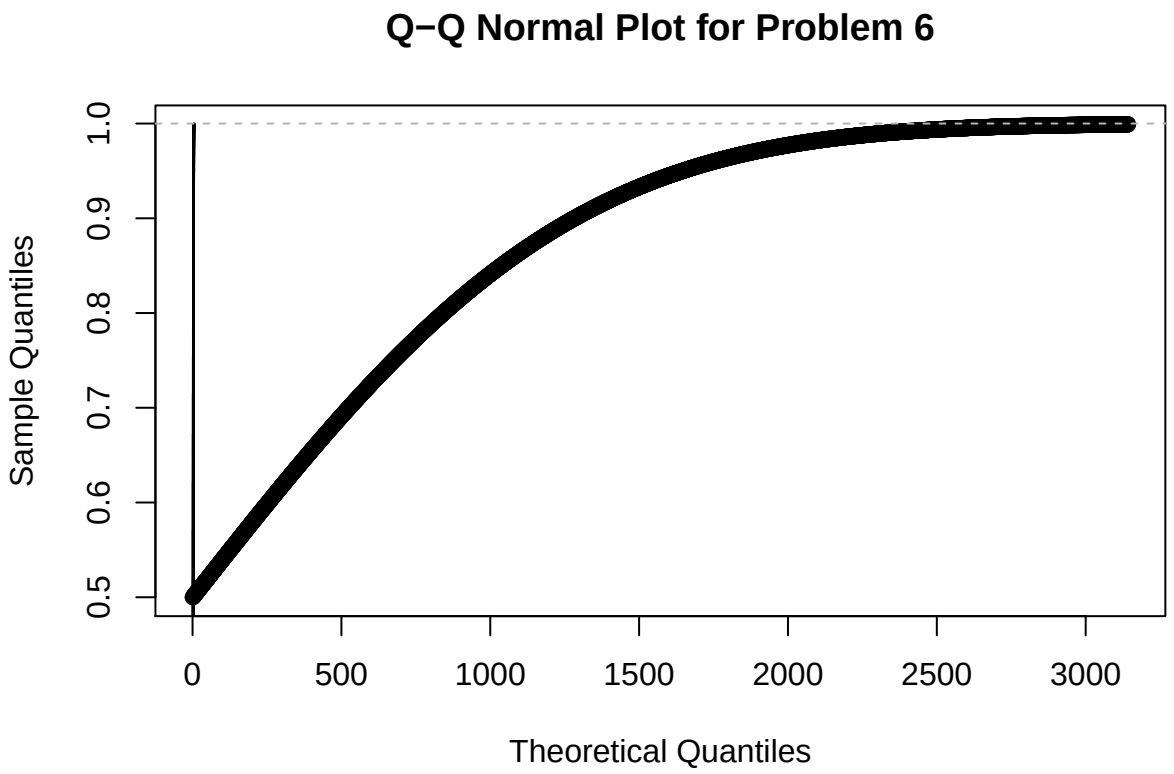
Create a QQ-norm plot of your accepted sample or normally distributed values. Comment on the plot.

Perform a Shapiro-Wilk test `shapiro.test()` to test normality. Comment on the results.

```
# Create my_rexp(...) function
my_rexp <- function(n, rate){
  set.seed(1)
  u <- runif(n) # random uniform
  inverse_exp_cdf <- (-1/rate) * log(u) # inverse CDF of exp(rate = lambda)
}

n <- 10^4
f6 <- function(x){ (2/sqrt(2*pi)) * exp((-x^2)/2) } # for x >= 0
g6 <- function(x){ exp(-x) } # for x >= 0
# using Problem 1's function
proposed_x6 <- my_rexp(n, rate = 1)
r_x6a <- f6(proposed_x6)/(g6(proposed_x6))
c_hat <- max(r_x6a)
r_x6b <- f6(proposed_x6)/(c_hat*g6(proposed_x6))
U6 <- runif(n, 0, 1)
accepted6 <- U6 < r_x6b
accepted_x6 <- proposed_x6[accepted6]
S6 <- runif(n)
rejected6 <- S6 > r_x6b
rejected_x6 <- proposed_x6[rejected6] # these are our negatives

# QQ Norm Plot
s <- seq(0, pi, by = 0.001)
plot(pnorm(s), main = "Q-Q Normal Plot for Problem 6", xlab = "Theoretical Quantiles", ylab = "Sample Q
plot(ecdf(accepted_x6), add = TRUE)
```




```
# Shapiro Test
shapiro.test(accepted_x6[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  accepted_x6[1:5000]
## W = 0.92301, p-value < 2.2e-16
```

**COMMENTS:** We can see that our $\hat{c}$ does help give a good estimate in maximizing our data. Our plot seems to follow quite well. As for our Shapiro Test, we can see that the p-value is less than 0.05, so there is room for doubt in the distribution and sampling.

## Problem 7 - Bivariate Normal Distribution

Generate 1000 random observations from a bivariate normal distribution.

$$\mathbf{X} \sim \mathcal{N}_2\left(\boldsymbol{\mu} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} 3 & -1.5 \\ -1.5 & 3 \end{pmatrix}\right)$$

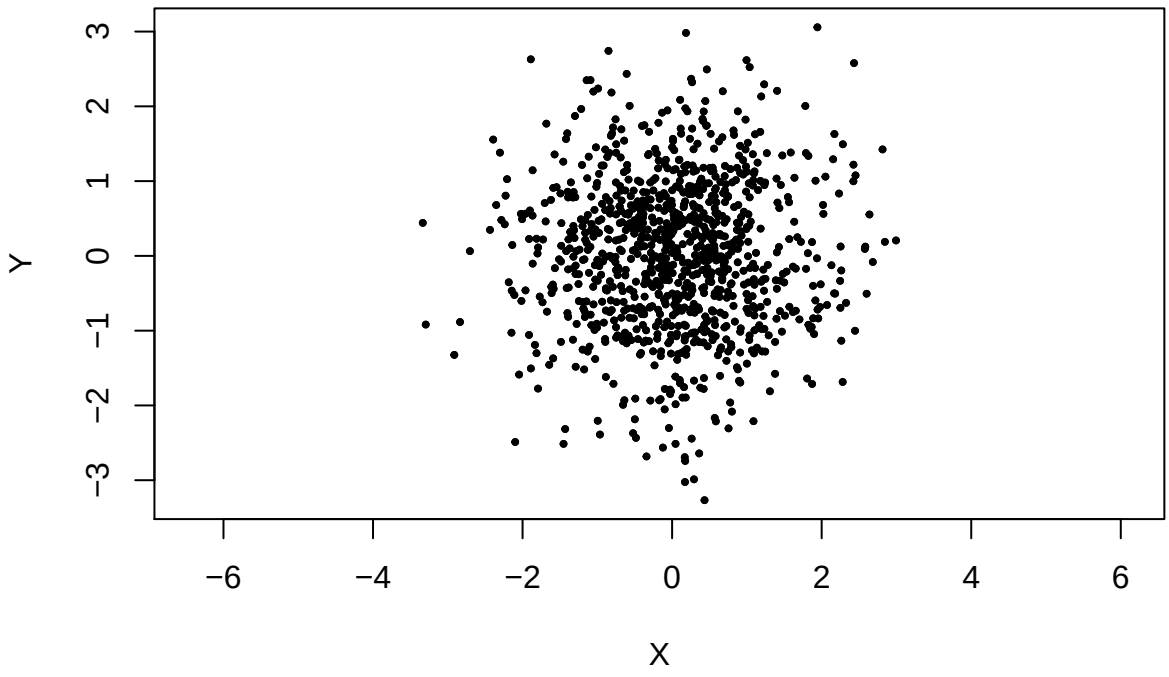
Implement the Box-Muller transform to generate standard normal values. Use `runif()` as your only source of randomness.

Once you have standard normal values, apply the necessary transform to get the desired bivariate distribution. You may use `chol` to find the Cholesky decomposition of a matrix.

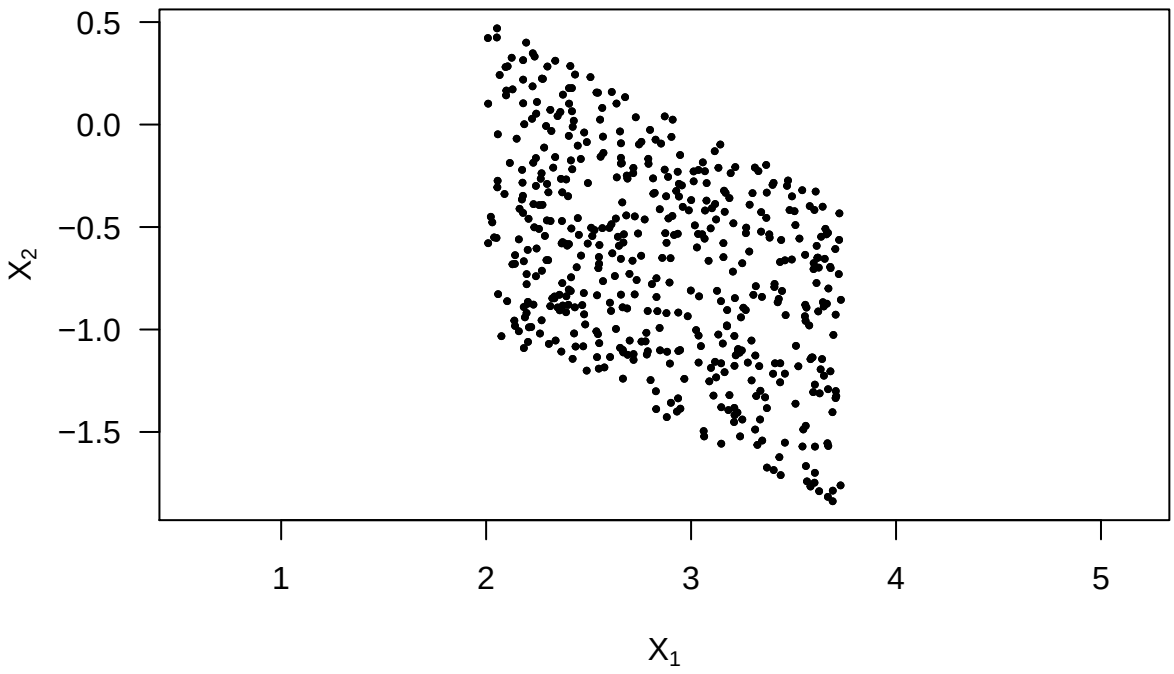
Create a plot the resulting generated data.

```
n <- 1000
U <- matrix(runif(n,0,1), nrow = 2, ncol = n)
mu <- c(2, -1)
Sigma <- cbind(c(3,-1.5), c(-1.5,3))
A <- t(chol(Sigma))

# Box-Muller
Theta <- 2*pi*U
V <- matrix(runif(n,0,1), nrow = 2, ncol = n)
R <- sqrt(-2*log(V))
X <- R*cos(Theta)
Y <- R*sin(Theta)
plot(X, Y, cex = 0.4, asp = 1, pch = 19)
```




```
# Desired Bivariate Distribution
X_prime <- mu + A %*% U
plot(X_prime[1,], X_prime[2,], cex = 0.4,
     asp = 1, pch = 19, las = 1,
     xlab = expression(X[1]), ylab = expression(X[2]))
```


0.5
0.0
−0.5
−1.0
−1.5
X2
1
2
3
4
5
X1