STAT GU4206/GR5206 Homework 5 [150 pts]
Due 11:59pm Monday, November 25th on Canvas

Your homework should be submitted on Canvas as an `R` Markdown file. **Please submit the knitted .pdf file (or .html)** along with the .Rmd file. Please clearly label the questions in your responses and support your answers by textual explanations and the code you use to produce the result. Please do not print the dataset or any vector over, say, length 20.

Goals: Simulating probability distributions using the Inverse Transform Method and Accept-Reject Method. Built in `R` distribution functions. Estimate the mathematical constant π using Monte Carlo techniques. More practice writing functions and plotting.

First set your seed as 0, i.e., `set.seed(0)`

Part 1: Inverse Transform Method

Consider the *Cauchy random variable* X with probability density function

$$f_X(x) = \frac{1}{\pi}\frac{1}{(1+x^2)}, \quad -\infty < x < \infty.$$

1. Let U be a uniform random variable over [0,1]. Find a transformation of U that allows you to simulate X from U.
2. Write a `R` function called `cauchy.sim` that generates n simulated Cauchy random variables. The function should have the single input `n` and should use the inverse-transformation from Part 1. Test your function using 10 draws.
3. Using your function `cauchy.sim`, simulate 1000 random draws from a Cauchy distribution. Store the 1000 draws in the vector `cauchy.draws`. Construct a histogram of the simulated Cauchy random variable with $f_X(x)$ overlaid on the graph. **Note:** when plotting the density curve over the histogram, include the argument `prob = T`. **Also note:** the Cauchy distribution produces extreme outliers. I recommend plotting the histogram over the interval $(-10, 10)$.

Part 2: Reject-Accept Method

Problem 2

Let random variable X denote the temperature at which a certain chemical reaction takes place. Suppose that X has probability density function

$$f(x) = \begin{cases} \frac{1}{9}(4-x^2) & -1 \le x \le 2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Perform the following tasks:

4. Write a function `f` that takes as input a vector `x` and returns a vector of `f(x)` values. Plot the function between -3 and 3. Make sure your plot is labeled appropriately.
5. Determine the maximum of $f(x)$ and find an envelope function $e(x)$ by using a uniform density for $g(x)$. Write a function `e` which takes as input a vector `x` and returns a vector of `e(x)` values.
6. Using the **Accept-Reject Algorithm**, write a program that simulates 10,000 draws from the probability density function $f(x)$ from Equation 1. Store your draws in the vector `f.draws`.
7. Plot a histogram of your simulated data with the density function `f` overlaid in the graph. Label your plot appropriately.

Problem 3: Reject-Accept Method Continued

Consider the standard normal distribution X with probability density function

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\big(-\frac{1}{2}x^2\big), \quad -\infty < x < \infty.$$

In this exercise, we will write a function named `normal.sim` that simulates a standard normal random variable using the **Accept-Reject Algorithm**.

Perform the following tasks:

8. Write a function `f` that takes as input a vector `x` and returns a vector of `f(x)` values. Plot the function between -5 and 5. Make sure your plot is labeled appropriately.
9. Let the known density g be the Cauchy density defined by pdf

   $$g(x) = \frac{1}{\pi}\frac{1}{(1+x^2)}, \quad -\infty < x < \infty.$$

   Write a function `e` that takes as input a vector `x` and constant `alpha` $(0 < \alpha < 1)$ and returns a vector of `e(x)` values. The envelope function should be defined as $e(x) = g(x)/\alpha$.
10. Determine a "good" value of α. You can solve this problem graphically. To show your solution, plot both $f(x)$ and $e(x)$ on the interval $[-10, 10]$.
11. Write a function named `normal.sim` that simulates `n` standard normal random variables using the **Accept-Reject Algorithm**. The function should also use the **Inverse-Transformation** from Part 1. Test your function using `n=10` draws.
12. Using your function `normal.sim`, simulate 10,000 random draws from a standard normal distribution. Store the 10,000 draws in the vector `normal.draws`. Construct

a histogram of the simulated standard normal random variable with $f(x)$ overlaid on the graph. **Note:** when plotting the density curve over the histogram, include the argument `prob = T`.

Part 3: Simulation with Built-in `R` Functions

Consider the following "random walk" procedure:

- Start with $x = 5$
- Draw a random number r uniformly between -2 and 1.
- Replace x with $x + r$
- Stop if $x \leq 0$
- Else repeat

Perform the following tasks:

13. Write a `while()` loop to implement this procedure. Importantly, save all the positive values of x that were visited in this procedure in a vector called `x.vals`, and display its entries.

14. Produce a plot of the random walk values `x.vals` from above versus the iteration number. Make sure the plot has an appropriately labeled x-axis and and y-axis. Also use `type="o"` so that we can see both points and lines.

15. Write a function `random.walk()` to perform the random walk procedure that you implemented in question (9). Its inputs should be: `x.start`, a numeric value at which we will start the random walk, which takes a default value of 5; and `plot.walk`, a boolean value, indicating whether or not we want to produce a plot of the random walk values `x.vals` versus the iteration number as a side effect, which takes a default value of `TRUE`. The output of your function should be a list with elements: `x.vals`, a vector of the random walk values as computed above; and `num.steps`, the number of steps taken by the random walk before terminating. Run your function twice with the default inputs, and then twice times with `x.start` equal to 10 and `plot.walk = FALSE`.

16. We'd like to answer the following question using simulation: if we start our random walk process, as defined above, at $x = 5$, what is the expected number of iterations we need until it terminates? To estimate the solution produce $10,000$ such random walks and calculate the average number of iterations in the $10,000$ random walks you produce. You'll want to turn the plot off here.

17. Modify your function `random.walk()` defined previously so that it takes an additional argument `seed`: this is an integer that should be used to set the seed of the random number generator, before the random walk begins, with `set.seed()`. But, if seed is `NULL`, the default, then no seed should be set. Run your modified function `random.walk()` function twice with the default inputs, then run it twice with the input seed equal to (say) 33 and `plot.walk = FALSE`.

Part 4: Monte Carlo Integration

Consider the integral

$$\int_0^1 g(x)dx = \int_0^1 e^{-x^3}dx.$$

Perform the following tasks:

18. Run the following code:

```
g <- function(x) {
  return(exp(-x^3))
}
x <- seq(0,1,.01)
alpha <- 2
beta <- 2
plot(x,g(x),type="l",xlab="x",ylab="",ylim=c(-.1,1.4))
polygon(c(0,seq(0,1,0.01),1),c(0,g(seq(0,1,0.01)),0) ,col="pink")
lines(x,rep(1,length(x)),col="red")
lines(x,dbeta(x,shape1=alpha,shape2=beta),col="blue")
legend("topleft",legend=c("g(x)","uniform","beta(2,2)"),
    lty=c(1,1,1),col=c("black","red","blue"),cex=.6)
```

19. Using **Monte Carlo Integration**, approximate the integral $\int_0^1 e^{-x^3}dx$ using $n = 1000^2$ random draws from the distribution uniform(0,1).

20. Using **Monte Carlo Integration**, approximate the integral $\int_0^1 e^{-x^3}dx$ using $n = 1000^2$ random draws from the distribution beta($\alpha = 2$, $\beta = 2$).