STAT S4206/5206 Homework 5 [100 pts]
Due 11:59pm Friday, June 23rd on Canvas

Your homework should be submitted on Canvas as an `R` Markdown file. **Please submit the knitted .pdf file** 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 the Accept-Reject Method. Built in `R` distribution functions. Estimate the mathematical constant π using Monte Carlo techniques. More practice writing functions and plotting.

Part 1: Inverse Transform Method

We continue working with the World Top Incomes Database, and the Pareto distribution, as in Lab 5. Recall that for most countries in most time periods, the upper end of the income distribution roughly follows a Pareto distribution, with probability density function

$$f(x) = \frac{(a-1)}{x_{min}} \left(\frac{x}{x_{min}} \right)^{-a} \tag{1}$$

for incomes $x \geq x_{min}$. In Lab 5 , we estimated the parameter a based on the `wtid-report` dataset for years ranging from 1913 to 2015.

Now suppose that we are interested in simulating the upper end of income just for 2015 using the Pareto distribution, (1). Let the 'upper end' begin at the 99th annual income percentile for 2015 (meaning, let $x_{min} = \$407,760$) and we'll estimate the Pareto exponent using $\hat{a} = 2.654$.

Perform the following tasks:

1. Define a function `f` which takes three inputs x, a vector, and scalars a and x_{min} having default values of $a = \hat{a}$ and $x_{min} = \$407,760$. The function should output $f(x)$ for a given input $x > x_{min}$. Plot the function between x_{min} and $1,000,000$. Make sure your plot is labeled appropriately.

2. For $x > x_{min}$, the cdf equals

$$F(x) = 1 - \left(\frac{x}{x_{min}} \right)^{-a+1}$$

   Find the inverse function $F^{-1}(u)$ and define a function `upper.income`. The function should have three inputs u, a vector, and scalars a and x_{min} taking default values of $a = \hat{a}$ and $x_{min} = \$407,760$. The function should output $F^{-1}(u)$ for a given input $u \in (0,1)$. Make sure `upper.income(.5)` returns 620020.2.

3. Using the **Inverse Transform Method**, simulate 1000 draws from the Pareto distribution (1) and plot a histogram of your values. Overlay the simulated distribution with the Pareto density (1). Make sure to label the histogram appropriately.

4. Using your simulated set, estimate the median income for the richest 1% of the world. Recall from lab that the proportion of people whose income is at least x_{min} whose income is also at or above any level $w \geq x_{min}$ is

$$\mathbf{Pr}(X \geq w) = \left(\frac{w}{x_{min}}\right)^{-a+1}.$$

Compare your estimated 50th percentile to the actual 50th percentile of the Pareto distribution.

Part 2: Reject-Accept Method

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

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

Perform the following tasks:

5. 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.

6. 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.

7. Using the **Accept-Reject Algorithm**, write a program that simulates 1000 draws from the probability density function $f(x)$ from Equation 2.

8. Plot a histogram of your simulated data with the density function `f` overlaid in the graph. Label your plot appropriately.

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:

9. 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.

10. 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.

11. 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`.

12. 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.

13. 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

The goal of this exercise is to estimate the mathematical constant π using **Monte Carlo Integration**. Consider the function

$$g(x) = \begin{cases} \sqrt{1-x^2} & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The above function traces out a quartile circle over the interval $[0, 1]$.

Perform the following tasks:

14. Run the following code:

```
g <- function(x) {
  return(sqrt(1-x^2))
}
library(ggplot2)
ggplot() +
  geom_line(aes(x=seq(0,1,.01),y=g(seq(0,1,.01))))+
  labs(x="x",y="g(x)")
```

    The above code should produce the plot of a quarter circle.

15. Identify the true area under the curve $g(x)$ by using simple geometric formulas.

16. Using **Monte Carlo Integration**, approximate the mathematical constant π within a 1/1000 of the true value. When performing this simulation, make sure to choose a probability density function that has support over the unit interval, i.e. uniform(0,1) or beta(α,β).

Please submit the knitted .pdf file!