Statistics 133 -Writing Functions in R

1. Write a function that takes a number `year` and returns a logical value which is either `TRUE`, if the year was or will be a leap year, or `FALSE` if not. The rule for determining leap years is that one of the following conditions must hold:

   1) the year is divisible by 4 but not divisible by 100, or

   2) the year is divisible by 400.

   Now write an `R` expression using your function to calculate how many leap years you have lived through during your lifetime.

2. It is possible to approximate the area of any two-dimensional shape using a *Monte Carlo* technique. To do this, we specify a region of known area that contains the shape, sample uniformly on that region, then multiply the known area by the fraction of the points that fall within the shape we're trying to measure. In this problem you'll write code to carry this out for a circle with radius equal to one. We can compare this to what we already know about the area of circles to see how well our approximation method performs.

   (a) Write expressions in `R` to do the following:

      - Create two vectors `x` and `y` of length 100, where each is sampled from the Uniform distribution with lower limit -1 and upper limit 1.
      - Calculate the proportion of these for which the point (x, y) lies within the unit circle.
      - Make a scatter plot of `x` and `y`, coloring the points differently depending on whether they're inside the unit circle or not.
      - Multiply the proportion by 4 (the area of the square over which we're sampling) to approximate the area inside the circle.

   (b) Now copy your code from part (a) and use it to create a function that takes arguments `n` (for sample size) and `plotit` (to indicate whether or not to make the plot) and returns the approximated area and optionally makes a plot depending on whether `plotit` is `TRUE` or `FALSE`. Give each argument a default value of your choosing.

   (c) The `replicate` function takes an `R` expression and evaluates it a specified number of times. You can think of it as a very simplified `for` loop in which exactly the same code gets evaluated each time through the loop. Use this function to approximate the area of the circle 100 times, first for `n=50` and then for `n=500` and storing the results in two different vectors.

(d) Make two histograms (facetted by vector size) showing the results from part (c). Add a vertical line to the plot to indicate the true area. Give your plot appropriate axis names and titles and a nice readable theme.

3. Functions can be passed as arguments to other functions. Here's an example that we will explore. (This problem uses a `while` loop, which we will cover in class, however you don't need to know how to write a `while` loop yourself to do this problem.)

   Newton's method is a popular numerical method to find a root of an algebraic equation $f(x) = 0$. If $f(x)$ has a derivative $f'(x)$, then the following iteration will converge to a root of the above equation if started close enough to the root:

$$\begin{aligned} x_1 &= \text{initial guess} \\ x_n &= x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \end{aligned}$$

   This idea is based on the Taylor approximation

$$f(x_n) \approx f(x_{n-1}) + (x_n - x_{n-1})f'(x_{n-1})$$

   Suppose $f(x) = x^3 + 2x^2 - 7$, so that $f'(x) = 3x^2 + 4x$. The file `Newton.R` contains some `R` code to implement Newton's method for this problem. You may adapt this code as needed to do the following problems.

   (a) Write a function that implements Newton's method for this particular choice of function f. It should have arguments for the initial guess and the tolerance for convergence (currently set to 0.00001 in my code). Pay special attention to whether either of these should have default values. The function should return a single number and not plot anything.

   (b) Write two `R` functions for f and f'. Each should take argument `x`.

   (c) Rewrite your function from (a) so that it also takes arguments for the function whose root you want to find and its first derivative. You may find it helpful to first use your functions from (b) and make sure the function still works, then modify the arguments to the function.

   (d) Choose a different function f and write `R` functions for it and the corresponding derivative. Make a plot as I did in `Newton.R` to verify that the algorithm found a root of your function. (*Note: There are some cases in which Newton's method will fail to converge, which you will experience as the* `while` *loop never ending. If that happens to you, just choose another initial guess and/or function.*)