Programming with R Test #1

Instructions

R markdown is a plain-text file format for integrating text and R code, and creating transparent, reproducible and interactive reports. An R markdown file (.Rmd) contains metadata, markdown and R code "chunks", and can be "knit" into numerous output types. Answer the test questions by adding R code to the fenced code areas below each item. Once completed, you will "knit" and submit the resulting .html file, as well the .Rmd file. The .html will include your R code *and* the output. The .html file will be graded and returned with comments as a .pdf document.

Before proceeding, look to the top of the .Rmd for the (YAML) metadata block, where the *title* and *output* are given. Please change *title* from 'Programming with R Test #1' to your name, with the format 'lastName_firstName.'

If you encounter issues with knitting the .html, please send an email via Canvas to your TA.

Each code chunk is delineated by six (6) backticks; three (3) at the start and three (3) at the end. After the opening ticks, arguments are passed to the code chunk and in curly brackets. **Please do not add or remove backticks, or modify the arguments or values inside the curly brackets**. An example code chunk is included here:

```
# Comments are included in each code chunk, simply as prompts

...R code placed here

...R code placed here
```

You need only enter text inside the code chunks for each test item.

Depending on the problem, grading will be based on: 1) the correct result, 2) coding efficiency and 3) graphical presentation features (labeling, colors, size, legibility, etc). I will be looking for well-rendered displays. Do not print and display the contents of vectors or data frames unless requested by the problem. You should be able to display each solution in fewer than ten lines of code.

Submit both the .Rmd and .html files for grading

Example Problem with Solution: Use *rbinom()* to generate two random samples of size 10,000 from the binomial distribution. For the first sample, use p = 0.45 and n = 10. For the second sample, use p = 0.55 and n = 10.

(a) Convert the sample frequencies to sample proportions and compute the mean number of successes for each sample. Present these statistics.

```
set.seed(123)
sample.one <- table(rbinom(10000, 10, 0.45)) / 10000
sample.two <- table(rbinom(10000, 10, 0.55)) / 10000

successes <- seq(0, 10)

sum(sample.one * successes) # [1] 4.4827
```

```
## [1] 4.4827
```

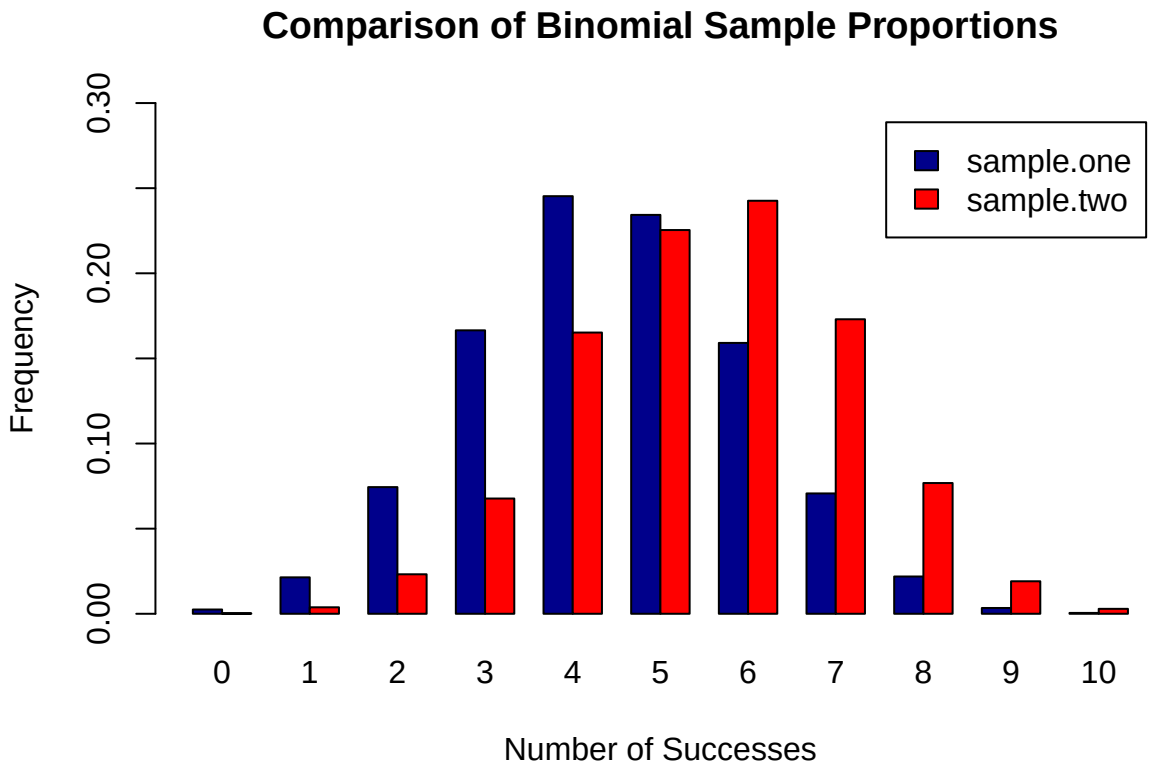
```
sum(sample.two * successes) # [1] 5.523
```

```
## [1] 5.523
```

(b) Present the proportions in a vertical, side-by-side barplot color coding the two samples.

```
counts <- rbind(sample.one, sample.two)

barplot(counts, main = "Comparison of Binomial Sample Proportions",
  ylab = "Frequency", ylim = c(0,0.3),xlab = "Number of Successes",
  beside = TRUE, col = c("darkblue","red"),legend = rownames(counts),
  names.arg = c("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10"))
```



Please delete the Instructions and Examples shown above prior to submitting your .Rmd and .html files.

Test Items

Read each question carefully and address each element. Do not print contents of vectors or data frames unless requested.

(1) (4 points) This problem deals with vector manipulations.

(1)(a) Create a vector that contains the following, in this order, and print the contents. Do not round off any values unless requested. * A sequence of integers from 0 to 6, inclusive. * Two repetitions of the vector c(2, -5.1, -23). * The sum of 7/42 and 3

(1)(b) Determine the length of the vector created in (1)(a) and denote as L. Print L. Generate a sequence starting with 1 and ending with L and add to the vector from (1)(a). This is vector addition, not vector combination. Print the contents. Do not round off any values.

(1)(c) Extract the first and last elements of the vector you have created in (1)(b) to form another vector using the extracted elements. Form a third vector from the elements not extracted. Print these vectors.

(1)(d) Use the vectors from (c) to reconstruct the vector in (b). Print this vector. Sum the elements and round to two decimal places.

---

(2) (5 points) The expression y = sin(x) - cos(x) is a trigonometric function.

(2)(a) Using the trigonometric function above, write a function as defined by R in the proper format that accepts values for x and returns a value for y.

(2)(b) Create a vector, x, of 4001 equally-spaced values from -2 to 2, inclusive. Compute values for y using the vector x and your function in (a). **Do not print x or y.** Find the value in the vector x that corresponds to the minimum value in the vector y. In other words, restrict attention to only the values of x and y you have computed. Round to 3 decimal places and print the value of x you find and the corresponding minimum value for y.

Finding the two desired values can be accomplished in as few as two lines of code. Do not use packages or programs you may find on the internet or elsewhere. Do not print the elements of the vectors x and y. Use coding methods shown in the *Quick Start Guide for R.*

(2)(c) Plot y versus x in color, with x on the horizontal axis. Show the location of the minimum value of y determined in 2(b). Add a title and other features such as text annotations. Text annotations may be added via *text()*.

---

(3) (8 points) Use the "trees" dataset for the following items. This dataset has three variables (Girth, Height, Volume) on 31 trees.

(3)(a) Use *data(trees)* to load the file. Check the structure with *str()*. Use *apply()* to return the median values for the three variables in "trees." Using R and logicals, print the row number and the three measurements: Girth, Height and Volume, of the tree with Volume equal to median Volume.

(3)(b) Girth is defined as the diameter of a tree taken at 4 feet 6 inches from the ground. Convert each diameter to a radius, r. Calculate the cross-sectional area of each tree using pi times the squared radius. Sort r ascending and print. Present the stem-and-leaf plot of the radii, and a histogram of the radii in color.

(3)(c) Use *par(mfrow = c(1, 4))* and present colored boxplots of the radii and areas calculated in (b) along with Height and Volume. Label each accordingly.

(3)(d) Demonstrate that the outlier revealed in the boxplot of Volume is not an extreme outlier. It is possible to do this with one line of code using *boxplot.stats* or logicals.

---

(4) (2 points) Use matrix operations shown in the "Quick Start Guide" to solve the following system of linear equations. Display the R script and the numerical solutions for x, y and z. Use matrix operations with your solution to reproduce the values 1, 1, 3 as a means of checking if your solution is correct. This last demonstration can be accomplished with matrix multiplication in one line of code. Print your result.

x - y + z = 1

x + y + z = 1

x + y - z = 3

---

(5) (6 points) The Cauchy distribution is an example of a "heavy-tailed" distribution in that it will have (more) outliers in both tails. This problem involves comparing it with a normal distribution which typically has very few outliers.

5(a) Use *set.seed(124)* and *rcauchy()* with n = 100, location = 0 and scale = 0.1 to generate a random sample designated as y. Generate a second random sample designated as x with *set.seed(127)* and *rnorm()* using n = 100, mean = 0 and sd = 0.15.

Generate a new object using *cbind(x, y)*. Do not print this object. Use *apply()* with this object to compute the inter-quartile range or IQR for each column: x and y. Round the results to four decimal places and present. (The point of this exercise is to demonstrate the similarity of the IQR values.)

For information about rcauchy(), use help in R (?rcauchy). **Do not print x and y.**

(5)(b) This item will illustrate the difference between a heavy-tailed distribution and one which does not have heavy tails. Use *par(mfrow = c(2, 2))* to generate a display with four diagrams. On the first row, present two histograms in color, one for x and the second for y. Set *xlim = c(-7, 7)* for the Cauchy results, and *xlim = c(-0.45, 0.45)* for the Normal results. Do not specify ylim for the histograms. On the second, show vertical boxplots for x and y. Use the xlim interval values respectively for the ylim of the boxplots.

(5)(c) QQ plots are useful for detecting the presence of heavy-tailed distributions. Use *par(mfrow = c(1, 2))* and present side-by-side plots, one for each sample, using *qqnorm()* and *qqline()*. Add color and titles. Use cex = 0.5 to control the size of the plotted data points.