Practice Problems for R Day 2

UVA Research Computing Summer Education Series

June 2019

For Beginners

Project 1

Suppose that your code needs to read a list of files with the names Data_01.csv, Data_02.csv, . . . , Data_99.csv. Create a list of these file names by

i) Using a for-loop,

ii) Using vectorization.

Which technique is easier for you? Which technique is more efficient?

Project 2

Sometimes, it is necessary to normalize data before analysis to ensure accurate results. The basic idea is to replace each value, x, in a column or list of data with

(x – mean)/stdev ,

where mean and stdev are the mean and standard deviation of the column or list.

Write a function that will take as input a list of numbers and will return the normalized list.

Test your function with the data

```
myList <- c(1, 2, 3, 4, 5).
```

Your result should be approximately (-1.2649, -0.6325, 0, 0.6325, 1.2649).

Download the file bodyfat.csv and read in the data.

https://workshops.somrc.virginia.edu/data/bodyfat.csv

Compute the normalized values for the column labeled percent.body.fat.

Project 3

In statistics, the error function (often called erf) produces the probability of a normally distributed value, x (with mean 0 and variance 0.5), falling within –x to x. In programming, this function can be computed as:

erf(x) = 2*pnorm(x * sqrt(2)) – 1

Write a function for erf and test it by computing erf(1.0). You should get approximately 0.842701.

Next, plot the points (x, erf(x)) for x between -3 and 3. The plot should show an S-shaped curve, called a sigmoid curve.

For Intermediates

Project 4

Generate a list of temperatures from -40 to 100 degrees Celsius (inclusive) by increments of 5 degrees.

i) Convert the temperatures to Fahrenheit and save as a separate list by using each of the following techniques (Hint: F = C*9/5 + 32) :
   a. Using a loop;
   b. Using vectorization.

ii) Use the temperatures to do the following:
   a. Write code that will identify the locations where the Celsius temperatures are less than zero and the Fahrenheit temperatures are greater than zero. Print a table of the results with two columns: Celsius and Fahrenheit.
   b. Write a function that will perform this same task. Pass to the function the lists of Celsius temperatures and Fahrenheit temperatures. Have the function create a data frame with the temperatures where Celsius is less than zero and Fahrenheit is greater than zero. Return the data frame to the calling code. Have the calling code print the data frame.

Project 5

Background for Project 5:

When you collect data in an experiment, you may want to know if any of the data are outliers and (although this may be controversial) throw out the outliers before performing any analysis.

There are several formulas for determining outliers. We will write code which uses the Chauvenet criterion. In brief, the Chauvenet criterion determines the probability of a value being outside of an expected range of values. An algorithm for the Chauvenet criterion is as follows:

Suppose A is a list of measurements.

1. For each element of A, compute its chauvenet value using the formula:

   chauvenet = n*(1-erf(abs(A[i] - mean)/standard_deviation))

   Note: Use the **erf** function that you wrote in Project #3 (above).

2. If chauvenet < 0.5, reject A[i] ; otherwise, keep A[i].

<u>What you are doing for Project 5</u>:

Write a program that

1. Reads the data from the file bodyfat.csv;
2. Identifies the rows (if any) where there are outliers in the weight column and the height column;
3. Updates the data frame to eliminate the rows with outliers;
4. Plots a histogram of the percent body fat; and
5. Plots a graph of percent body fat as a function of age.

Project 6

<u>Background for Project 6</u>:

The Collatz conjecture is a fun little exercise in number theory. You start with a positive integer. If the integer is odd, you multiply it by 3 and add 1. But, if the integer is even, you divide it by 2. You then take the resulting number and repeat the procedure. Mathematicians believe that if you keep repeating the procedure, you will eventually get the number 1 as a result. The number of steps required to reach 1 is called the *stopping time*.

No one has been able to prove that you *always* end up with a 1, regardless of which positive integer is the starting point. But, no exceptions have been found!

What you are doing for Project 6:

Write a program that takes a positive integer, N, and prints a table that lists the stopping time for the integers from 1 to N. The program will be required to do the following:

1. Start with a number, N, and verify that N is greater than zero.
2. Determine the stopping time for the integers from 1 to N;
3. Print a table of the results, where the columns are labeled "Integer" and "Stopping Time".

Test your program with N = 5. The expected output is

Integer	**Stopping Time**
1	0
2	1
3	7
4	2
5	5

After you are satisfied that the code performs correctly, run the program with N = 35.

For Experts

Project 7

Background for Project 7

I have a room thermometer on my desk that measures temperature and humidity. Every time someone (including myself) complained about the temperature, I would glance at the thermometer. I've noticed that we have the most complaints if both temperature and humidity are high or both temperature and humidity are low.

The optimal comfort level seems to be at 70 degrees Fahrenheit and with the humidity at 42%. However, there appears to be other combinations of temperature and humidity that feel comfortable to us. If the temperature increases, I don't hear any complaints as long as the humidity drops by at least 1.25 units for every one-degree increase in the temperature. Once we get to 81 degrees (or higher), we complain no matter what the humidity is. On the other hand, if the temperature drops, we are comfortable as long as the humidity increases by at least 4 units for every one degree that the temperature drops. Again, we all complain when the temperature is 66 degrees (or below), regardless of the humidity. I've also noticed that when the temperature is 70 degrees, the humidity can increase to 72% or decrease to 12% and still be comfortable.

These numbers for temperature and humidity are my theory based on empirical observations. I would like to test out my theory by having a program where I can enter the temperature and humidity, and it will tell me if the room is either comfortable, too hot, or too old.

What you are doing for Project 7

Write a program that takes as input, the temperature in degrees Fahrenheit and the relative humidity; have the program report on the comfort level of the room.

1. The program will use conditional logic and appropriate calculations to determine the comfort level of the room.

2. The program will write the results to the screen in a user-friendly format, stating whether the room is comfortable, too hot, or too cold.

3. Test your program with the following values:

Test Input	Expected Output
75 degrees, 28% humidity	Comfortable
66 degrees, 33.5% humidity	Too Cold
79.3 degrees, 33.5% humidity	Too Hot
70 degrees, 50% humidity	Comfortable

Project 8

Background for Project 8:

There may be times when you need to combine data from different sources to have a more complete set of data for analysis. In this project, you are going to read in data from three files and combine them into a single data frame. The data are related to atmospheric conditions that produce El Niño and La Niña weather patterns.

According to the National Oceanic and Atmospheric Administration (NOAA), El Niño and La Niña are two complex weather patterns that affect temperatures between the Pacific Ocean and the atmosphere.

El Niño is the warm phase of the cycle. That is, the waters in the Pacific Ocean, near the equator, are warmer than usual. Similarly, La Niña is the cold phase. Both the El Niño and La Niña patterns can impact weather and climate globally.

NOAA has a well-defined process for determining if an El Niño or La Niña pattern exists. However, it is somewhat complicated and only determines the weather pattern after several months of observations.

Data have been collected for many years. It would be interesting to investigate whether machine learning techniques could be used to predict an El Niño or a La Niña. But first, we must gather the data and put it into a usable format.

What you are doing for Project 8:

This project will have three step:

i) getting the data and reading it into your code;
ii) processing the data into a usable format; and
iii) performing some exploratory analysis on the data.

Step #1: Getting the data.

Some of the data that we will be using for this project are located on the UCI Machine Learning Repository. You will need to download the following files from https://archive.ics.uci.edu/ml/machine-learning-databases/el_nino-mld/:

- tao-all2.dat.gz
- tao-all2.col

The file ending in .gz is a compressed file. You can expand it using the gunzip function in the R.utils package – you may need to install R.utils. The file ending with .col is a list of the column names for the data in the compressed file. You will need to figure out how you to read the data from each file into your R script. For example, are the data comma-separated, space-separated, or something else?

If you need more information about the data, see https://archive.ics.uci.edu/ml/machine-learning-databases/el_nino-mld/el_nino.html
(Make sure that you also click on description of the data on that page.)

The next file that you need is oni.data, which is located at https://www.esrl.noaa.gov/psd/data/correlation/oni.data/
It contains values that NOAA has computed for various months/years.

The final file is patterns.csv.

https://workshops.somrc.virginia.edu/data/patterns.csv

This file will indicate whether the months in various years were labelled as El Niño, La Niña, or both.

Notice that the data files could have a header, or even a footer. You will need to think about how you can read in the data.

Step #2 Processing the data.

Combine the data so that you have a single data frame, which includes the measured values (tao-all2.dat), the computed values (oni.dat), and the categories (patterns.csv). Make sure that each column has an appropriate name. Note: The files oni.dat and patterns.csv may include months/years where there is no collected data in tao-all2.dat. Use the dates in tao-all2.dat as the guideline for the data to be included in your data frame.

Step #3 Exploring the data.

Explore the data so that you know the types and range of values. Look at the descriptive statistics. Look at plots or histograms of the data. Is there anything interesting that you can fnd?