Coding Challenge #5 - For Loop

SA Donohoo

2025-03-27

Table of Contents

[Github Link 1](#_Toc193963163)

[Question 1 - Reproducibility 1](#_Toc193963164)

[Question 2 - Explain Functions and For Loops 1](#_Toc193963165)

[Question 3 - Adding Data 2](#_Toc193963166)

[Question 4 - Generate a For Loop 3](#_Toc193963167)

[Question 5 - Compute a Single Distance 4](#_Toc193963168)

[Question 6 - For Loop to Compute Distances for All Cities 4](#_Toc193963169)

# Github Link

Sam Donohoo - PLPA 6820 - <https://github.com/sad0046/PLPA6820_SP25>

### Question 1 - Reproducibility

Regarding reproducibility, what is the main point of writing your own functions and iterations?

**Answer**: Functions and iterations simplify the code and help to minimize copy and paste errors that occur when you manually type out or paste/edit chunks of code.

### Question 2 - Explain Functions and For Loops

In your own words, describe how to write a function and a for loop in R and how they work. Give me specifics like syntax, where to write code, and how the results are returned.

**Answer**: A function is composed of variables, code to be run using said variables, and usually a way to output the results of the function. An initial function can be written without variables in either the consule or an R Script. For Example:

# Base Function without Assigning Values to Variables  
Sams.Function <- function(Variable1, Variable2) { # Inputs or Variables to Be Read. Uses {}.  
 Mathing <- Variable1 + Variable2 # Code to Be Run  
 return(Mathing) # Way to Output Results  
}

Now we can assign values to the two variables and run the function.These results are then printed in the consule or can appended to a dataframe.

# Base Function without Assigning Values to Variables  
Mathing <- Sams.Function(1:5,2)  
  
# Print Results  
print(Mathing)

## [1] 3 4 5 6 7

**Answer**: A for loop is composed of a variable (organized in a list/array/sequence) and code to be run using said variable. Since for loops iteratively execute a block of code either a specific number of times or for each item in a list/array, an input variable can be defined before the for loop or during the for loop. Both the variable and for loop can be written in the consule or an R Script. The results of the for loop are then printed in the consule or can appended to a dataframe. For Example:

# Input Variable Organized as a Vector  
Input.Variable <- 1:5  
  
# For Loop Using Previously Assigned Variable  
for (i in Input.Variable) { #   
 print(i + 5)  
}

## [1] 6  
## [1] 7  
## [1] 8  
## [1] 9  
## [1] 10

# For Loop Assigning Variable in Loop  
for (i in 1:5) { #   
 print(i + 5)  
}

## [1] 6  
## [1] 7  
## [1] 8  
## [1] 9  
## [1] 10

### Question 3 - Adding Data

Read in the Cities.csv file from Canvas using a relative file path.

# Load in Packages  
library(ggplot2)  
library(drc)   
library(tidyverse)  
library(dplyr)  
  
# Read in Data  
  
All.Cities <- read.csv("../Raw\_Data\_for\_Class\_Exercises/Cities.csv", header = TRUE, na.strings = "NA")

### Question 4 - Generate a For Loop

Write a function to calculate the distance between two pairs of coordinates based on the Haversine formula (see below). The input into the function should be lat1, lon1, lat2, and lon2. The function should return the object distance\_km. All the code below needs to go into the function.

#### Haversine formula

# Convert to radians  
rad.lat1 <- lat1 \* pi/180  
rad.lon1 <- lon1 \* pi/180  
rad.lat2 <- lat2 \* pi/180  
rad.lon2 <- lon2 \* pi/180  
  
# Haversine formula  
delta\_lat <- rad.lat2 - rad.lat1  
delta\_lon <- rad.lon2 - rad.lon1  
a <- sin(delta\_lat / 2)^2 + cos(rad.lat1) \* cos(rad.lat2) \* sin(delta\_lon / 2)^2  
c <- 2 \* asin(sqrt(a))   
  
# Earth's radius in kilometers  
earth\_radius <- 6378137  
  
# Calculate the distance  
distance\_km <- (earth\_radius \* c)/1000

#### Answer - Function

distance <- function(lat1,lon1,lat2,lon2) {  
 rad.lat1 <- lat1 \* pi/180  
 rad.lon1 <- lon1 \* pi/180  
 rad.lat2 <- lat2 \* pi/180  
 rad.lon2 <- lon2 \* pi/180  
 delta\_lat <- rad.lat2 - rad.lat1  
 delta\_lon <- rad.lon2 - rad.lon1  
 a <- sin(delta\_lat / 2)^2 + cos(rad.lat1) \* cos(rad.lat2) \* sin(delta\_lon / 2)^2  
 c <- 2 \* asin(sqrt(a))  
 earth\_radius <- 6378137  
 distance\_km <- (earth\_radius \* c)/1000  
  
}

### Question 5 - Compute a Single Distance

Using your function, compute the distance between Auburn, AL and New York City

* Subset/filter the Cities.csv data to include only the latitude and longitude values you need and input as input to your function.
* The output of your function should be 1367.854 km

# Subset and Preset New York Coordinates  
NewYork.lat <- All.Cities$lat[All.Cities$city == "New York"]  
NewYork.lon <- All.Cities$long[All.Cities$city == "New York"]  
  
# Subset and Preset Auburn Coordinates  
Auburn.lat <- All.Cities$lat[All.Cities$city == "Auburn"]  
Auburn.lon <- All.Cities$long[All.Cities$city == "Auburn"]  
  
# Generate Empty Dataframe  
Cities.Distance.NewYork <- NULL # create a null object  
  
# Run Function  
Distance.NewYork <- distance(NewYork.lat,NewYork.lon,Auburn.lat,Auburn.lon)  
  
# Combine Empty and Output Dataframe  
Cities.Distance.NewYork <- rbind.data.frame(Cities.Distance.NewYork, Distance.NewYork)  
  
# Edit Column Name  
colnames(Cities.Distance.NewYork) <- "km\_distance"  
  
# Print Values  
print(Cities.Distance.NewYork)

## km\_distance  
## 1 1367.854

### Question 6 - For Loop to Compute Distances for All Cities

Now, use your function within a for loop to calculate the distance between all other Cities in the data. Bonus point if you can have the output of each iteration append a new row to a dataframe, generating a new column of data. In other words, the loop should create a dataframe with three columns called city1, city2, and distance\_km, as shown below.

# Select All Cities That Are Not Auburn  
Cities <- All.Cities[All.Cities$city != "Auburn",]  
  
# Create a vector of City1 Names  
Cities.names <- unique(Cities$city)  
  
# Assign City2 Name  
Auburn.name <- "Auburn"  
  
# Generate Empty Dataframe  
Cities.Distance.All <- NULL # create a null object  
  
# Create for Loop that runs through all Cities and Calculates Distance to Auburn  
for (i in seq\_along(Cities.names)) {  
 Cities.Distance.Single <- distance(Cities$lat[i],  
 Cities$long[i],  
 Auburn.lat,  
 Auburn.lon) # Calculate Distance Between Two Cities  
 Temp.df <- data.frame(Cities$city[i],Auburn.name,Cities.Distance.Single) # Create a Row with City1, City2, and km\_Distance  
 Cities.Distance.All <- rbind.data.frame(Cities.Distance.All, Temp.df) # Append to Previous Datframe  
}  
  
# Edit Column Names  
colnames(Cities.Distance.All) <- c("City1", "City2", "km\_distance")  
  
# Print Values  
print(Cities.Distance.All)

## City1 City2 km\_distance  
## 1 New York Auburn 1367.8540  
## 2 Los Angeles Auburn 3051.8382  
## 3 Chicago Auburn 1045.5213  
## 4 Miami Auburn 916.4138  
## 5 Houston Auburn 993.0298  
## 6 Dallas Auburn 1056.0217  
## 7 Philadelphia Auburn 1239.9732  
## 8 Atlanta Auburn 162.5121  
## 9 Washington Auburn 1036.9900  
## 10 Boston Auburn 1665.6985  
## 11 Phoenix Auburn 2476.2552  
## 12 Detroit Auburn 1108.2288  
## 13 Seattle Auburn 3507.9589  
## 14 San Francisco Auburn 3388.3656  
## 15 San Diego Auburn 2951.3816  
## 16 Minneapolis Auburn 1530.2000  
## 17 Tampa Auburn 591.1181  
## 18 Brooklyn Auburn 1363.2072  
## 19 Denver Auburn 1909.7897  
## 20 Queens Auburn 1380.1382  
## 21 Riverside Auburn 2961.1199  
## 22 Las Vegas Auburn 2752.8142  
## 23 Baltimore Auburn 1092.2595  
## 24 St. Louis Auburn 796.7541  
## 25 Portland Auburn 3479.5376  
## 26 San Antonio Auburn 1290.5492  
## 27 Sacramento Auburn 3301.9923  
## 28 Austin Auburn 1191.6657  
## 29 Orlando Auburn 608.2035  
## 30 San Juan Auburn 2504.6312  
## 31 San Jose Auburn 3337.2781  
## 32 Indianapolis Auburn 800.1452  
## 33 Pittsburgh Auburn 1001.0879  
## 34 Cincinnati Auburn 732.5906  
## 35 Manhattan Auburn 1371.1633  
## 36 Kansas City Auburn 1091.8970  
## 37 Cleveland Auburn 1043.2727  
## 38 Columbus Auburn 851.3423  
## 39 Bronx Auburn 1382.3721