# Coding Practice 6

# Functions and Iterations

PLPA-5820 (Spring 2025)

Gabriel A. A. Silva

2025-03-25

## Contents



### Instructions

Loops and iterations – 25 pts

**1. 2 pts. Regarding reproducibility, what is the main point of writing your own functions and iterations?**

> It automates repetitive tasks, thus reducing potential errors. It also makes the code more readable, easier to maintain and troubleshoot.

**2. 2 pts. 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.**

> A function is created using `function()`. Inside the parenthesis, variables are defined and then the body is enclosed in curly braces. This structure is assined to a function name using `<-` and then it can be reused by calling the function name. At the end of the body, the command `return()` is used to explicit the object which will be returned by the function. A function can be written anywhere in the script, but it must be defined before it's called for the first time.
>
> A for loop is defined using the `for ()` command. Inside the parenthesis the iterable variable is defined (e.g., i in -30:100), and then the body is enclosed by curly braces. Each iteration, the variable (in this case `i`), will hold the value of the current iteration, and can be used inside the body.

This dataset contains the population and coordinates (latitude and longitude) of the 40 most populous cities in the US, along with Auburn, AL. Your task is to create a function that calculates the distance between Auburn and each other city using the Haversine formula. To do this, you'll write a for loop that goes through each city in the dataset and computes the distance from Auburn. Detailed steps are provided below.

```
# Set up the environment
knitr::opts_chunk$set(results = 'hold', message = FALSE,
                      warning = FALSE, fig.show = "hold",
                      out.width = "80%", fig.align = "center",
                      cache=TRUE)
```

```r
# Import general libraries
library(tidyverse)
```

**3. 2 pts. Read in the Cities.csv file from Canvas using a relative file path.**

```r
# Read in the data
data <- read_csv("Cities.csv")
```

**4. 6 pts. 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.**

```r
# Function to calculate the distance between two pairs of coordinates
dist_converter <- function(lat1, lon1, lat2, lon2) {

  # 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

  return(distance_km)
}
```

**5. 5 pts. Using your function, compute the distance between Auburn, AL and New York City**

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

```r
# Filter the data
auburn <- data %>% filter(city == "Auburn", state_id == "AL")
new_york <- data %>% filter(city == "New York", state_id == "NY")

# Calculate the distance
au_ny <- dist_converter(auburn$lat, auburn$long, new_york$lat, new_york$long)

paste("Distance Auburn, AL and New York City:", round(au_ny, 3), "km")
```

```
## [1] "Distance Auburn, AL and New York City: 1367.854 km"
```

**6. 6 pts. Now, use your function within a for loop to calculate the distance between all other cities in the data. The output of the first 9 iterations is shown below.**

> I was unsure if it was Auburn vs all cities or all cities vs all cities. I did both, and also did one with the `combn()` function to avoid duplicates since the distance between City1 and City2 is the same as City2 and City1.

```
# Auburn vs all cities
dist_df <- data.frame(city1 = character(),
                      city2 = character(),
                      distance_km = numeric())

for (i in 1:nrow(data)) {
  distance <- dist_converter(auburn$lat, auburn$long,
                             data$lat[i], data$long[i])

  dist_df <- rbind(dist_df, data.frame(city1 = data$city[i],
                                       city2 = "Auburn",
                                       distance_km = distance))
}

head(dist_df)
```

```
##         city1  city2 distance_km
## 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
```

```
# All cities vs all cities
dist_df <- data.frame(city1 = character(),
                      city2 = character(),
                      distance_km = numeric())

for (i in 1:nrow(data)) {
  for (j in 1:nrow(data)) {
    distance <- dist_converter(data$lat[i], data$long[i],
                               data$lat[j], data$long[j])
    dist_df <- rbind(dist_df, data.frame(city1 = data$city[i],
                                         city2 = data$city[j],
                                         distance_km = distance))
  }
}

head(dist_df)
```

```
##      city1       city2 distance_km
## 1 New York    New York       0.000
## 2 New York Los Angeles    3958.048
## 3 New York     Chicago    1157.245
## 4 New York       Miami    1758.655
## 5 New York     Houston    2288.873
## 6 New York      Dallas    2210.803
```

```r
# All cities vs all cities without duplicates
data_lists <- as.list(as_tibble(t(data)))

comp_df <- combn(data_lists, 2, function(x) {
  city1 <- as.numeric(c(x[[1]][7], x[[1]][8]))
  city2 <- as.numeric(c(x[[2]][7], x[[2]][8]))

  distance <- dist_converter(city1[1], city1[2],city2[1], city2[2])
  c(city1 = x[[1]][1], city2 = x[[2]][1], distance_km = round(distance, 4))
})

comp_df <- as.data.frame(t(comp_df))

colnames(comp_df) <- c("city1", "city2", "distance_km")

head(comp_df)
```

```
##      city1        city2 distance_km
## 1 New York  Los Angeles   3958.0481
## 2 New York      Chicago   1157.2449
## 3 New York        Miami   1758.6549
## 4 New York      Houston    2288.873
## 5 New York       Dallas   2210.8033
## 6 New York Philadelphia    127.9107
```

**7. 2 pts. Commit and push a gfm .md file to GitHub inside a directory called Coding Challenge 6. Provide me a link to your github written as a clickable link in your .pdf or .docx**

Link : Coding Challenge 6