Coding Challenge 6

Dustyn Lewis

2025-03-27

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

-Writing your own functions and using iterations (loops) makes your code more reproducible by reducing repetition and human error. Instead of copying and pasting the same code many times, you encapsulate the logic in a function or a loop which is easier to maintain and share. Also, changes made in one place apply everywhere consistently.

##2) In your own words, describe how to write a function and a for loop in R, including syntax, where to write code, and how results are returned:

-A function in R is created with the function keyword, followed by parentheses containing parameters, and braces ({}) enclosing the code. For example:

```
my_function <- function(x, y) { # You can define functions at the top of
  result <- x + y
  return(result)                #"return()" states which caluess
}
```

-A for loop in R uses the for keyword, a variable in parentheses, and braces for the loop body. For example:

```
for (i in 1:5) {                # Loops iterate over a collection of
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

##3) Read in "Cities.csv" as relative file path:

-The data set contains 40 most populous cities plut Auburn, I've downloaded the data set into my working directory already.

```
cities <- read.csv("Cities.csv")
head(cities)
```

```
##          city  city_ascii state_id state_name county_fips county_name     lat
## 1    New York    New York       NY   New York       36081      Queens 40.6943
## 2 Los Angeles Los Angeles       CA California        6037 Los Angeles 34.1141
## 3     Chicago     Chicago       IL   Illinois       17031        Cook 41.8375
```

```
## 4       Miami       Miami       FL    Florida       12086  Miami-Dade 25.7840
## 5     Houston     Houston       TX      Texas       48201      Harris 29.7860
## 6      Dallas      Dallas       TX      Texas       48113      Dallas 32.7935
##        long population density
## 1  -73.9249   18832416 10943.7
## 2 -118.4068   11885717  3165.8
## 3  -87.6866    8489066  4590.3
## 4  -80.2101    6113982  4791.1
## 5  -95.3885    6046392  1386.5
## 6  -96.7667    5843632  1477.2
```

##4) Write a function to calculate distance between two pairs of coordinates using Haversine formul:

-Input = "lat1, lon1, lat2, lon2" -Output = "distance_km"

```r
haversine_distance <- function(latA, lonA, latN, lonN) {

  # Convert degrees to radians
  rad.latA <- latA * pi / 180
  rad.lonA <- lonA * pi / 180
  rad.latN <- latN * pi / 180
  rad.lonN <- lonN * pi / 180

  # Differences
  delta_lat <- rad.latN - rad.latA
  delta_lon <- rad.lonN - rad.lonA

  # Haversine formula
  a <- sin(delta_lat / 2)^2 + cos(rad.latA) * cos(rad.latN) * sin(delta_lon / 2)^2
  c <- 2 * asin(sqrt(a))

  # Earth's radius in meters; convert to km
  earth_radius <- 6378137
  distance_km <- (earth_radius * c) / 1000

  return(distance_km)
}
```

##5) Test function by computing distance between Auburn and NYC:

-We need to identify rows for Auburn and NYC, extract lat/lon, apply "haversine_distance()", and print the results which we expext to be ~1367.854 km.

```r
# Filter or subset the data:
Auburn <- subset(cities, city == "Auburn")
NYC <- subset(cities, city == "New York")

# Extract their lat/lon
latA <- Auburn$lat
lonA <- Auburn$long
latN <- NYC$lat
lonN <- NYC$long

# Calculate the distance
```

```r
dist_auburn_nyc <- haversine_distance(latA, lonA, latN, lonN)
dist_auburn_nyc
```

```
## [1] 1367.854
```

##6) Calculate distances between Auburn and all cities:

-We will use a loop to compute distances between Auburn and all other cities similiar to the prior function, except we use "for (i in 1:nrow(cities)) {...}"

```r
# 1) Extract Auburn's lat/lon
latA <- Auburn$lat
lonA <- Auburn$long

# 2) We will loop over all rows in "cities" EXCEPT Auburn's row, or skip if city is "Auburn".
for (i in 1:nrow(cities)) {
  if (cities$city[i] == "Auburn") {
    next # skip Auburn itself
  }

  lat2 <- cities$lat[i]
  lon2 <- cities$long[i]

  dist_km <- haversine_distance(latA, lonA, lat2, lon2)
  print(dist_km)
}
```

```
## [1] 1367.854
## [1] 3051.838
## [1] 1045.521
## [1] 916.4138
## [1] 993.0298
## [1] 1056.022
## [1] 1239.973
## [1] 162.5121
## [1] 1036.99
## [1] 1665.699
## [1] 2476.255
## [1] 1108.229
## [1] 3507.959
## [1] 3388.366
## [1] 2951.382
## [1] 1530.2
## [1] 591.1181
## [1] 1363.207
## [1] 1909.79
## [1] 1380.138
## [1] 2961.12
## [1] 2752.814
## [1] 1092.259
## [1] 796.7541
## [1] 3479.538
## [1] 1290.549
```

```
## [1] 3301.992
## [1] 1191.666
## [1] 608.2035
## [1] 2504.631
## [1] 3337.278
## [1] 800.1452
## [1] 1001.088
## [1] 732.5906
## [1] 1371.163
## [1] 1091.897
## [1] 1043.273
## [1] 851.3423
## [1] 1382.372
```

##Bonus) Building a Dataframe

-We can make a dataframe with three columns: City1|City2|Distance_km| where each time we loop over a city we append/add additionnal rows.

```r
# Create an empty data frame
distance_df <- data.frame(city1 = character(),
                          city2 = character(),
                          Distance_km = numeric(),
                          stringsAsFactors = FALSE)

for (i in 1:nrow(cities)) {
  if (cities$city[i] == "Auburn") next

  lat2 <- cities$lat[i]
  lon2 <- cities$long[i]

  dist_km <- haversine_distance(latA, lonA, lat2, lon2)

  # Create a one-row data frame for the pair: (City i, Auburn)
  new_row <- data.frame(city1 = cities$city[i],
                        city2 = "Auburn",
                        Distance_km = dist_km,
                        stringsAsFactors = FALSE)

  # Append to the main data frame
  distance_df <- rbind(distance_df, new_row)
}

# Print the first few rows
head(distance_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
```

##7) Commit and Push GitHub