Coding Challenge 6

Sarah Batten

2025-03-27

Question 1

The main purpose of writing your own functions and iterations is to make your analysis more reproducible and less prone to copy and pasting errors. When multiple code chunks are being copied and pasted with variables being switched around, it makes the script very likely to be wrong by mistake. It also simplifies your code.

Question 2

A function is written by having the function name, followed by the command function then the variables that are required followed by a curly bracket. The code for the funtion is then underneath this with the desired output being listed at the end with the return function, ending with the second curly bracket:

function_name <- function(variable){ code for function, using variable return(desired_output) } This function can then be called using its name, and will apply the code to the variable given, then return the desired_output.

The for loop runs the code within the loop for iterator, called i, from the start condition to the end conditions that are specified in the loop setup. Curly brackets are used before and after the code that the variable/iterator will be applied to, as with the function:

for (i in 1:10){ print(i - 1) }

The results will be printed to the console unless it is specified within the loop that it should be saved to a dataframe, and the null dataframe is created before the loop is implemented.

Question 3

Reading data into R:

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

Question 4

Creating function to calculate distances using Haversine formula:

```
dist_calc <- function(lat1, long1, lat2, log2){
rad.lat1 <- lat1 * pi/180
rad.lon1 <- long1 * pi/180
rad.lat2 <- lat2 * pi/180
```

```
rad.lon2 <- log2 * 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)
}
```

Question 5

Computing difference in distance between New York City and Auburn using the function created:

```
cities_subset <- subset(cities, city %in% c("New York", "Auburn"), select = c(long, lat)) # subsetting
print(cities_subset) # returning values to see to input into function
```

```
##        long     lat
## 1  -73.9249 40.6943
## 40 -85.4903 32.6087
```

```
nyc_auburn <- dist_calc(40.6943, -73.9249, 32.6087, -85.4903) # using subset data in function
print(nyc_auburn)
```

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

Question 6

Using function in a loop to calculate the distance between Auburn and all the other cities:

```
dist_all <- NULL
for (i in 1:nrow(cities)){
  distance_i <- dist_calc(cities$lat[i], cities$long[i], 32.6087, -85.4903) # applying function to ever
  dist_all <- rbind.data.frame(dist_all, distance_i) # making matrix
}

print(dist_all) # printing
```

```
##    X1367.85395084397
## 1          1367.8540
## 2          3051.8382
## 3          1045.5213
## 4           916.4138
## 5           993.0298
## 6          1056.0217
## 7          1239.9732
## 8           162.5121
```

```
## 9          1036.9900
## 10         1665.6985
## 11         2476.2552
## 12         1108.2288
## 13         3507.9589
## 14         3388.3656
## 15         2951.3816
## 16         1530.2000
## 17          591.1181
## 18         1363.2072
## 19         1909.7897
## 20         1380.1382
## 21         2961.1199
## 22         2752.8142
## 23         1092.2595
## 24          796.7541
## 25         3479.5376
## 26         1290.5492
## 27         3301.9923
## 28         1191.6657
## 29          608.2035
## 30         2504.6312
## 31         3337.2781
## 32          800.1452
## 33         1001.0879
## 34          732.5906
## 35         1371.1633
## 36         1091.8970
## 37         1043.2727
## 38          851.3423
## 39         1382.3721
## 40            0.0000
```

Link to GitHub: Coding Challenge 6 Directory