Codingchallenge6

Samit Kafle

2025-03-27

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

1. 2 pts. Regarding reproducibility, what is the main point of writing your own functions and iterations? Functions encapsulate specific tasks, making code modular and easier to understand, debug, and reuse. Iterations (loops) allow us to apply the same operation across multiple elements, ensuring consistent processing.Functions and loops reduce redundancy, making code more efficient and less error-prone.

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.

```r
#Define a function to calculate the square of a number*
#square <- function(x) {   This defines a function named square that takes one argument x
#  result <- x * x        This line calculates the square of x and stores it in result.
#  return(result)          This returns the value of result
#}

# Call the function
# square(4)  # Returns 16
```

```r
### For loop to calculate the square of each number
### Initialize an empty vector to store results
##squares <- c()

#for (i in numbers) {  This defines a loop that iterates over each element in the numbers
#squares <- c(squares, i * i)  This line calculates the square of i and appends it to the
#}

### Print the results
##print(squares)  # Returns c(1, 4, 9, 16, 25)
```

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

```
datum=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.

```
# Define the function to calculate distance using the Haversine formula
haversine_distance <- 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 the distance
  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.

```
# Filter the data to include only Auburn, AL and New York City
filtered_data <- datum %>%
  filter(city %in% c("Auburn", "New York"))

# Extract latitude and longitude values
lat1=datum[datum$city=="Auburn", "lat"]
lon1=datum[datum$city=="Auburn", "long"]
lat2=datum[datum$city=="New York", "lat"]
lon2=datum[datum$city=="New York", "long"]
```

b. The output of your function should be 1367.854 km

```
distance <- haversine_distance(lat1, lon1, lat2, lon2)
print(distance)
```

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

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.

```r
# empty list to store distances
distances <- list()

# Iterate over each pair of cities in the data
for (i in 1:nrow(datum)) {
lat1=datum[datum$city=="Auburn", "lat"]
lon1=datum[datum$city=="Auburn", "long"]
lat2=datum[i, "lat"]
lon2=datum[i, "long"]


    # Calculate the distance using the function
    distance <- haversine_distance(lat1, lon1, lat2, lon2)

    # Append the result to the distances list
    distances <- append(distances, distance)
  }
print(distances)
```

```
## [[1]]
## [1] 1367.854
##
## [[2]]
## [1] 3051.838
##
## [[3]]
## [1] 1045.521
##
## [[4]]
## [1] 916.4138
##
## [[5]]
## [1] 993.0298
##
## [[6]]
## [1] 1056.022
##
## [[7]]
## [1] 1239.973
##
## [[8]]
## [1] 162.5121
##
## [[9]]
## [1] 1036.99
##
## [[10]]
## [1] 1665.699
##
## [[11]]
## [1] 2476.255
```

```
##
## [[12]]
## [1] 1108.229
##
## [[13]]
## [1] 3507.959
##
## [[14]]
## [1] 3388.366
##
## [[15]]
## [1] 2951.382
##
## [[16]]
## [1] 1530.2
##
## [[17]]
## [1] 591.1181
##
## [[18]]
## [1] 1363.207
##
## [[19]]
## [1] 1909.79
##
## [[20]]
## [1] 1380.138
##
## [[21]]
## [1] 2961.12
##
## [[22]]
## [1] 2752.814
##
## [[23]]
## [1] 1092.259
##
## [[24]]
## [1] 796.7541
##
## [[25]]
## [1] 3479.538
##
## [[26]]
## [1] 1290.549
##
## [[27]]
## [1] 3301.992
##
## [[28]]
## [1] 1191.666
##
## [[29]]
## [1] 608.2035
```

```
##
## [[30]]
## [1] 2504.631
##
## [[31]]
## [1] 3337.278
##
## [[32]]
## [1] 800.1452
##
## [[33]]
## [1] 1001.088
##
## [[34]]
## [1] 732.5906
##
## [[35]]
## [1] 1371.163
##
## [[36]]
## [1] 1091.897
##
## [[37]]
## [1] 1043.273
##
## [[38]]
## [1] 851.3423
##
## [[39]]
## [1] 1382.372
##
## [[40]]
## [1] 0
```

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. The first six rows of the dataframe are shown below.

```r
# Append the result to the dataframe
dist=data.frame(t(distances))# transforming the matrix to make it column
distance=t(dist)#
distances_list <- rbind(data.frame(City1 = "Auburn", City2 = datum$city, Distance_km = distance))
print(distances_list)
```

```
##      City1        City2 Distance_km
## X1  Auburn     New York    1367.854
## X2  Auburn  Los Angeles    3051.838
## X3  Auburn      Chicago    1045.521
## X4  Auburn        Miami    916.4138
## X5  Auburn      Houston    993.0298
## X6  Auburn       Dallas    1056.022
## X7  Auburn Philadelphia    1239.973
```

```
## X8  Auburn       Atlanta    162.5121
## X9  Auburn    Washington     1036.99
## X10 Auburn        Boston    1665.699
## X11 Auburn       Phoenix    2476.255
## X12 Auburn       Detroit    1108.229
## X13 Auburn       Seattle    3507.959
## X14 Auburn San Francisco    3388.366
## X15 Auburn     San Diego    2951.382
## X16 Auburn   Minneapolis      1530.2
## X17 Auburn         Tampa    591.1181
## X18 Auburn      Brooklyn    1363.207
## X19 Auburn        Denver     1909.79
## X20 Auburn        Queens    1380.138
## X21 Auburn     Riverside     2961.12
## X22 Auburn     Las Vegas    2752.814
## X23 Auburn     Baltimore    1092.259
## X24 Auburn     St. Louis    796.7541
## X25 Auburn      Portland    3479.538
## X26 Auburn   San Antonio    1290.549
## X27 Auburn    Sacramento    3301.992
## X28 Auburn        Austin    1191.666
## X29 Auburn       Orlando    608.2035
## X30 Auburn      San Juan    2504.631
## X31 Auburn      San Jose    3337.278
## X32 Auburn  Indianapolis    800.1452
## X33 Auburn    Pittsburgh    1001.088
## X34 Auburn    Cincinnati    732.5906
## X35 Auburn     Manhattan    1371.163
## X36 Auburn   Kansas City    1091.897
## X37 Auburn      Cleveland   1043.273
## X38 Auburn      Columbus    851.3423
## X39 Auburn         Bronx    1382.372
## X40 Auburn        Auburn           0
```

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 to the github