Coding Challenge 6

Prativa Chhetri and Karamjit Kaur Baryah

2025-03-27

Table of Contents

# Question 1

· reducing the chance of errors from manual repetition.

· Custom functions make your code more organized and reusable.

· If a built-in function changes in future versions of R or a package update alters behavior, your custom function ensures stability

· Writing your own functions allows you to tailor calculations, data transformations, or iterations to specific requirements.

# Question 2

1. Writing a Function in R

A function in R is a reusable block of code that takes inputs (arguments), processes them, and returns an output. Functions help keep code organized, reusable, and easier to debug.

Syntax

$$my\_function = function(arg1, arg2) {

result = arg1 + arg2 # Perform operation

return(result) # Return output

}$$

How it Works:

· Define the function with function().

· Inside {}, write the code to process inputs.

· Use return() to output a value (if omitted, R returns the last evaluated expression).

· Call the function like this:

$$my\_function(3, 5) # should return to 8$$

1. Writing a for Loop in R

A for loop in R repeats a block of code for each value in a sequence (like a vector).

Syntax $$ for (i in 1:5) {

print(i) # Prints numbers 1 to 5

} $$

How it Works:

· for (i in sequence): Iterates over each value in sequence.

· Inside {}, write the code that executes on each iteration.

· The loop stops when all values in sequence are processed.

$$squares = c()

for (i in 1:5) {

squares[i] = i^2 # Store squared values

}

print(squares) # Returns [1] 1 4 9 16 25 $$

# Question 3

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

# Question 4

haversine\_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 # in meters  
 distance\_km <- (earth\_radius \* c) / 1000  
   
 return(distance\_km)  
}

# Question 5

# Filter data for Auburn, AL and New York City  
auburn <- cities %>% filter(city == "Auburn")  
nyc <- cities %>% filter(city == "New York")  
  
# Compute distance between Auburn, AL and New York City  
distance\_nyc\_auburn <- haversine\_distance(auburn$lat, auburn$long, nyc$lat, nyc$long)  
print(distance\_nyc\_auburn)

## [1] 1367.854

results <- data.frame(City1 = character(), City2 = character(), Distance\_km = numeric(), stringsAsFactors = FALSE)  
  
for (i in 1:nrow(cities)) {  
 city <- cities[i, ]  
 if (city$city != "Auburn") {  
 dist <- haversine\_distance(auburn$lat, auburn$long, city$lat, city$long)  
 results <- rbind(results, data.frame(City1 = city$city, City2 = "Auburn", Distance\_km = dist))  
 }  
}  
print(head(results))

## 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