Iterations and Functions

Billy Lozowski

2025-03-27

Contents

convert to radians **2**

Haversine formula **2**

Earth's radius in metres **2**

Calculate the distance **2**

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

Writing your own functions & iterations means the exact same process is performed each time you run the code. They not only save time, but they also help to limit syntax errors which may prevent long coding scripts from working correctly.

2. 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 needs and name (specific to the job it does), and code to perform a certain task. For instance, if we wanted to find the log value for each value in a data frame, we could write a function to do just that... For example:

```
log_value <- function(variable_name){
  log_x <- log(variable_name)
  return(log_x)
}

log_value(20)
```

```
## [1] 2.995732
```

```
# continue for other variables
```

A loop on the other hand, repeats a set of commands on selected data (e.g. multiplying it by a certain number. For example:

```
log.df = NULL
for(i in 1:100){
  result_i <- data.frame(i, log_value(i))
  log.df <- rbind.data.frame(log.df, result_i)
}

colnames(log.df) <- c("x", "log(x)")
```

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

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

4. 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.

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 metres

earth_radius <- 6378137

Calculate the distance

distance_km <- (earth_radius * c)/1000

```
distance <- function(lat1, lon1, lat2, lon2){
  # convert latitudes and longitudes 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 metres
  earth_radius <- 6378137

  # Calculate the distance in km
  distance_km <- (earth_radius * c)/1000

  # print the distance in km
  return(distance_km)
}

# New York to Los Angeles
paste0(distance(cities[1,7], cities[1,8], cities[2,7], cities[2,8]), " km")
```

```
## [1] "3958.0481422106 km"
```

5. 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

```
library(tidyverse)
```

```
## Warning: package 'stringr' was built under R version 4.4.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
auburn.NY <- cities %>%
  filter(city == "New York" | city == "Auburn")

distance(auburn.NY[1,7], auburn.NY[1,8], auburn.NY[2,7], auburn.NY[2,8])
```

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

```
# this was how I originally calculated the distance before reading the question fully...
# paste0(distance(cities[1,7], cities[1,8], cities[40,7], cities[40,8]), "km")
```

6. 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
# Create a vector containing all city names
u.city <- unique(cities$city)

# Get the latitude and longitude of Auburn
auburn_lat <- cities[cities$city == "Auburn", "lat"]
auburn_long <- cities[cities$city == "Auburn", "long"]

# Create a data frame with the columns "city1", "city2", and "distance_km"
auburn.US <- data.frame(city1 = character(),
                        city2 = character(),
                        distance_km = numeric(),
                        stringsAsFactors = FALSE)

for(i in  seq_along(u.city)){
  city1 <- cities[i, "city"]
  city2 <- "Auburn"
  lat1 <- cities[i, "lat"]
  lon1 <- cities[i, "long"]

# Using the distance function, calculate the distance in km between Auburn and city(i)
  distance_km <- distance(auburn_lat, auburn_long, lat1, lon1)

# Bind each data frame to auburn.US
auburn.US <- rbind(auburn.US, data.frame(city1 = city1,
                                         city2 = city2,
                                         distance_km = distance_km))
}

# Print the first 9 rows of auburn.US
print(head(auburn.US, 9))
```

```
##          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 Philadelphia Auburn   1239.9732
## 8      Atlanta Auburn    162.5121
## 9   Washington Auburn   1036.9900
```