CodeChal6

2025-03-27

Table of Contents

1. 2 pts. Regarding reproducibility, what is the main point of writing your own functions and iterations?
   * To create functions that can complete the problems needed to limit copy paste errors for reproducibility because the function can do all the work with 1 press in R
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.
   * Function: name the object and then use the word ‘function’ to start the function, with the word of input in () after function. Example- function(word). the use { } to include your equation that you want the function to use. The function will then input whatever number you put into the object name wherever you have the word in the equation.
   * For loops: Allow you to run a function over a set number of numbers. To do this, you use for(i in #:#) with the range in numbers being what you want.Then you can include functions or equations that you want the loop to run. The loop will run all the numbers specified through the loop and will put out whatever you tell it to- print, df, etc.
3. 2 pts. Read in the Cities.csv file from Canvas using a relative file path.

cities<- read.csv("cities.csv", na.strings = 'na')  
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

str(cities)

## 'data.frame': 40 obs. of 10 variables:  
## $ city : chr "New York" "Los Angeles" "Chicago" "Miami" ...  
## $ city\_ascii : chr "New York" "Los Angeles" "Chicago" "Miami" ...  
## $ state\_id : chr "NY" "CA" "IL" "FL" ...  
## $ state\_name : chr "New York" "California" "Illinois" "Florida" ...  
## $ county\_fips: int 36081 6037 17031 12086 48201 48113 42101 13121 11001 25025 ...  
## $ county\_name: chr "Queens" "Los Angeles" "Cook" "Miami-Dade" ...  
## $ lat : num 40.7 34.1 41.8 25.8 29.8 ...  
## $ long : num -73.9 -118.4 -87.7 -80.2 -95.4 ...  
## $ population : int 18832416 11885717 8489066 6113982 6046392 5843632 5696588 5211164 5146120 4355184 ...  
## $ density : num 10944 3166 4590 4791 1386 ...

1. 6 pts. Write a function to calculate the distance between two pairs of coordinates based on the Haversine formula. 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 kilometers

earth\_radius <- 6378137

# Calculate the distance

distance\_km <- (earth\_radius \* c)/1000

distance <- function(lat1, long1, lat2, long2){  
rad.lat1 <- lat1 \* pi/180  
rad.long1 <- long1 \* pi/180  
rad.lat2 <- lat2 \* pi/180  
rad.long2 <- long2 \* pi/180  
delta\_lat <- rad.lat2 - rad.lat1  
delta\_long <- rad.long2 - rad.long1  
a <- sin(delta\_lat / 2)^2 + cos(rad.lat1) \* cos(rad.lat2) \* sin(delta\_long / 2)^2  
c <- 2 \* asin(sqrt(a))   
earth\_radius <- 6378137  
distance\_km <- (earth\_radius \* c)/1000  
return(distance\_km)  
}

1. 5 pts. Using your function, compute the distance between Auburn, AL and New York City
   * Subset/filter the Cities.csv data to include only the latitude and longitude values you need and input as input to your function.
   * The output of your function should be 1367.854 km

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.4.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.4 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.4   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

cities$lat[cities$city == "Auburn"]

## [1] 32.6087

#subset data to just the cities, lat, and long  
lat\_long <- function(lat1, long1, lat2, long2){  
 lat1 <- cities$lat[cities$city == "Auburn"]  
 long1 <- cities$long[cities$city == "Auburn"]  
 lat2 <- cities$lat[cities$city == "New York"]  
 long2 <- cities$long[cities$city == "New York"]  
 rad.lat1 <- lat1 \* pi/180  
 rad.long1 <- long1 \* pi/180  
 rad.lat2 <- lat2 \* pi/180  
 rad.long2 <- long2 \* pi/180  
 delta\_lat <- rad.lat2 - rad.lat1  
 delta\_long <- rad.long2 - rad.long1  
 a <- sin(delta\_lat / 2)^2 + cos(rad.lat1) \* cos(rad.lat2) \* sin(delta\_long / 2)^2  
 c <- 2 \* asin(sqrt(a))   
 earth\_radius <- 6378137  
 distance\_km <- (earth\_radius \* c)/1000  
 return(distance\_km)  
}  
  
#plugging in values into the function created above  
lat\_long(lat1, long1, lat2, long2) #distance = 1367.854 km

## [1] 1367.854

1. 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. +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.

nm <- unique(cities$city)  
  
lat\_longall <- function(lat1, long1, lat2, long2){  
 lat1 <- cities$lat[cities$city == "Auburn"]  
 long1 <- cities$long[cities$city == "Auburn"]  
 lat2 <- cities$lat[cities$city == nm[[i]]]  
 long2 <- cities$long[cities$city == nm[[i]]]  
 rad.lat1 <- lat1 \* pi/180  
 rad.long1 <- long1 \* pi/180  
 rad.lat2 <- lat2 \* pi/180  
 rad.long2 <- long2 \* pi/180  
 delta\_lat <- rad.lat2 - rad.lat1  
 delta\_long <- rad.long2 - rad.long1  
 a <- sin(delta\_lat / 2)^2 + cos(rad.lat1) \* cos(rad.lat2) \* sin(delta\_long / 2)^2  
 c <- 2 \* asin(sqrt(a))   
 earth\_radius <- 6378137  
 distance\_km <- (earth\_radius \* c)/1000  
 return(distance\_km)  
}  
  
distance.df <- NULL  
for(i in seq\_along(nm)){  
 results\_i <- data.frame(lat\_longall(nm[[i]]))  
 distance.df <- rbind.data.frame(distance.df, results\_i)  
}  
view(distance.df)

1. 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

[grm.md file](https://github.com/kingjad33/PLPA6820Class/blob/main/Code%20Challenge%206/CodeChal6.md)