CodingChallenge6_Loops and iterations

Pankaj Gaonkar

2025-03-27

1. 2 pts. Regarding reproducibility, what is the main point of writing your own functions and iterations?

Answer: They are useful when you need to perform the same code on different data and you want to avoid copy and paste errors. Especially with function we can specify the formula to be performed and with iterations we can have several repetitions which wont be practically possible to write without error. Overall they help in reproducibility by avoiding the errors.

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.

Answer:

We write a code in R chunk in R markdown.I have explained the functions and loop using the example.

Example of Function:

Function:

```
F_to_C <- function(f_temp){
  celsius <- (5*(f_temp - 32)/9)  # we are saving value in the celsius
  return(celsius)             # gives the answer to the input after running the functions. We get Farhe
}
```

- Function is defined as F_to_C
- f_temp is our input
- Calling the function by giving input as 32 for this example:
- F_to_C(32) . Here we wil get the 32F to its celsius value.

Loop:

Example of "for" loop:

```
celcius.df <- NULL
for (i in -30:100){
  result <- data.frame(F_to_C(i), i)   #create one row dataframe called data.frame contain two columns
  celcius.df <- rbind.data.frame(celcius.df, result)  #binding celsius.df to result

}

celcius.df
```

- We start by creating a NULL object so that we can call that dataframe and we can have to output values.

- for (i in -30:100) – This is a for loop where it will go through every value between -30 tp 100 Farheneit.
- We create results as data.frame as per desired column arrangement.
- Finally we input the results in the initial NULL dataframe.
- cecius,df will print out our entire dataframe with our loop output.

```
#load packages

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
```

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

```
#load data

datum <- read.csv("Cities.csv")

head(datum)
```

```
##          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
```

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.

```
Distance_pair <- 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(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. b. The output of your function should be 1367.854 km

```
#Subsetting data to include only new york and auburn

  lat1 <- datum$lat[datum$city_ascii == "Auburn"]
  lon1 <- datum$long[datum$city_ascii == "Auburn"]

  lat2 <- datum$lat[datum$city_ascii == "New York"]
  lon2 <- datum$long[datum$city_ascii == "New York"]

Distance_pair(lat1, lon1, lat2, lon2)
```

```
## [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.

```
#looping


Distance_all_cities.df <- NULL

for (i in datum$city_ascii){
  lat2 <- datum$lat[datum$city_ascii == i]
  lon2 <- datum$long[datum$city_ascii == i]

  result <- Distance_pair(lat1, lon1, lat2, lon2)

  Distance_all_cities.df <- rbind.data.frame(Distance_all_cities.df, result)


}

Distance_all_cities.df
```

```
##    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
```

Q) 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
#Bonus

Distance_all_cities.df <- NULL

for (i in datum$city_ascii){
  lat2 <- datum$lat[datum$city_ascii == i]
  lon2 <- datum$long[datum$city_ascii == i]
```

```
  result <- Distance_pair(lat1, lon1, lat2, lon2)

  Combined <- data.frame(city1 = "Auburn", City2 = i, distance_km = result)   #important to specify the

  Distance_all_cities.df <- rbind.data.frame(Distance_all_cities.df, Combined)

}

Distance_all_cities.df
```

```
##     city1         City2 distance_km
## 1  Auburn      New York   1367.8540
## 2  Auburn   Los Angeles   3051.8382
## 3  Auburn       Chicago   1045.5213
## 4  Auburn         Miami    916.4138
## 5  Auburn       Houston    993.0298
## 6  Auburn        Dallas   1056.0217
## 7  Auburn  Philadelphia   1239.9732
## 8  Auburn       Atlanta    162.5121
## 9  Auburn    Washington   1036.9900
## 10 Auburn        Boston   1665.6985
## 11 Auburn       Phoenix   2476.2552
## 12 Auburn       Detroit   1108.2288
## 13 Auburn       Seattle   3507.9589
## 14 Auburn San Francisco   3388.3656
## 15 Auburn     San Diego   2951.3816
## 16 Auburn   Minneapolis   1530.2000
## 17 Auburn         Tampa    591.1181
## 18 Auburn      Brooklyn   1363.2072
## 19 Auburn        Denver   1909.7897
## 20 Auburn        Queens   1380.1382
## 21 Auburn     Riverside   2961.1199
## 22 Auburn     Las Vegas   2752.8142
## 23 Auburn     Baltimore   1092.2595
## 24 Auburn     St. Louis    796.7541
## 25 Auburn      Portland   3479.5376
## 26 Auburn   San Antonio   1290.5492
## 27 Auburn    Sacramento   3301.9923
## 28 Auburn        Austin   1191.6657
## 29 Auburn       Orlando    608.2035
## 30 Auburn      San Juan   2504.6312
## 31 Auburn      San Jose   3337.2781
## 32 Auburn  Indianapolis    800.1452
## 33 Auburn    Pittsburgh   1001.0879
## 34 Auburn    Cincinnati    732.5906
## 35 Auburn     Manhattan   1371.1633
## 36 Auburn   Kansas City   1091.8970
## 37 Auburn     Cleveland   1043.2727
## 38 Auburn      Columbus    851.3423
## 39 Auburn         Bronx   1382.3721
## 40 Auburn        Auburn      0.0000
```

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

Coding_challenge_6 Folder