Programming Assignment-Air Quality

Hariharan

This PDF contains the code and output generated for programming assignment 1 in the course Find the week_2_prog_assignment.Rmd in the same folder as this file to interact with the code and make changes for a better learninng experience

Overview

The zip file contains 332 comma-separated-value (CSV) files containing pollution monitoring data for fine particulate matter (PM) air pollution at 332 locations in the United States. Each file contains data from a single monitor and the ID number for each monitor is contained in the file name. For example, data for monitor 200 is contained in the file "200.csv". Each file contains three variables:

Date: the date of the observation in YYYY-MM-DD format (year-month-day) sulfate: the level of sulfate PM in the air on that date (measured in micrograms per cubic meter) nitrate: the level of nitrate PM in the air on that date (measured in micrograms per cubic meter)

For this programming assignment you will need to unzip this file and create the directory 'specdata'. Once you have unzipped the zip file, do not make any modifications to the files in the 'specdata' directory. In each file you'll notice that there are many days where either sulfate or nitrate (or both) are missing (coded as NA). This is common with air pollution monitoring data in the United States.

The data set can be found here: https://d396qusza40orc.cloudfront.net/rprog%2Fdata%2Fspecdata.zip

Part 1

The function 'pollutantmean' takes three arguments: 'directory', 'pollutant', and 'id'. Given a vector monitor ID numbers, 'pollutantmean' reads that monitors' particulate matter data from the directory specified in the 'directory' argument and returns the mean of the pollutant across all of the monitors, ignoring any missing values coded as NA.

```
pollutantmean <- function(directory, pollutant , id= 1:332){
  filelist<-list.files(path = directory , pattern = ".csv" , full.names = TRUE)
  #print(filelist)
  values <- numeric()
  for (i in id ){

    data <- read.csv(filelist[i])
    # values <- c(values , data[[pollutant]])
    values <- c(values , data[[pollutant]])
  }
  return(mean(values , na.rm = TRUE))
}
```

Sample usage

```
pollutantmean ("/Users/MAHE/Desktop/programming/Data Science/Swirl/Swirl_excercises/specdata","sulfate")
```

```
## [1] 3.189369
```

Part 2

Write a function that reads a directory full of files and reports the number of completely observed cases in each data file. The function should return a data frame where the first column is the name of the file and the second column is the number of complete cases.

```
complete <- function(directory, id=1:332){
  filelist<-list.files(path = directory , pattern = ".csv" , full.names = TRUE)
  results<- data.frame(id=numeric(),nobs= numeric(0))
  for(i in id){
    monitor_data <- read.csv(filelist[i])
    interested_data <- monitor_data[(!is.na(monitor_data$sulfate)), ]
    interested_data <- interested_data[(!is.na(interested_data$nitrate)), ]
    nobs <- nrow(interested_data)
    results <- rbind(results, data.frame(id=i,nobs=nobs))

  }
  return(results)
}
```

Sample usage

```
complete("specdata", 1)
```

```
##   id nobs
## 1  1  117
```

```
complete("specdata", c(2, 4, 8, 10, 12))
```

```
##   id nobs
## 1  2 1041
## 2  4  474
## 3  8  192
## 4 10  148
## 5 12   96
```

Part 3

Write a function that takes a directory of data files and a threshold for complete cases and calculates the correlation between sulfate and nitrate for monitor locations where the number of completely observed cases

(on all variables) is greater than the threshold. The function should return a vector of correlations for the monitors that meet the threshold requirement. If no monitors meet the threshold requirement, then the function should return a numeric vector of length 0.

```
corr <- function(directory, threshold = 0){
  cor_results <- numeric(0)

  complete_cases <- complete(directory)
  complete_cases <- complete_cases[complete_cases$nobs>=threshold, ]

  if(nrow(complete_cases)>0){
    for(monitor in complete_cases$id){
      path <- paste(getwd(), "/", directory, "/", sprintf("%03d", monitor), ".csv", sep = "")
      #print(path)
      monitor_data <- read.csv(path)
      #print(monitor_data)
      interested_data <- monitor_data[(!is.na(monitor_data$sulfate)), ]
      interested_data <- interested_data[(!is.na(interested_data$nitrate)), ]
      sulfate_data <- interested_data["sulfate"]
      nitrate_data <- interested_data["nitrate"]
      cor_results <- c(cor_results, cor(sulfate_data, nitrate_data))
    }
  }
  return(cor_results)
}
```

Sample Usage

```
cr <- corr("specdata", 400)
head(cr)
```

```
## [1] -0.01895754 -0.04389737 -0.06815956 -0.07588814  0.76312884 -0.15782860
```

Quiz

1. What value is returned by the following call to pollutantmean()? You should round your output to 3 digits.

```
pollutantmean("specdata", "sulfate", 1:10)
```

```
## [1] 4.064128
```

2. What value is returned by the following call to pollutantmean()? You should round your output to 3 digits.

```
pollutantmean("specdata", "nitrate", 70:72)
```

```
## [1] 1.706047
```

3. What value is returned by the following call to pollutantmean()? You should round your output to 3 digits.

```
pollutantmean("specdata", "sulfate", 34)
```

```
## [1] 1.477143
```

4.What value is returned by the following call to pollutantmean()? You should round your output to 3 digits.

```
pollutantmean("specdata", "nitrate")
```

```
## [1] 1.702932
```

5.What value is printed at end of the following code?

```
cc <- complete("specdata", c(6, 10, 20, 34, 100, 200, 310))
print(cc$nobs)
```

```
## [1] 228 148 124 165 104 460 232
```

6. What value is printed at end of the following code?

```
cc <- complete("specdata", 54)
print(cc$nobs)
```

```
## [1] 219
```

7.What value is printed at end of the following code?

```
RNGversion("3.5.1")
```

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
set.seed(42)
cc <- complete("specdata", 332:1)
use <- sample(332, 10)
print(cc[use, "nobs"])
```

```
##  [1] 711 135  74 445 178  73  49   0 687 237
```

8.What value is printed at end of the following code?

```
cr <- corr("specdata")
cr <- sort(cr)
RNGversion("3.5.1")
```

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
set.seed(868)
out <- round(cr[sample(length(cr), 5)], 4)
print(out)
```

```
## [1]  0.2688  0.1127 -0.0085  0.4586  0.0447
```

9.What value is printed at end of the following code?

```
cr <- corr("specdata", 129)
cr <- sort(cr)
n <- length(cr)
RNGversion("3.5.1")
```

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
set.seed(197)
out <- c(n, round(cr[sample(n, 5)], 4))
print(out)
```

```
## [1] 243.0000   0.2540   0.0504  -0.1462  -0.1680   0.5969
```

10.What value is printed at end of the following code?

```
cr <- corr("specdata", 2000)
n <- length(cr)
cr <- corr("specdata", 1000)
cr <- sort(cr)
print(c(n, round(cr, 4)))
```

```
## [1]  0.0000 -0.0190  0.0419  0.1901
```