Air Pollution - Data Analysis

Arin Parsa

6/3/2020

Contents

```r
# Print R Version
print(R.version.string)
```

```
## [1] "R version 4.0.0 (2020-04-24)"
```

Description

The air pollution data analysis assignment from JHU is for learning how to read multiple csv files from a directory, load them into data frames, and calculating mean and correlation.

Specdata.zip

The specdata.zip file in the data folder in this repo 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 measurement
- nitrate: the level of nitrate PM in the air on that date (measured in micrograms per cubic meter)

Unzip specdata.zip either manually or through the command below.

Unzipping the zip file should create a specdata directory that contains all 332 csv files Make sure to change eval=TRUE in the section below if you are running this markdown file. It has been set to FALSE for repeated runs.

```r
unzip("specdata.zip")
```

Verify 332 .csv files were unzipped in the newly created "specdata" directory

```
dir.exists("data/specdata") #This should return TRUE
```

```
## [1] TRUE
```

```
length(list.files("data/specdata")) #This should return 332
```

```
## [1] 332
```

```
#list.files("specdata", full.names = TRUE)
```

In a given file, calculate mean of sulphate levels excluding NA values

```
file_1 <- read.csv("data/specdata/001.csv")
mean(file_1$sulfate, na.rm = TRUE)
```

```
## [1] 3.880701
```

Create a function named 'pollutantmean' that calculates the mean of a pollutant (sulfate or nitrate) across a specified list of monitors

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) {
  ## 'directory' is a character vector of length 1 indicating
  ## the location of the CSV files

  ## 'pollutant' is a character vector of length 1 indicating
  ## the name of the pollutant for which we will calculate the
  ## mean; either "sulfate" or "nitrate".

  ## 'id' is an integer vector indicating the monitor ID numbers
  ## to be used

  ## We need to return the mean of the pollutant across all monitors list
  ## in the 'id' vector (ignoring NA values)

  ## Load files into data frames

  files <- list.files(directory, full.names = TRUE)
  df_list <- lapply(files[id], read.csv)

  # Alternative way of loading files into a single data frame using a for loop
  # as opposed to lapply

  # files <- list.files(directory, full.names = TRUE)
  # for (i in 1:length(id)) {
  #   df <- read.csv(files[id[i]])
  #   single_df <- rbind(single_df, df)
  # }
  ## Merging all data frames into one

  airpollution_df <- data.frame(matrix(vector(), 0, 4))

  for (i in 1:length(df_list)) {
    airpollution_df <- rbind(airpollution_df, df_list[[i]])
  }

  names(airpollution_df) <- c("Date","sulfate","nitrate","ID")

  #Alternate way of rbinding using lapply followed by do.call

  #one_df_list <- lapply(df_list, rbind)
  #single_df <- do.call(rbind.data.frame, one_df_list)

  if (pollutant == "sulfate") {
    mean(airpollution_df$sulfate, na.rm = TRUE)
  } else if (pollutant == "nitrate") {
    mean(airpollution_df$nitrate, na.rm = TRUE)
  }



}
```

Calling polluntantmean function for different scenarios

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

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

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

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

```
pollutantmean("data/specdata", "nitrate", 23)
```

```
## [1] 1.280833
```

Create a function "complete" to return a data frame of ID and its corresponding complete cases

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) {
  ## 'directory' is a character vector of length 1 indicating
  ## the location of the CSV files

  ## 'pollutant' is a character vector of length 1 indicating
  ## the name of the pollutant for which we will calculate the
  ## mean; either "sulfate" or "nitrate".

  ## 'id' is an integer vector indicating the monitor ID numbers
  ## to be used

  # Create an empty data frame with columns of names id and nobs
  single_df <- data.frame(matrix(vector(), 0, 2))
  names(single_df) <- c("id", "nobs")
  # Get file names relevant files from the directory
  files <- list.files(directory, full.names = TRUE)
  # For each file name, read.csv into dataframe for each id
  #assume you're doing id=3:5
  for (i in 1:length(id)) {
    df <- read.csv(files[id[i]])
    # Find nobs: number of complete cases (no NA values) for each dataframe
    nobs <- sum(complete.cases(df))
    # Binding the id and nobs to the data frame created to hold these values
    single_df <- rbind(single_df, c(id[i], nobs))
  }


  names(single_df) <- c("id", "nobs")

  return(single_df)

}
```

Calling the complete function with various monitor IDs

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

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

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

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

```
complete("data/specdata", 30:25)
```

```
##   id nobs
## 1 30  932
## 2 29  711
## 3 28  475
## 4 27  338
## 5 26  586
## 6 25  463
```

```
complete("data/specdata", 3)
```

```
##   id nobs
## 1  3  243
```

Create a function to calculate the correlation between sulfate and nitrate for monitor locations where the number of completely observed cases (on all variables) is greater than the threshold

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

For this function you will need to use the 'cor' function in R which calculates the correlation between two vectors. Please read the help page for this function via '?cor' and make sure that you know how to use it.

```r
 corr <- function(directory, threshold = 0) {
  # Load all files into data frames (list.files, list of 332 dataframes)
  files <- list.files(directory, full.names = TRUE)
  df_list <- lapply(files, read.csv)
  cor_vector = c()
  # For each id, compare the number of complete cases of all variables to the threshold value
  for (i in 1:length(files)) {
    df <- na.omit(df_list[[i]])
    # If the nobs is greater than the threshold, then compute the correlation of the sulfate and nitrat
    nobs <- sum(complete.cases(df))
    # print(nobs)
    # browser()
    if (nobs > threshold[1]) {
      correlation <- cor(df$sulfate, df$nitrate)
      # Put the correlation in a vector
      cor_vector <- c(cor_vector, correlation)
    }
  }
  # Return a vector of correlations for the monitors that meet the threshold requirement
  # summary(cor_vector)
  # length(cor_vector)
  return (cor_vector)
}
```

Calling the corr function with sample threhold value

```r
cr <- corr("data/specdata", 150)
head(cr)
```

```
## [1] -0.01895754 -0.14051254 -0.04389737 -0.06815956 -0.12350667 -0.07588814
```

```r
summary(cr)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.21057 -0.04999  0.09463  0.12525  0.26844  0.76313
```

Assignment questions from R Programming Course by John Hopkins Univ., Week 2

```r
pollutantmean("data/specdata", "sulfate", 1:10)
```

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

```r
pollutantmean("data/specdata", "nitrate", 70:72)
```

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

```r
pollutantmean("data/specdata", "sulfate", 34)
```

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

```r
pollutantmean("data/specdata", "nitrate")
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```r
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
```

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

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