R_Programming Project 1

Andrew Abisha Hu

Introduction:

For this first programming assignment you will write three functions that are meant to interact with dataset that accompanies this assignment. The dataset is contained in a zip file specdata.zip that you can download from the Coursera web site.

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)

Key takeaways:

The point of this assignment is to know how to do a **simple "for loop"", combine all the files into a same file, subset the data with conditions**, and then return the results.

One of the key step is to create a blank vector first, and then use the "for-loop" to duplicate numeric values into that blank vector

Here is a practice:

```
nums <- c(2,4,6,8)
dup <- numeric()

dup #the blank one, nothing is in there yet.
```

```
## numeric(0)
```

```
for (i in nums) {
        dup<- c(dup,nums) #the new dup=  the original "blank "dup + nums

}

dup # now it contains c(2,4,6,8) * 4 times!
```

```
##  [1] 2 4 6 8 2 4 6 8 2 4 6 8 2 4 6 8
```

Part 1:

Write 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

```
#part 1
setwd("/Users/andrewhu/Documents/GitHub/Coursera-DataScienceSpecialization/02_RProgramming")
pollutantmean <- function(directory, pollutant, id= 1:332) {

        filelist<- list.files(path=directory, pattern= ".csv", full.names=TRUE)#Read the list of files,

         values<- numeric() #Create a blank numeric vector, before the for-loop

        for (i in id){
        data<-read.csv(filelist[i]) # read the filelist
        values <- c(values, data[[pollutant]])   # add the value into the blank vector you created befo
        }
        mean(values,na.rm=TRUE) #return the mean of the values

}

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

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

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

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

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

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

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

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

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

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

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

```
#part2

#Practice: calculate the complete cases in a file.
data<- read.csv("specdata/001.csv")
sum(complete.cases(data)) #117 complete cases in 001.csv
```

```
## [1] 117
```

```
#Checking the length of filelist
filelist<- list.files(path="specdata", pattern=".csv",full.names = TRUE)
length(filelist) # check to get all the files
```

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

```
complete<- function(directory, id= 1:332) {
        filelist<-list.files(path=directory,pattern=".csv", full.names = TRUE) #list all the files
```

```
        nobs<- numeric() #create an empty vector

        for (i in id) {
        data<- read.csv(filelist[i]) #data=read how many files (which file)
        nobs<- c(nobs,sum(complete.cases(data))) #add value into the original blank vector, nobs

        }
        nobs #return the nobs
        data.frame(id,nobs) #Create a simple df
}

complete("specdata",1:10)
```

```
##    id nobs
## 1   1  117
## 2   2 1041
## 3   3  243
## 4   4  474
## 5   5  402
## 6   6  228
## 7   7  442
## 8   8  192
## 9   9  275
## 10 10  148
```

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

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

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

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

```
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
```

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. A prototype of this function follows

```
#partIII
corr<- function(directory, threshold=0) {

        filelist= list.files(path =directory, pattern =".csv", full.names = TRUE )#create a filelist
        dat<- numeric() # blank numeric vector
```

```
        for (i in 1:length(filelist)) {
        temp<- read.csv(filelist[i]) #reading files ~ xth file
        temp<- temp[complete.cases(temp),] #subsetting complete cases
        nrow<- nrow(temp) #counting the rows
        if (nrow > threshold) {
        dat<- c(dat, cor(temp$sulfate,temp$nitrate))    } #if nrow> threshold then return the correlati

        }
        dat # return the result
}

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

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

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

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

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

```
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
```