Programming Assignment 1: Air Pollution: Instructions

Help Center

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.

Details about grading and submission procedures can be found near the bottom of this page.

Data

The zip file containing the data can be downloaded here:

- specdata.zip [2.4MB]

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.

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. A prototype of the function is as follows

```
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
```

```
        ## Return the mean of the pollutant across all monitors list
        ## in the 'id' vector (ignoring NA values)
}
```

You can see some example output from this function. The function that you write should be able to match this output. Please save your code to a file named **pollutantmean.R**.

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. A prototype of this function follows

```
complete <- function(directory, id = 1:332) {
        ## 'directory' is a character vector of length 1 indicating
        ## the location of the CSV files

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

        ## Return a data frame of the form:
        ## id nobs
        ## 1  117
        ## 2  1041
        ## ...
        ## where 'id' is the monitor ID number and 'nobs' is the
        ## number of complete cases
}
```

You can see some example output from this function. The function that you write should be able to match this output. Please save your code to a file named **complete.R**. To run the submit script for this part, make sure your working directory has the file **complete.R** in it.

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

```
corr <- function(directory, threshold = 0) {
        ## 'directory' is a character vector of length 1 indicating
        ## the location of the CSV files

        ## 'threshold' is a numeric vector of length 1 indicating the
        ## number of completely observed observations (on all
        ## variables) required to compute the correlation between
        ## nitrate and sulfate; the default is 0
```

```
        ## Return a numeric vector of correlations
}
```

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.
You can see some example output from this function. The function that you write should be able to match this output. Please save your code to a file named **corr.R**. To run the submit script for this part, make sure your working directory has the file **corr.R** in it.

Grading and Submission

This assignment will be graded using unit tests executed via the submit script you run on your computer. To obtain the submit script, run the following code in R:

```
source("http://d396qusza40orc.cloudfront.net/rprog%2Fscripts%2Fsubmitscript1.R")
```

Or you can download the script to your working directory (NOTE: you may need to rename the file to be "submitscript1.R". Then source the file locally via

```
source("submitscript1.R")
```

The first time you run the submit script it will prompt you for your Submission login and Submission password. These can be found at the top of the Programmi ng Assignments page. To execute the submit script, type

```
submit()
```

at the console prompt (after source-ing the file). **NOTE that the submit script requires that you be connected to the Internet in order to work properly**. When you execute the `submit` script in R, you will see the following menu (after typing in your submission login email and password):

```
[1] 'pollutantmean' part 1
[2] 'pollutantmean' part 2
[3] 'pollutantmean' part 3
[4] 'pollutantmean' part 4
[5] 'complete' part 1
[6] 'complete' part 2
[7] 'complete' part 3
[8] 'corr' part 1
[9] 'corr' part 2
[10] 'corr' part 3
Which part are you submitting [1-10]?
```

We will compare the output of your functions to the correct output. For each test passed you receive the specified number of points on the Assignments List web page.

You are finished when you have successfully submitted everything using submit() and you see scores on the assignment page. You can ignore the Submit buttons to the right of each score. They are only to be used when firewall or proxy settings prevent users from successfully using the submit() script. The submit() script will describe how to create files for uploading if there are problems, but under normal circumstances, there is NO NEED to use the Submit buttons on the assignment page.