Computing for Data Analysis

Johns Hopkins Bloomberg School of Public Health

Computing for Data Analysis

[1]

Programming Assignment 1 — View Assignment

Assignment Name	Programming Assignment 1
Due Date	Mon 8 Oct 2012 11:59:00 PM PDT (1 late day used) *Originally Sun 7 Oct 2012 11:59:00 PM PDT*
Hard Deadline	Sun 14 Oct 2012 11:59:00 PM PDT
Submission	Go to Assignments List [2] page to submit your solutions.
Instructions	

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 containing the data can be downloaded here:

- specdata.zip [3] [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 'getmonitor' that takes three arguments: 'id', 'directory', and 'summarize'. Given a monitor ID number, 'getmonitor' reads that monitor's particulate matter data from the directory specified in the 'directory' argument and returns a data frame containing that monitor's data. If 'summarize = TRUE', then 'getmonitor' produces a summary of the data frame with the 'summary' function and prints it to the console. A prototype of the function is as follows

```
getmonitor <- function(id, directory, summarize = FALSE) {
        ## 'id' is a vector of length 1 indicating the monitor ID
        ## number. The user can specify 'id' as either an integer, a
        ## character, or a numeric.

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

        ## 'summarize' is a logical indicating whether a summary of
        ## the data should be printed to the console; the default is
        ## FALSE

        ## Your code here
}
```

You can see some example output from this function [4]. The function that you write should be able to match this output. Please save your code to a file named **getmonitor.R**. To run the test script for this part, make sure your working directory has the file **getmonitor.R** in it and the run

```
source("http://spark-public.s3.amazonaws.com/compdata/scripts/getmonitor-test.R")
getmonitor.testscript()
```

Afterwards, upload the output files on the Assignments List page.

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 [5]. 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 test script for this part, make sure your working directory has the file **complete.R** in it and the run

```
source("http://spark-public.s3.amazonaws.com/compdata/scripts/complete-test.R")
complete.testscript()
```

Afterwards, upload the output files on the Assignments List page.

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 [6]. 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 test script for this part, make sure your working directory has the file **corr.R** in it and the run

```
source("http://spark-public.s3.amazonaws.com/compdata/scripts/corr-test.R")
corr.testscript()
```

Afterwards, upload the output files on the Assignments List page.

Grading

This assignment will be graded using unit tests executed via the test scripts you run on your computer. 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. For each function in this assignment there will be 10 points to earn, for a total of 30 points for the entire assignment.

1. https://class.coursera.org/compdata-2012-001/class/index
2. https://class.coursera.org/compdata-2012-001/assignment/list
3. http://spark-public.s3.amazonaws.com/compdata/data/specdata.zip
4. http://spark-public.s3.amazonaws.com/compdata/documents/getmonitor-output.pdf
5. http://spark-public.s3.amazonaws.com/compdata/documents/complete-output.pdf
6. http://spark-public.s3.amazonaws.com/compdata/documents/corr-output.pdf