Pollutant Substance

2025-08-26

R Programming

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.

Although this is a programming assignment, you will be assessed using a separate quiz.

The zip file containing the data can be downloaded here: specdata.zip [2.4MB] Description: The zip file contains 332 comma-separated-value (CSV) files containing pollution monitoring data.

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.

```
library(data.table)
unzip("rprog-data-specdata.zip", exdir = "./R Programming", overwrite = TRUE)
pollutantmean <- function(directory, pollutant, id = 1:332) {
        obs_data <- function(monitor_id, directory, path = "R Programming") {
        files_id <- paste0(stringr::str_pad(string = monitor_id, width = 3, pad = "0"), ".csv")
                dt_frame <- read.csv(file.path(path, directory, files_id))
        return(dt_frame)
}
if(length(id) == 1) {
        if( id <= 0 | id > length(list.files(path = "R Programming/specdata"))) {
                return(print("Please check your input id. Should be between 1 and 332"))
}
        else {
        dt_frame0 <- obs_data(monitor_id = id, directory = directory, path = "R Programming")
        return(mean(x = dt_frame0[[pollutant]], na.rm = TRUE))
        }
}
        else {
                if(sum(id > 332 | id <= 0) > 0)
                        return(print("Please, check your id input. Should be between 1 and 332"))
        }
        if(length(id) > 1 & length(id) <= length(list.files(path = "R Programming/specdata"))) {
                dt_frame0 <- data.frame()
                for (i in id) {
                        dt_frame0 <- rbind(dt_frame0,
                        obs_data(monitor_id = i, directory = directory, path = "R Programming"))
                return(mean(x = dt_frame0[[pollutant]], na.rm = TRUE))
        }
    }
}

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

```
## [1] 0.2551667
```

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.

```
## Create the function complete, load and read the csv files
complete <- function(directory, id = 1:332){
        obs_data <- function(monitor_id, directory, path = "R Programming") {
                file_id <- paste0(stringr::str_pad(string = monitor_id, width = 3,
                                                   pad = "0"), ".csv" )
                data_frame <- read.csv(file = file.path(path, directory, file_id))
                return(data_frame)
        }
        if (length(id) == 1) {
               if (id <= 0 | id > length(list.files(path = "R Programming/specdata"))) {
                       return(print("Please, check your input to ID. Should be between 1 and 332"))
        }
        else {
                data_frame0 <- obs_data(monitor_id = id, directory = directory,
                                        path = "R Programming")
                dataf_clean <- na.omit(data_frame0)
                nobs <- nrow(dataf_clean)
                return(data.frame(id, nobs))
        }
}
else {
        if (sum(id > 332 | id <= 0) > 0) {
                return(print("Please, check your input to ID.
                             Should be between 1 and 332"))
        }

        if ( length(id) > 1 & length (id) <= length(list.files(path =
        "R Programming/specdata"))) {
                data_frame0 <- data.frame()
                nobs <- vector()
                # for each .csv file in id
                for (i in id) {
                        data_frame0 <- obs_data(monitor_id = i,
                   directory = directory, path = "R Programming")
                   dataf_clean <- na.omit(data_frame0)
                   nobs <- append(nobs, nrow(dataf_clean))
                }
                return(data.frame(id, nobs))
        }
    }
}

##Example:
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) {
        t_nobs <- complete(directory, 1:332)
        t_threshold <- t_nobs[t_nobs$nobs > threshold, ]
        obs_files <- function(monitor_id, directory, path = "R Programming") {
                file_id <- paste0(stringr::str_pad(string = monitor_id, width = 3,
                                                   pad = "0"), ".csv")
        data_frame <- read.csv(file = file.path(path, directory, file_id))
        return(data_frame)
        }
        correlation <- vector(mode = "numeric")
        for (j in  t_threshold$id) {
                dataf_clean <- na.omit(obs_files(monitor_id = j, directory = directory,
                                path = "R Programming"))
                correlation <- append(correlation, cor(dataf_clean$sulfate,
                                                       dataf_clean$nitrate))
        }
        t_threshold['corr'] <- correlation
        return(correlation)
}

#Example:
dt <- corr("specdata", 150)
head(dt)
```

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