R Programming Course - Week 2 Programming Assignment

```
pollutantmean <- function(directory, pollutant, id=1:332) {

  # Create a list of files in the directory argument
  files_list <- list.files(directory, full.names = TRUE)
  df <- data.frame() #creates an empty data frame

  # Loop through the files, rbinding them together
  for (i in id) {
    df <- rbind(df, read.csv(files_list[i]))
  }

  # Subset the column that matches the 'pollutant'  argument
  df_subset <- df[,pollutant]

  # Calculate the mean value
  mean(df_subset, na.rm = TRUE)
}
```

```
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
```

```
complete <- function(directory,id=1:332) {

  # Create a list of files in the directory argument
  files_list <- list.files(directory, full.names = TRUE)
  df1 <- data.frame() #creates an empty data frame

  # Loop through the files, rbinding them together
  for (i in id) {
    df1 <- rbind(df1, read.csv(files_list[i]))
  }

  # Filter out the rows where 'NA' shows up in either the sulfate or nitrate columns
  df1 <- df1[ !is.na(df1$sulfate & df1$nitrate), ]

  # Initialize the row index of the output dataframe
  running.index <- 0
```

```
  # Set up the 'container' vectors for 'monitorid' and 'nobs'
  monitorid <- numeric(length(id))
  nobs <- numeric(length(id))

  for (i in id) {
    running.index <- running.index+1
    monitorid[running.index] <- i
    nobs[running.index] <- sum(df1$ID == i)
  }

  monitorid <- monitorid[1:running.index]
  nobs <- nobs[1:running.index]

  # Create a dataframe from the 'monitorid' and 'nobs' vectors.
  result <- data.frame(monitorid,nobs)
}
```

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

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

```
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
```

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

  # Make sure the R data.table package is loaded
  require(data.table)

  # Reading in all files and making a large data.table
  files_list <- lapply(file.path(directory, list.files(path = directory,pattern ="*.csv")), data.table:

  # Bind all of the csv files together into one big file.
  dt <- rbindlist(files_list)

  # Only keep the completely observed cases
  dt <- dt[complete.cases(dt),]

  # Apply threshold and calculate correlations for those IDs whose nobs exceed the threshold
  dt <- dt[, .(nobs = .N, corr = cor(x = sulfate, y = nitrate)), by = ID][nobs > threshold]

  return(dt[, corr])
}
```

```
v = corr("specdata",150)
```

```
## Loading required package: data.table
```

```
v
```

```
##   [1] -0.018957541 -0.140512544 -0.043897372 -0.068159562 -0.123506666
##   [6] -0.075888144 -0.159673652 -0.086841940  0.763128837 -0.157828603
```

```
##  [11] -0.156998919 -0.044898818  0.117249264  0.259057178  0.133274607
##  [16]  0.366201078  0.580751264  0.006863930  0.726693888  0.057741676
##  [21]  0.115338086  0.465754012  0.515804375  0.412693537  0.375631176
##  [26]  0.315725317  0.244560561  0.594426499  0.553514976  0.614340566
##  [31]  0.460513619  0.405022501  0.434789780  0.088421364  0.118136697
##  [36] -0.091022820 -0.033091304  0.440660466 -0.029683708  0.268525390
##  [41]  0.277220958 -0.049108453  0.322627410  0.091139374 -0.025750053
##  [46]  0.120521602 -0.061746831  0.041306963 -0.146202136 -0.162485185
##  [51] -0.097254393  0.089262856  0.568403991  0.711864008  0.268203237
##  [56]  0.190644585  0.227222983  0.229238882  0.005635506  0.018628108
##  [61] -0.064750174  0.096614297  0.002864405  0.107184775  0.128477284
##  [66] -0.042533572 -0.137041337  0.136609030  0.118975253  0.098073855
##  [71]  0.066928310  0.100212474 -0.063984344 -0.066525489 -0.129245884
##  [76] -0.111066409 -0.089441210 -0.114090325 -0.106280702 -0.176855164
##  [81] -0.116984680  0.019138583  0.100643502 -0.073858484  0.036665921
##  [86] -0.107957809  0.296744105  0.347421569  0.146528765  0.362414577
##  [91]  0.093330832  0.198915192  0.164602262  0.180626975  0.176508543
##  [96]  0.139158631  0.231984399  0.227615918  0.275903634  0.299630040
## [101]  0.248143145  0.298344178 -0.056325366 -0.178114558  0.002032940
## [106] -0.022802183 -0.001202233  0.085217423 -0.076409023  0.010021716
## [111]  0.016411646 -0.038785934 -0.075297768  0.041917773  0.193324040
## [116]  0.596929143  0.113596590 -0.143750037 -0.017703373  0.284905360
## [121]  0.305506111  0.150031306  0.134895077  0.172850003  0.286076203
## [126] -0.106687748  0.244744168  0.337120085  0.424798956  0.095921881
## [131]  0.022899033  0.143330735  0.087196218  0.408741028  0.425176879
## [136]  0.361728434 -0.035090337 -0.082388453 -0.094742313 -0.087573726
## [141] -0.060405837 -0.092398269 -0.183197353  0.124650112 -0.053001162
## [146] -0.039911536  0.010158287  0.451828854  0.295793699  0.615268727
## [151] -0.075214053  0.132207405  0.089547098 -0.019086127 -0.045552626
## [156]  0.211599525 -0.073972834  0.112668377  0.138387891 -0.003207550
## [161] -0.052643174  0.042168144 -0.067460173 -0.030882797  0.017805647
## [166]  0.026138073 -0.050287543  0.016535643  0.199919014  0.482158286
## [171]  0.355110474  0.589606340  0.368038099 -0.029094866 -0.074495323
## [176]  0.262101561 -0.005386993  0.258826380  0.144110820  0.101915017
## [181]  0.023020993  0.074594252  0.256665139  0.162401158 -0.003454405
## [186]  0.190141976  0.184581239  0.120596460 -0.176233152 -0.144699131
## [191]  0.147074115  0.273520382  0.109557323 -0.092863394 -0.182752126
## [196] -0.008836513  0.356592359 -0.089133895 -0.017185129 -0.156323514
## [201] -0.042538204  0.010235676 -0.009912754 -0.042910367 -0.210567709
## [206] -0.155957816  0.046211272 -0.060808231  0.160865053  0.615095781
## [211]  0.598343330  0.506535631  0.191834811 -0.024723462 -0.150627164
## [216] -0.002500089 -0.166201361  0.619349867  0.531380642  0.520115665
## [221]  0.466673962  0.518820173  0.394191512  0.379446208 -0.123172036
## [226] -0.061565518 -0.180133963  0.253978075  0.139867175  0.316429404
## [231]  0.268780500  0.279397143  0.267260662  0.287133842
```

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

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

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

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

```
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
```