

```
Assignment1_Part1.R x
Source on Save
1
2
3 pollutantmean <- function(directory, pollutant, id = 1:10) {
4
5     directory <- getwd()
6     files <- list.files(directory, pattern="*.csv", full.names=FALSE)
7     merged = NULL
8
9     for (i in id) {
10         data <- read.csv(file=files[i], header = TRUE)
11         merged <- rbind(merged,data)
12         sulfate <- subset(merged, sulfate != "NA")
13         nitrate <- subset(merged, nitrate != "NA")
14     }
15     if(pollutant == "sulfate") {
16         meanpollutant <- sum(sulfate$sulfate)/nrow(sulfate)
17     }
18     else if (pollutant == "nitrate") {
19         meanpollutant <- sum(nitrate$nitrate)/nrow(nitrate)
20     }
21     return(meanpollutant)
22 }
23
```

```
Assignment1_Part1.R x Assignment1_Part2.R x
Source on Save
1
2 complete <- function(directory, id=1:332) {
3     directory <- getwd()
4     files <- list.files(directory, pattern="*.csv", full.names=FALSE)
5     merged = NULL
6
7     for (i in id) {
8         data <- read.csv(file=files[i], header = TRUE)
9         merged <- rbind(merged,data)
10        complete <- subset(merged, sulfate != "NA" & nitrate != "NA")
11        counts <- data.frame(table(complete$ID))
12        names(counts) <- c("id","nobs")
13    }
14    {
15        return(counts)
16    }
17
18 }
```

```
Assignment1_Part1.R x Assignment1_Part2.R x Assignment1_Part3.R x
Source on Save
1 forr <- function(directory, threshold=0) {
2   directory <- getwd()
3   files <- list.files(directory, pattern="*.csv", full.names=FALSE)
4   merged = NULL
5   cvect = NULL
6
7   for (i in 1:332) {
8     data <- read.csv(file=files[i], header = TRUE)
9
10    data <- subset(data, sulfate != "NA" & nitrate != "NA")
11
12    if (nrow(data)>threshold){
13      cvect = c(cvect,cor(data$sulfate, data$nitrate))
14    }
15
16  }
17
18  return(cvect)
19
20 }
21 }
```

```
cachematrix.R x
Source on Save Run
1
2 ## This program caches inverse of matrix and creates inverse if absent
3
4 ## makeCacheMatrix: This function creates a special "matrix" object that can cache its inverse.
5
6 makeCacheMatrix <- function(x = matrix()) {
7
8   m <- NULL
9   set <- function(y) {
10     x <- y
11     m <- NULL
12
13   }
14   get <- function() x
15   setinv <- function(inverse) m <- inverse
16   getinv <- function() m
17   list(set=set, get=get, setinv=setinv, getinv=getinv)
18 }
19
20
21 ## cacheSolve: This function computes the inverse of the special "matrix" returned by makeCacheMatr
22 ## If the inverse has already been calculated (and the matrix has not changed), then the cacheSolve
23
24 cacheSolve <- function(x, ...) {
25   ## Return a matrix that is the inverse of 'x'
26   m <- x$getinv()
27
28   if (!is.null(m)) {
29     message("getting cached data")
30     return(m)
31   }
32
33   mx.data <- x$get()
34   m <- solve(mx.data, ...)
35   x$setinv(m)
36
37   return(m)
38 }
```

```

1 rankhospital <- function(state, outcome, num="best") {
2
3   ocm <- read.csv("outcome-of-care-measures.csv", colClasses = "character", na.strings = "Not Available")
4   vstates <- unique(c(ocm$state))
5   testst = grep(state, vstates)
6   testout = grep(outcome, c("heart attack", "heart failure", "pneumonia"))
7   #Subset required variables - smaller dataset
8   cb <- cbind(ocm[,2],ocm[,7],ocm[,11],ocm[,17],ocm[,23])
9
10  #Rename variable names for easier handling
11  names(cb) <- c("hname","state", "hattack", "hfailure","pneumonia")
12
13  if(!state %in% outsub1[, "state"]) {
14    stop('invalid state')
15  }
16  if(!outcome %in% c("heart attack", "heart failure", "pneumonia")){
17    stop('invalid outcome')
18  }
19
20  if(testout == "1"){
21    cb1 <- cb[, c(1,2,3)]
22    cb1 <- subset(cb1, state == vstates[testst])
23    cb1[,3] <- suppresswarnings(as.numeric(cb1[,3]))
24    cb1 <- na.omit(cb1)
25    nc <- nrow(cb1)
26    if(num == "best"){
27      ret <- best(state, outcome)
28    }
29    else if(num == "worst"){
30      num = nc
31      ord <- cb1[order(cb1$hattack, cb1$hname),]
32      rank <- rank(ord$hattack, na.last=TRUE, ties.method = "first")
33      srt <- cbind(ord,rank)
34      out <- srt[order(srt$rank, srt$hname), ]
35      ret <- out[out$rank == num, 1]
36    }
37  }
38  else if(num > nc){
39    stop(print("NA"))
40  }
41  else if(num > 0){
42    ord <- cb1[order(cb1$hattack, cb1$hname),]
43    rank <- rank(ord$hattack, na.last=TRUE, ties.method = "first")
44    srt <- cbind(ord,rank)
45    out <- srt[order(srt$rank, srt$hname), ]
46    ret <- out[out$rank == num, 1]
47  }
48  }
49  if(testout == "2"){
50    cb1 <- cb[, c(1,2,4)]

```

```

50    cb1 <- subset(cb1, state == vstates[testst])
51    cb1[,3] <- suppresswarnings(as.numeric(cb1[,3]))
52    cb1 <- na.omit(cb1)
53    nc <- nrow(cb1)
54    if(num == "best"){
55      ret <- best(state, outcome)
56    }
57    else if(num == "worst"){
58      num = nc
59      ord <- cb1[order(cb1$hfailure, cb1$hname),]
60      rank <- rank(ord$hfailure, na.last=TRUE, ties.method = "first")
61      srt <- cbind(ord,rank)
62      out <- srt[order(srt$rank, srt$hname), ]
63      ret <- out[out$rank == num, 1]
64    }
65    else if(num > nc){
66      stop(print("NA"))
67    }
68    else if(num > 0){
69      ord <- cb1[order(cb1$hfailure, cb1$hname),]
70      rank <- rank(ord$hfailure, na.last=TRUE, ties.method = "first")
71      srt <- cbind(ord,rank)
72      out <- srt[order(srt$rank, srt$hname), ]
73      ret <- out[out$rank == num, 1]
74    }
75  }
76  if(testout == "3"){
77    cb1 <- cb[, c(1,2,5)]
78    cb1 <- subset(cb1, state == vstates[testst])
79    cb1[,3] <- suppresswarnings(as.numeric(cb1[,3]))
80    cb1 <- na.omit(cb1)
81    nc <- nrow(cb1)
82    if(num == "best"){
83      ret <- best(state, outcome)
84    }
85    else if(num == "worst"){
86      num = nc
87      ord <- cb1[order(cb1$pneumonia, cb1$hname),]
88      rank <- rank(ord$pneumonia, na.last=TRUE, ties.method = "first")
89      srt <- cbind(ord,rank)
90      out <- srt[order(srt$rank, srt$hname), ]
91      ret <- out[out$rank == num, 1]
92    }
93    else if(num > nc){
94      stop(print("NA"))
95    }
96    else if(num > 0){
97      ord <- cb1[order(cb1$pneumonia, cb1$hname),]
98      rank <- rank(ord$pneumonia, na.last=TRUE, ties.method = "first")
99      srt <- cbind(ord,rank)
100     out <- srt[order(srt$rank, srt$hname), ]
101     ret <- out[out$rank == num, 1]
102   }
103 }
104 return(ret)
105 }

```

```

1- rankall <- function(outcome, num="best") {
2-   library(data.table)
3-   ocm <- read.csv("outcome-of-care-measures.csv", colClasses = "character", na.strings = "Not Available")
4-   testout = grep(outcome, c("heart attack", "heart failure", "pneumonia"))
5-
6-   cb <- cbind(ocm[,ocm[7]],ocm[,11],ocm[,17],ocm[,23]) #Subset with required variables
7-   names(cb) <- c("hname", "state", "hattack", "hfailure", "pneumonia") #Rename variable names
8-
9-   state <- unique(cb$state)
10-  states <- data.frame(state, state)
11-
12-  if(!outcome %in% c("heart attack", "heart failure", "pneumonia")){
13-    stop('invalid outcome')
14-  }
15-
16-  if(testout == "1"){
17-    cb1 <- cb[, c(1,2,3)]
18-    cb1[,3] <- suppressWarnings(as.numeric(cb1[,3]))
19-    cb1 <- na.omit(cb1)
20-    cb1 <- cb1[order(cb1$state, cb1$hname),]
21-    cb2 <- do.call(rbind,lapply(split(cb1,cb1$state),transform, order = rank(hattack,ties.method = "first")))
22-
23-    if(num == "best"){
24-      num = 1
25-      cb3 <- cb2[cb2$order == num, ]
26-    }
27-    else if(num == "worst"){
28-      num = 1
29-      cbw <- do.call(rbind,lapply(split(cb1,cb1$state),transform, order = rank(-hattack,ties.method = "first")))
30-      cb3 <- cbw[cbw$order == num, ]
31-    }
32-    else if(num > 0){
33-      cb3 <- cb2[cb2$order == num, ]
34-    }
35-  }
36-
37-  if(testout == "2"){
38-    cb1 <- cb[, c(1,2,4)]
39-    cb1[,3] <- suppressWarnings(as.numeric(cb1[,3]))
40-    cb1 <- na.omit(cb1)
41-    cb1 <- cb1[order(cb1$state, cb1$hname),]
42-    cb2 <- do.call(rbind,lapply(split(cb1,cb1$state),transform, order = rank(hfailure,ties.method = "first")))
43-
44-    if(num == "best"){
45-      num = 1
46-      cb3 <- cb2[cb2$order == num, ]
47-    }
48-    else if(num == "worst"){
49-      num = 1
50-      cbw <- do.call(rbind,lapply(split(cb1,cb1$state),transform, order = rank(-hfailure,ties.method = "first")))
51-      cb3 <- cbw[cbw$order == num, ]
52-    }
53-    else if(num > 0){
54-      cb3 <- cb2[cb2$order == num, ]
55-    }
56-  }
57-
58-  if(testout == "3"){
59-    cb1 <- cb[, c(1,2,5)]
60-    cb1[,3] <- suppressWarnings(as.numeric(cb1[,3]))
61-    cb1 <- na.omit(cb1)
62-    cb1 <- cb1[order(cb1$state, cb1$hname),]
63-    cb2 <- do.call(rbind,lapply(split(cb1,cb1$state),transform, order = rank(pneumonia,ties.method = "first")))
64-
65-    if(num == "best"){
66-      num = 1
67-      cb3 <- cb2[cb2$order == num, ]
68-    }
69-    else if(num == "worst"){
70-      num = 1
71-      cbw <- do.call(rbind,lapply(split(cb1,cb1$state),transform, order = rank(-pneumonia,ties.method = "first")))
72-      cb3 <- cbw[cbw$order == num, ]
73-    }
74-    else if(num > 0){
75-      cb3 <- cb2[cb2$order == num, ]
76-    }
77-  }
78-
79-  cb4 <- merge(x=cb3, y=states, by = "state", all = TRUE)
80-  cb5 <- cb4[order(cb4$state, cb4$hname, cb4$order),]
81-
82-  return(cb5[,2:1])
83- }
84-

```