Crystal Mosley
MSDS 454 Sec 55

# Individual Assignment #3

## Introduction

The purpose of this airline data analysis was to explore how to handle "big data" – meaning, large amounts of data in one sitting. "Big data" comes into play when the CPU time for any calculation takes longer than the cognitive process of designing a model. It's not about the size of the original data set, but about the size of the biggest object created during the analysis process. Prior to now, I had not explored anywhere near one million observations of data; however, now that I have massaged through this data set, I can say that I have handled over 11 million observations of data at one time.

Strategies to handle large amounts of data are:

- Sampling
    - Sampling is a strategy that should be utilized as a last resort. Reason being, sometimes taking a sample can lead to bias within the data or a decrease in the performance of the modeling being implemented.

- Bigger hardware
    - One thing that hindered my ability to get through this data set quickly, was lack of resources on my laptop. I had to use a colleague's gaming computer with more RAM and hyperthreading to run the code. This is because R keeps all objects in memory.

- Storing objects on an external hard drive and analyze it chunk-wise
    - Chunking leads to parallelization if the algorithms allow parallel analysis of the. A downside of this strategy is that only those algorithms can be performed that are explicitly designed to deal with hard drive specific datatypes.

- Integration of higher performing programming languages like C++ or Java
    - This strategy allows the user to move small parts of the R program to another language to avoid bottlenecks and expensive procedures in performance. The purpose is to balance R's well-designed way to handle the data and utilize the higher performance of another language at the same time.

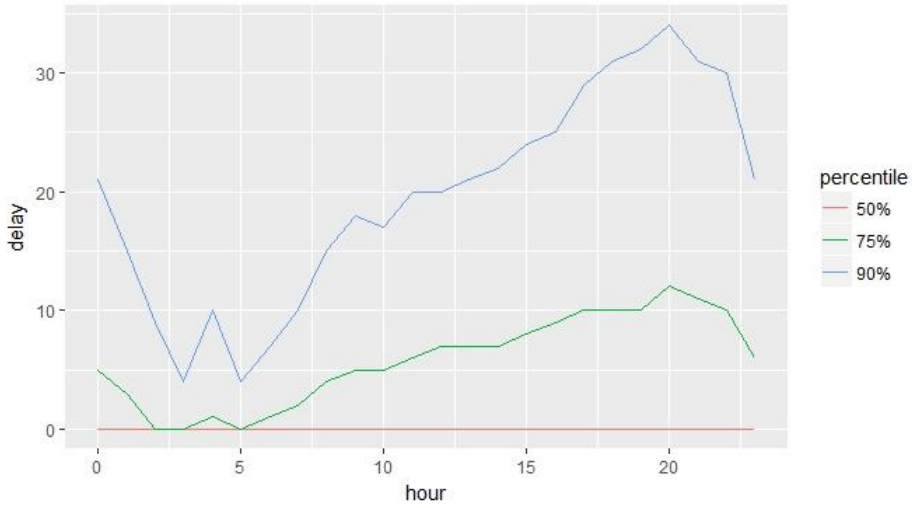
The strategy used to analyze the flight data were to only use the year's 1987-1989, rather than 1987-2013. Because the data totaled over 11 million observations, we used R's *bigmemory* package the provides matrix-like functionality to help with the issue of not having enough RAM available. This package is great in working with R's other functions due to not having to modify (or modify very little) the functions, which save on the amount of time needed to perform standard manipulations and analyses.

## Plot #1: *Hour of the Day*

**Figure 1** shows data for the hour of the day in which the most flight delays occur within the 50%, 75%, and 90% percentiles. Greater than 75% of flight delays occur between the hours of 5:00am -

8:00pm before the delays start to decrease for the rest of the night. Using these three percentile ranges show more of a difference than using the original 90% + percentile ranges. In comparison, the original percentiles indicate the worse flight delays occur early in the morning hours (starting at midnight), and between the hours of 6:00am – 4:00pm, flight delays are at its fewest.

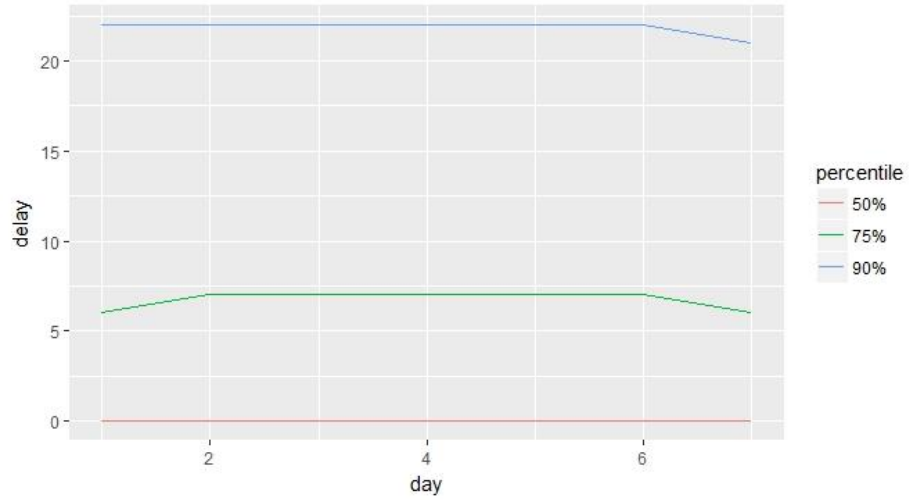
**Figure 1: Hour of day for most flight delays**




**Plot #2:** *Day of the Week*

**Figure 2** shows data for the day of the week in which the most flight delays occur within the 50%, 75%, and 90% percentiles. Tuesday – Saturday show about a 6-7-minute delay for 75% of flights, while Monday – Saturday show about a 22-minute delay for 90% or more of flights. Sunday seems to be the only day where delays decrease.

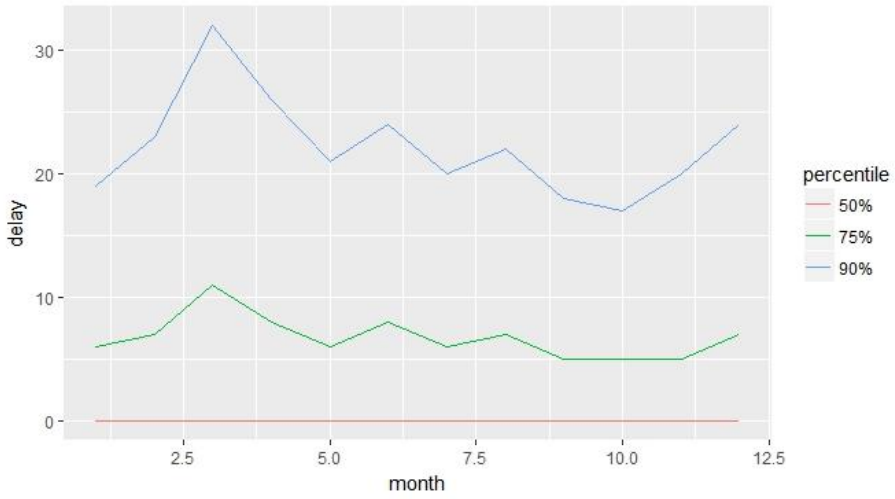
**Figure 2: Day of the week for most flight delays**



**Plot #3:** *Month of the Year*

**Figure 3** shows data for the month of the year in which the most flight delays occur within the 50%, 75%, and 90% percentiles. After reviewing this plot, we can see that most flight delays occur during the month of March and steadily decrease throughout the rest of the year until around the end of October or beginning of November when delays begin to increase for the rest of the year. This could be because of the holiday and winter seasons beginning in October through the end of March.

**Figure 3: Month of the year for most flight delays**



**Appendix A: Relevant R Code**

```
library(ggplot2)
library(readr)
library(ggplot2)
library(reshape)
library(bigmemory)
library(foreach)
library(parallel)
library(doSNOW)

# Get column names and types
integer.columns <- sapply(x, is.integer)
factor.columns  <- sapply(x, is.factor)
factor.levels   <- lapply(x[, factor.columns], levels)
n.rows <- 0L

# Process each file determining the factor levels
for (year in 1987:1989) {
  file.name <- paste(year, "csv", sep = ".")
  cat("Processing ", file.name, "\n", sep = "")
  d <- read.csv(file.name)
  n.rows <- n.rows + NROW(d)
  new.levels <- lapply(d[, factor.columns], levels)
  for ( i in seq(1, length(factor.levels)) ) {
    factor.levels[[i]] <- c(factor.levels[[i]], new.levels[[i]])
  }
  rm(d)
}
save(integer.columns, factor.columns, factor.levels, file = "factors.RData")

# Convert all factors to integers so we can create a bigmatrix of the data
col.classes <- rep("integer", length(integer.columns))
col.classes[factor.columns] <- "character"
cols  <- which(factor.columns)
first <- TRUE
csv.file <- "airlines.csv"   # Write combined integer-only data to this file
csv.con  <- file(csv.file, open = "w")

for (year in 1987:1989) {
  file.name <- paste(year, "csv", sep = ".")
  cat("Processing ", file.name, "\n", sep = "")
  d <- read.csv(file.name, colClasses = col.classes)
  ## Convert the strings to integers
  for ( i in seq(1, length(factor.levels)) ) {
    col <- cols[i]
    d[, col] <- match(d[, col], factor.levels[[i]])
  }
```

```
  write.table(d, file = csv.con, sep = ",",
              row.names = FALSE, col.names = first)
  first <- FALSE
}
close(csv.con)

#-------------------------------------------------
# Plot 1 for Assignment 3 - hour of day
#-------------------------------------------------

numParallelCores <- max(1, detectCores()-1)
cl <- makeCluster(rep("localhost", numParallelCores), type = "SOCK")
registerDoSNOW(cl)

myProbs1 <- c(0.50,0.75,0.90) #use for plots 2 & 3

delayQuantiles1 <- foreach(hour=hourInds, .combine=cbind) %dopar% {
  require(bigmemory)
  x4.1 <- attach.big.matrix("airlines.desc")
  quantile(x4.1[hour, "DepDelay"], myProbs1, na.rm=TRUE)
}
colnames(delayQuantiles1) <- names(hourInds)

stopCluster(cl)

dq1 <- melt(delayQuantiles1)
names(dq1) <- c("percentile", "hour", "delay")
qplot(hour, delay, data = dq1, color = percentile, geom = "line")

#-------------------------------------------------
# Plot 2 for Assignment 3 - day of week
#-------------------------------------------------

numParallelCores <- max(1, detectCores()-1)
cl <- makeCluster(rep("localhost", numParallelCores), type = "SOCK")
registerDoSNOW(cl)

dayInds <- split(1:nrow(x), x[,"DayOfWeek"])

delayQuantiles2 <- foreach(DayOfWeek=dayInds, .combine=cbind) %dopar% {
  require(bigmemory)
  x4.2 <- attach.big.matrix("airlines.desc")
  quantile(x4.2[DayOfWeek, "DepDelay"], myProbs1, na.rm=TRUE)
}
colnames(delayQuantiles2) <- names(dayInds)

stopCluster(cl)
```

```
dq2 <- melt(delayQuantiles2)
names(dq2) <- c("percentile", "day", "delay")
qplot(day, delay, data = dq2, color = percentile, geom = "line")

#------------------------------------------------------
# Plot 3 for Assignment 3 - month of year
#------------------------------------------------------

numParallelCores <- max(1, detectCores()-1)
cl <- makeCluster(rep("localhost", numParallelCores), type = "SOCK")
registerDoSNOW(cl)

monthInds <- split(1:nrow(x), x[,"Month"])

delayQuantiles3 <- foreach(Month=monthInds, .combine=cbind) %dopar% {
  require(bigmemory)
  x4.3 <- attach.big.matrix("airlines.desc")
  quantile(x4.3[Month, "DepDelay"], myProbs1, na.rm=TRUE)
}
colnames(delayQuantiles3) <- names(monthInds)

stopCluster(cl)

dq3 <- melt(delayQuantiles3)
names(dq3) <- c("percentile", "month", "delay")
qplot(month, delay, data = dq3, color = percentile, geom = "line")
```