Data Expo 2009: The Airline Data Set... What's the big deal?

Michael Kane and Jay Emerson

The Airline Data Set

- Flight arrival and departure details for all* commercial flights within the USA, from October 1987 to April 2008.
- Nearly 120 million records, 29 variables (mostly integer-valued)
- We preprocessed the data, creating a single CSV file, recoding the carrier code, plane tail number, and airport codes as integers.

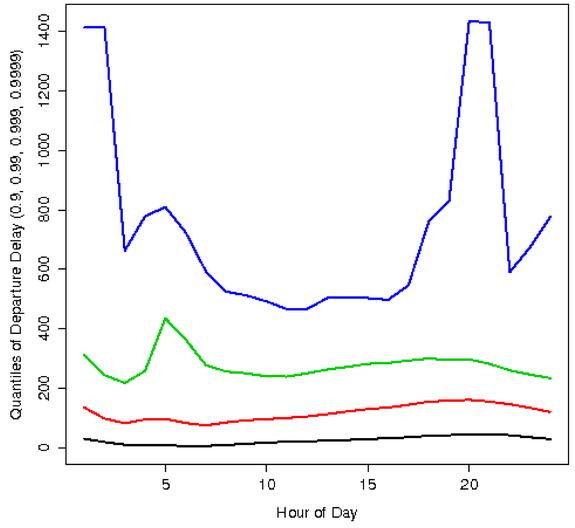
* Not really. Only for carriers with at least 1% of domestic flights in a given year.

Loading the data into a big.matrix object and doing fast, simple exploration

```
# One-time creation of the file-backing from the CSV file:
> require(bigmemory)
> x <- read.big.matrix('airline.csv',
+                      header=TRUE, type='integer',
+                      backingfile='airline.bin',
+                      descriptorfile='airline.desc', extraCols='age')
# Takes 23 minutes.  Subsequent sessions can connect to the
# backing instantaneously using:
# x <- attach.big.matrix(dget('airline.desc'))

# Now we can do things like:
> colnames(x)                # Note it's like a matrix, not a data.frame
 [1] "Year"              "Month"             "DayofMonth"
 [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
 [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
[10] "FlightNum"         "TailNum"           "ActualElapsedTime"
[13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
[16] "DepDelay"          "Origin"            "Dest"
[19] "Distance"          "TaxiIn"            "TaxiOut"
[22] "Cancelled"         "CancellationCode"  "Diverted"
[25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
[28] "SecurityDelay"     "LateAircraftDelay" "age"

# The column range for the first column
> colrange(x, 1, na.rm=TRUE)  # 6.4 seconds
      min  max
Year 1987 2008

# The first column is cached a second operation on the column is fast.
> colmean(x, 1, na.rm=TRUE)  # 0.148 seconds
    Year
1998.624
```

When is the best hour of the day to fly to minimize delays? A simple computation done in parallel on 3 cores.

```
require(foreach)
require(doMC)
registerDoMC(cores=3)

probs <- c(0.9, 0.99, 0.999, 0.9999)
desc <-  describe(x)

# delays by hour of day. Takes 26.7 seconds.
anshourofday <-
  foreach (i=seq(0, colmax(x,"CRSDepTime")-1, by=60),
           .combine=cbind)%dopar%
{
  require(bigmemory)
  x <- attach.big.matrix(desc)
  ind <- mwhich(x, "CRSDepTime", c(i, i+60), comps=c('ge', 'lt'))
  m <- cbind(probs, quantile(x[ind, "DepDelay"], probs=probs, na.rm=TRUE))
  colnames(m) <- c("Probabilites", "Quantiles")
  t(m)[2,]
}
```



Do older planes suffer more delays? Maybe. A computationally intensive example done in parallel.

```
planeStart <- big.matrix(nrow=length(unique(x[,11])), ncol=1, shared=TRUE)
psDesc <- describe(planeStart)
# This will take about 3 hours.
foreach(i=1:nrow(planeStart)) %dopar%
{
  require(bigmemory)
  x <- attach.big.matrix(desc)
  planeStart <- attach.big.matrix(psDesc)
  # The first year plane i can be found:
  yearInds <- mwhich(x, "TailNum", i, comps='eq')
  minYear <- min( x[yearInds, "Year"], na.rm=TRUE )
  # The first month in minYear where the plane can be found.
  minMonth <- min( x[yearInds, "Month"], na.rm=TRUE )
  planeStart[i,1] <- 12*minYear+minMonth
  return(TRUE)
}

badTailNum = mwhich(x, 11, NA, 'eq')
x[badTailNum,11] <- 1
x[,30] <- x[,1]*12 + x[,2] - planeStart[x[,11],1]  % 45 seconds
x[badTailNum,c(11,30)] <- NA

> blm1 = biglm.big.matrix( ArrDelay ~ age, data=x ) % 3 minutes 17 seconds
> summary(blm1)
Large data regression model: biglm(formula = formula, data = data, ...)
Sample size =  84216580
              Coef    (95%     CI)     SE p
(Intercept) 6.8378 6.8266 6.8489 0.0056 0
age         0.0122 0.0121 0.0124 0.0001 0

> blm2 <- biglm.big.matrix( ArrDelay ~ age + Year, data=x )
> summary(blm2)
Large data regression model: biglm(formula = formula, data = data, ...)
Sample size <-  84216580
               Coef    (95%     CI)     SE p
(Intercept) 94.3068 90.2993 98.3142 2.0037 0
age          0.0141  0.0139  0.0143 0.0001 0
Year        -0.0437 -0.0457 -0.0417 0.0010 0

# Without bigmemory... slow and next to impossible, even with 32 GB RAM.
# With bigmemory... what's the big deal?  The sky is the limit!
```