STATS 769 - Lab 07 - bole001

Bernard O'Leary

30 September 2019

Dataset description

Dataset is the same as what we used for Lab06, this time we use the entire dataset though rather than just 100000 records. Run the following code at the command prompt - result reports that 2096.1mb of memory is used by the procedure in R.

```
maxmem <- function(mem1, mem2) {
    mem2[2, 6] - mem1[2, 6]
}
mem1 <- gc(reset=TRUE)
colnames <-
  gsub(" ", ".",
    scan("/course/data.austintexas.gov/Dockless_Vehicle_Trips.csv",
      sep=",", what="character", strip.white=TRUE, nlines=1))
trips <-
  read.csv("/course/data.austintexas.gov/Dockless_Vehicle_Trips.csv",
    header=FALSE, skip=2, col.names=colnames)
trips <- subset(trips, Trip.Duration > 0 & Trip.Distance > 0)
trips$logDuration <- log(trips$Trip.Duration)
trips$logDistance <- log(trips$Trip.Distance)
fit <- lm(logDuration ~ logDistance, trips)
coef(fit)
mem2 <- gc()
maxmem(mem1, mem2)
```

Reduce file to on Trip Duration and Trip Distance

Run the following awk command to achieve this, including max memory usage statistics, which was 1712kb (just under 2mb - not much).

```
/usr/bin/time -f "%M" awk -F ',' '{print $4","$5}' /course/data.austintexas.gov/Dockless_Vehicle_Trips.c
```

```
## 1708
```

Read in file

First read in as a data.frame and then as a data.table, measure and report resulting sizes. Sizes are 52260224 bytes for the data.frame and 52260544 bytes for the data.table - this is not a significant difference.

```
# Read in data.frame
trips <- read.csv("Dockless_Vehicle_Trips_Reduced.csv")
object.size(trips)
```

```
## 52260528 bytes
```

```r
# Read in data.table
library(data.table)
tripsDT <- fread("Dockless_Vehicle_Trips_Reduced.csv", sep=",")
object.size(tripsDT)
```

```
## 52261032 bytes
```

Tidy up the data

In both the data.table and the data.frame, remove where non-positive and log both variables. The data.frame comes out at 203.8mb usage, whereas the data.table comes out at 318.2mb usage. UNcertain as to why the data.table uses more memory than the data.frame, expected it to be the opposite.

```r
# Define maxmem
maxmem <- function(mem1, mem2) {
  mem2[2, 6] - mem1[2, 6]
}

# Dataframe
mem1 <- gc(reset=TRUE)
trips <- subset(trips, Trip.Duration > 0 & Trip.Distance > 0)
trips$logDuration <- log(trips$Trip.Duration)
trips$logDistance <- log(trips$Trip.Distance)
mem2 <- gc()
maxmem(mem1, mem2)
```

```
## [1] 203.9
```

```r
# Datatable
mem1 <- gc(reset=TRUE)
tripsDT <- subset(tripsDT, `Trip Duration` > 0 & `Trip Distance` > 0)
tripsDT$logDuration <- log(tripsDT[,`Trip Duration`])
tripsDT$logDistance <- log(tripsDT[,`Trip Distance`])
mem2 <- gc()
maxmem(mem1, mem2)
```

```
## [1] 318.1
```

Fit Linear Models using lm

Using both data.frame and data.table. Measure memory for both. Memory usage is slightly lower for the data.table at 434mb, compared to 450.2mb for the data.frame.

```r
# Set seed for train/test split
set.seed(101) # Set Seed so that same sample can be reproduced in future also

# Selecting 70% of data as sample from total 'n' rows of the data.frame
```

```r
sample <- sample.int(n = nrow(trips), size = floor(.70*nrow(trips)), replace = F)
train <- trips[sample, ]
test  <- trips[-sample, ]

# Selecting 70% of data as sample from total 'n' rows of the data.table
sampleDT <- sample.int(n = nrow(tripsDT), size = floor(.70*nrow(tripsDT)), replace = F)
trainDT <- tripsDT[sampleDT, ]
testDT  <- tripsDT[-sampleDT, ]

# Dataframe
mem1 <- gc(reset=TRUE)
fit <- lm(y ~ x, data.frame(x=train$logDistance, y=train$logDuration))
mem2 <- gc()
maxmem(mem1, mem2)
```

```
## [1] 450.2
```

```r
# Datatable
mem1 <- gc(reset=TRUE)
fitDT <- lm(y ~ x, data.table(x=trainDT$logDistance, y=trainDT$logDuration))
mem2 <- gc()
maxmem(mem1, mem2)
```

```
## [1] 434
```

Fit Linear Models using biglm

Got another unexpected result here with biglm producing a memory usage profile of 578.5mb for data.frame compared to 642.7mb for data.table. Expected data.table to have lower memory usage. In-fact both results use significantly more memory than using lm.

```r
# Import the library
library(biglm)
```

```
## Loading required package: DBI
```

```r
# Dataframe
mem1 <- gc(reset=TRUE)
fitBigLM <- biglm(y ~ x, data.frame(x=train$logDistance, y=train$logDuration))
mem2 <- gc()
maxmem(mem1, mem2)
```

```
## [1] 578.5
```

```r
# Datatable
mem1 <- gc(reset=TRUE)
fitBigLMDT <- biglm(y ~ x, data.table(x=trainDT$logDistance, y=trainDT$logDuration))
mem2 <- gc()
maxmem(mem1, mem2)
```

```
## [1] 642.8
```

Note regarding “fair comparisons of memory use”

Have tried using profmem and creating an artificially large object to make a high gc trigger, but got similar results to what I have seen above, just larger, so it is likely that the usage I tried was incorrect. I have not included this code here.

Regression coefficients

Regression coefficients are slightly different between the data.frame and data.table, but exactly the same between data.frame whether the approach used was lm or biglm and also for data.table whether the approach was lm or biglm.

```
coef(fit)
```

```
## (Intercept)           x
##   2.3943080   0.5420267
```

```
coef(fitDT)
```

```
## (Intercept)           x
##   2.3916599   0.5423912
```

```
coef(fitBigLM)
```

```
## (Intercept)           x
##   2.3943080   0.5420267
```

```
coef(fitBigLMDT)
```

```
## (Intercept)           x
##   2.3916599   0.5423912
```

MSE for model fits

There may be issues when calculating MSE across the fits regarding the streaming aspect of the biglm. If the data are streamed as needed (uncertain if this is how biglm works), then a global calculation of MSE may not work.

Conculsion and summary

I have found it difficult to get a feel for how data.table is more efficient than data.frame in the context of this proble, because my results show if anything that data.frame is more efficient. I assume this is because I am using the data structure incorrectly and therefore look forward to seeing the model answer.