lab03

Yixuan Li, UPI:yil845

13 August 2021

#The purpose of this lab is to diagnose and resolve issues with large data sets using bash commands and/or R.

Tasks

check total numbers of the interested files.

```
ls /course/NZTA/*TMSTrafficQuarterHour.csv | wc -l
```

```
## 31
```

1, This shell command shows the size of each of the 31 CSV files and the total size of all 31 files.

```
du -ch /course/NZTA/*TMSTrafficQuarterHour.csv
```

```
## 537M /course/NZTA/20130101_20130331_TMSTrafficQuarterHour.csv
## 532M /course/NZTA/20130401_20130630_TMSTrafficQuarterHour.csv
## 550M /course/NZTA/20130701_20130930_TMSTrafficQuarterHour.csv
## 535M /course/NZTA/20131001_20131231_TMSTrafficQuarterHour.csv
## 560M /course/NZTA/20140101_20140331_TMSTrafficQuarterHour.csv
## 562M /course/NZTA/20140401_20140630_TMSTrafficQuarterHour.csv
## 633M /course/NZTA/20140701_20140930_TMSTrafficQuarterHour.csv
## 576M /course/NZTA/20141001_20141231_TMSTrafficQuarterHour.csv
## 571M /course/NZTA/20150101_20150331_TMSTrafficQuarterHour.csv
## 590M /course/NZTA/20150401_20150630_TMSTrafficQuarterHour.csv
## 631M /course/NZTA/20150701_20150930_TMSTrafficQuarterHour.csv
## 606M /course/NZTA/20151001_20151231_TMSTrafficQuarterHour.csv
## 593M /course/NZTA/20160101_20160331_TMSTrafficQuarterHour.csv
## 599M /course/NZTA/20160401_20160630_TMSTrafficQuarterHour.csv
## 640M /course/NZTA/20160701_20160930_TMSTrafficQuarterHour.csv
## 631M /course/NZTA/20161001_20161231_TMSTrafficQuarterHour.csv
## 634M /course/NZTA/20170101_20170331_TMSTrafficQuarterHour.csv
## 681M /course/NZTA/20170401_20170630_TMSTrafficQuarterHour.csv
## 724M /course/NZTA/20170701_20170930_TMSTrafficQuarterHour.csv
## 698M /course/NZTA/20171001_20171231_TMSTrafficQuarterHour.csv
## 687M /course/NZTA/20180101_20180331_TMSTrafficQuarterHour.csv
## 724M /course/NZTA/20180401_20180630_TMSTrafficQuarterHour.csv
## 772M /course/NZTA/20180701_20180930_TMSTrafficQuarterHour.csv
## 758M /course/NZTA/20181001_20181231_TMSTrafficQuarterHour.csv
## 709M /course/NZTA/20190101_20190331_TMSTrafficQuarterHour.csv
## 712M /course/NZTA/20190401_20190630_TMSTrafficQuarterHour.csv
## 753M /course/NZTA/20190701_20190930_TMSTrafficQuarterHour.csv
## 754M /course/NZTA/20191001_20191231_TMSTrafficQuarterHour.csv
## 711M /course/NZTA/20200101_20200331_TMSTrafficQuarterHour.csv
## 707M /course/NZTA/20200401_20200630_TMSTrafficQuarterHour.csv
## 693M /course/NZTA/20200701_20200930_TMSTrafficQuarterHour.csv
## 20G  total
```

2, this shell commands shows the total memory in VM and Ubuntu. The results show that VM has 186Gb available while Ubuntu only has 13Gb. - Ubuntu memory allocation: Total mem 15G, used mem 2.0G, free mem 10G, shared mem 68M, buff/cache 3.1G, availalbe 13G, Swap total 975M, used 0B, free 975M.

```
free -h
```

```
##               total        used        free      shared  buff/cache   available
## Mem:          196Gi       2.5Gi        10Gi       1.0Mi       183Gi       192Gi
## Swap:          11Gi        16Mi        11Gi
```

Import

3, We use profmem in R to check the memory usage of read.csv command of importing data file '20130101_20130331_TMSTrafficQuarterHour.csv'. The result shows 2.55 Gb memory used, which is ~ 5 times of file size. The file has 10583470 rows and 6 columns. A few rows of this file are shown as underlying.

```
library(profmem)
b <- profmem (countsDF<-read.csv("/course/NZTA/20130101_20130331_TMSTrafficQuarterHour.csv"))
total(b)
```

```
## [1] 2546771616
```

```
head(countsDF)
```

```
##   class  siteRef     startDatetime       endDatetime direction count
## 1     H 01S00376 21-FEB-2013 04:15 21-FEB-2013 04:30         1   4.5
## 2     H 01S00376 23-FEB-2013 19:15 23-FEB-2013 19:30         2   4.5
## 3     H 01S00376 23-FEB-2013 23:30 23-FEB-2013 23:45         2   0.5
## 4     L 01S00376 24-FEB-2013 19:00 24-FEB-2013 19:15         2 122.0
## 5     L 01S00376 25-FEB-2013 00:45 25-FEB-2013 01:00         1   5.0
## 6     H 01S00376 25-FEB-2013 07:15 25-FEB-2013 07:30         2   6.5
```

```
dim (countsDF)
```

```
## [1] 10583470        6
```

4, change the command to data.table::fread and check the memory. The total memory used is 512Mb. Comparing to result of read.csv, we use only 1/5 of memory. A few examples of countsDT are shown.

```
library (data.table)
a<-profmem(countsDT<-data.table::fread("/course/NZTA/20130101_20130331_TMSTrafficQuarterHour.cs
v"))
total(a)
```

```
## [1] 512407000
```

```
head (countsDT)
```

```
##    class  siteRef     startDatetime       endDatetime direction count
## 1:     H 01S00376 21-FEB-2013 04:15 21-FEB-2013 04:30         1   4.5
## 2:     H 01S00376 23-FEB-2013 19:15 23-FEB-2013 19:30         2   4.5
## 3:     H 01S00376 23-FEB-2013 23:30 23-FEB-2013 23:45         2   0.5
## 4:     L 01S00376 24-FEB-2013 19:00 24-FEB-2013 19:15         2 122.0
## 5:     L 01S00376 25-FEB-2013 00:45 25-FEB-2013 01:00         1   5.0
## 6:     H 01S00376 25-FEB-2013 07:15 25-FEB-2013 07:30         2   6.5
```

```
dim (countsDT)
```

```
## [1] 10583470        6
```

Transform

5, we used profmem to check memory usage of below commands which converts the startDatetime from character to date format. The total memory used is 931Mb.

```
d<-profmem(countsDF$day <- as.Date(countsDF$startDatetime, format="%d-%b-%Y"))
total(d)
```

```
## [1] 931382520
```

we used profmem to check memory usage of below commands which sum the count per day and select column day, siteRef and class to form a new dataframe dailyCountsDF.The total memory used is 7.6Gb, which is >15 times of file size.

```
f<-profmem(dailyCountsDF<- aggregate(countsDF["count"],countsDF[c("day","siteRef","class")],su
m))
total(f)
```

```
## [1] 7643654880
```

The following code uses ‘data.table’ to do the same thing.The total memory usage is 1.23Gb, which is >5 times saving of memory comparing to data.frame operation.

```
t<- profmem(dailyCountsDT <- countsDT[, day := as.Date(startDatetime, format="%d-%b-%Y")][, sum
(count), .(day, siteRef, class)])
total(t)
```

```
## [1] 1232887256
```

The following code uses shell commands to do the same thing. The total shell memory usage is 4.3Kb. This is a huge saving of memory while using the shell commands. Thus shell commands for data importing and transforming is of the first choice.

```
/usr/bin/time -f "%M" awk -F, 'NR > 1 {gsub(/ .+/,"",$3); print $1","$2","$3", "$6}' /course/NZT
A/20130101_20130331_TMSTrafficQuarterHour.csv | sort -t, -k1,1 -k2,2 -k3,3 | awk '{a[$1]+=$2}END
{for (i in a) print i,a[i]}' > /dev/null
```

```
## 4240
```

6, These R codes count rows of results generated in 5, sort decreasingly by count and show first 10 rows.

```
head(dailyCountsDF [order(dailyCountsDF$count, decreasing=T),], 10)
```

```
##              day  siteRef class count
## 45597 2013-03-28 01N10431     L 99129
## 45558 2013-02-14 01N10431     L 99035
## 45565 2013-02-22 01N10431     L 98618
## 50005 2013-03-28 01N29424     L 98540
## 45564 2013-02-21 01N10431     L 97987
## 45545 2013-02-01 01N10431     L 97696
## 45591 2013-03-21 01N10431     L 97461
## 45572 2013-03-01 01N10431     L 97422
## 45537 2013-01-24 01N10431     L 97257
## 45578 2013-03-08 01N10431     L 97243
```

```
dim (dailyCountsDF)
```

```
## [1] 72157     4
```

```
head (dailyCountsDT [order(dailyCountsDT$V1, decreasing=TRUE)], 10)
```

```
##            day  siteRef class    V1
##  1: 2013-03-28 01N10431     L 99129
##  2: 2013-02-14 01N10431     L 99035
##  3: 2013-02-22 01N10431     L 98618
##  4: 2013-03-28 01N29424     L 98540
##  5: 2013-02-21 01N10431     L 97987
##  6: 2013-02-01 01N10431     L 97696
##  7: 2013-03-21 01N10431     L 97461
##  8: 2013-03-01 01N10431     L 97422
##  9: 2013-01-24 01N10431     L 97257
## 10: 2013-03-08 01N10431     L 97243
```

```
nrow (dailyCountsDT)
```

```
## [1] 72157
```

These shell commands show total rows of output and a few lines in reversed order by “count”.

```
awk -F, 'NR > 1 {gsub(/ .+/,"",$3); print $3","$2","$1", "$6}' /course/NZTA/20130101_20130331_TM
STrafficQuarterHour.csv | sort -t, -k1,1 -k2,2 -k3,3 | awk '{a[$1]+=$2}END{for (i in a) print i,
a[i]}' | wc -l

awk -F, 'NR > 1 {gsub(/ .+/,"",$3); print $3","$2","$1", "$6}' /course/NZTA/20130101_20130331_TM
STrafficQuarterHour.csv | awk '{a[$1]+=$2}END{for (i in a) print i,a[i]}' | sort -t, -n -r -k4 |
head
```

```
## 72157
## 28-MAR-2013,01N10431,L, 99129
## 14-FEB-2013,01N10431,L, 99035
## 22-FEB-2013,01N10431,L, 98618
## 28-MAR-2013,01N29424,L, 98540
## 21-FEB-2013,01N10431,L, 97987
## 01-FEB-2013,01N10431,L, 97696
## 21-MAR-2013,01N10431,L, 97461
## 01-MAR-2013,01N10431,L, 97422
## 24-JAN-2013,01N10431,L, 97257
## 08-MAR-2013,01N10431,L, 97243
```

Model

7,the following code creates a new variable scount that is the square-root of the count.

```
dailyCountsDF$scount<-sqrt(dailyCountsDF$count)
```

8,Split the data into training (90%) and test (10%) sets and fit two models using the training data: one that predicts scount from the overall mean of scount and one that predicts scount from class.

The memory allocation for linear regression of scount with class is 15.5 Mb. Comparing to memory usage of model fitting scount vs. class, the total memory for linear fitting of scount vs.class+siteRef is 1.45Gb (100 times more than that of scount vs. class only). Considering the files size of 573Mb, the memory allocation for modelling is about 3 times of file size.

```
index <- sample(rep(1:10, length.out=nrow(dailyCountsDF)))
train <- dailyCountsDF[index > 1, ]
test <- dailyCountsDF[index == 1, ]

predMean <- mean(train$scount)
l1 <- profmem(lmfit1 <- lm(scount ~ class, train))
l2 <- profmem(lmfit2 <- lm(scount ~ class + siteRef, train))
total(l1)
```

```
## [1] 15449936
```

```
total(l2)
```

```
## [1] 1453733720
```

9, Calculate the RMSE for both models using the test data.

We use model.matrix.lm function to calculate the size of the model matrix for the model 1 and model 2. The dimension of model 2 (64941 X 1327)is much bigger than model 1 (64941 X 2). This is due to matrix is composed of rows of scount by transpose of columns of class in the model1, because we only have H&L two unique class, therefore the matrix is 64941*2 dimension. However, in model 2, the value of siteRef is much bigger, hence the reulting matrix is much bigger than that of model1.

```
RMSE <- function(obs, pred) {
  sqrt(mean((obs - pred)^2))
}
obs <-test$scount
# size of model matrix of model scount vs. class
dim(model.matrix.lm(scount ~ class, train))
```

```
## [1] 64941     2
```

```
# size of model matrix of model scount vs. class + siteRef
dim(model.matrix.lm(scount ~ class + siteRef, train))
```

```
## [1] 64941  1327
```

```
predLM1 <- predict (lmfit1, test)
predLM2 <- predict (lmfit2, test)
# RMSE of model 1 scount vs. class
RMSE(obs, predLM1)
```

```
## [1] 44.75804
```

```
# RMSE of model 2 scount vs class + siteRef
RMSE(obs, predLM2)
```

```
## [1] 20.3645
```

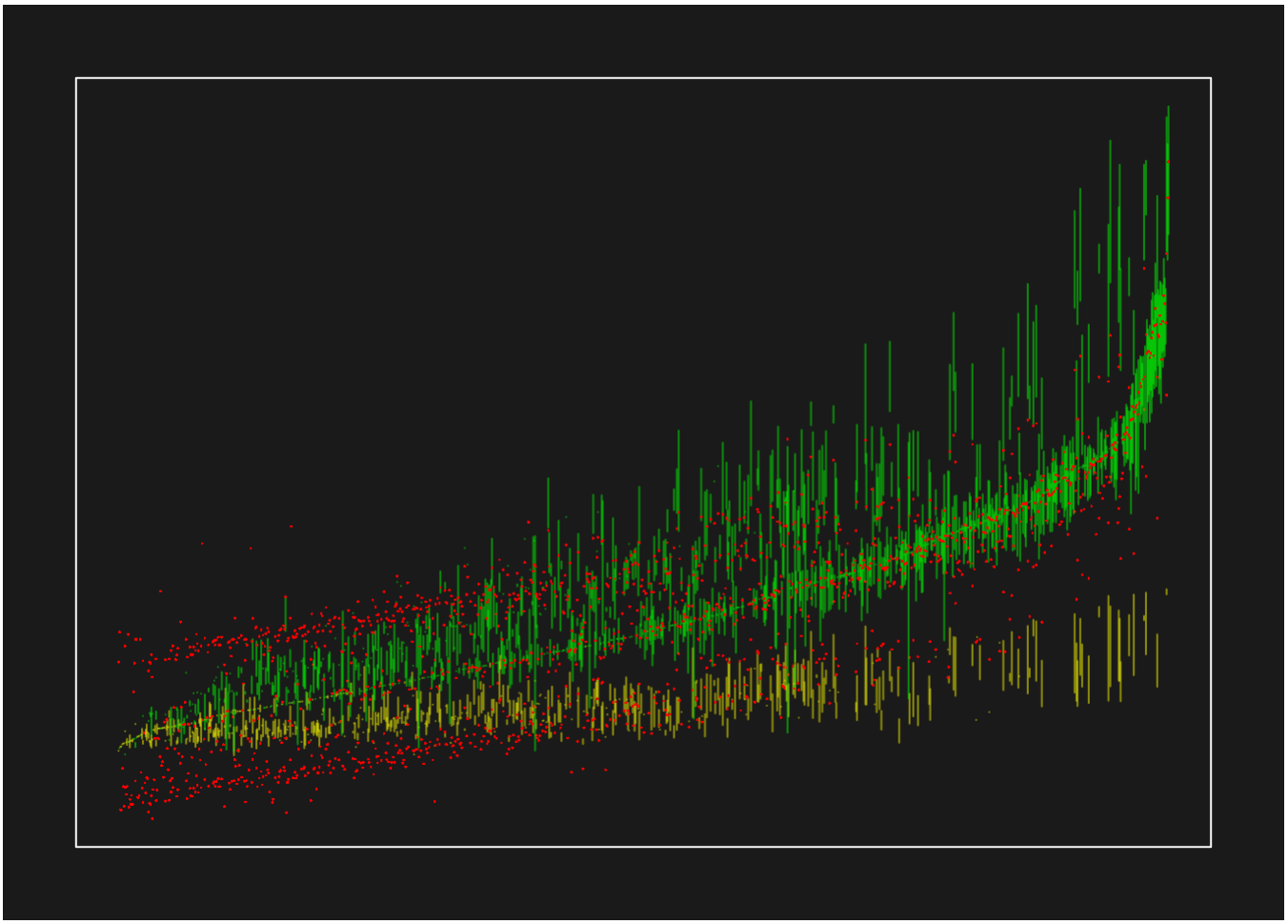
10, We have calculated modelling one file uses memory of about 3 times of files size. For total 20Gb files, we need about 60Gb free memory for linear fitting per person. Taking account of total available memory on VM is ~180Gb, it allows maximum 3 persons to work on VM simutaneouly, which is not possible for group assignment.

The ideal solution is stream-reading. So far the best memory saving way to import and transform data is to use shell commands. Here are the recommended approaches, (1) using awk to extract columns of class, date, siteRef and count from all 31 files and store them in a database, for example SQLite; (2) using biglm to do linear fitting based on a number of data rows consecutively, for example 1000 rows.

11, visulization

The following code produces a plot showing the range of scount values at each site for both H (yellow) and L (green) vehicles along with the value predicted by the second model (red dots).

```
test$pred <- predLM2
test$site <- reorder(as.factor(test$siteRef), test$scount)
testClass <- split(test, test$class)
plotClass <- function(x, col) {
   sites <- split(x, x$site, drop=TRUE)
   siteMin <- sapply(sites, function(y) min(y$scount))
   siteMax <- sapply(sites, function(y) max(y$scount))
   pred <- sapply(sites, function(y) y$pred[1])
   s <- as.numeric(factor(sapply(sites, function(y) y$siteRef[1]),
                          levels=levels(x$site)))
   segments(s, siteMin, s, siteMax, col=col)
   points(s, pred, pch=16, cex=.2, col="red")
}
bg <- "grey10"
par(bg=bg, mar=rep(2, 4))
plot(test$site, test$scount, border=NA, ann=FALSE, axes=FALSE,
     ylim=range(test$scount, test$pred))
usr <- par("usr")
rect(usr[1], usr[3], usr[2], usr[4], col=bg)
box(col="white")
plotClass(testClass[[1]], adjustcolor("yellow", alpha=.5))
plotClass(testClass[[2]], adjustcolor("green", alpha=.5))
```



Summary

In this lab, we explore the memory allocation of using R and shell commands for data import, transformation and modelling. The most memory consuming command for data import is read.csv, which uses ~ 5 times of data size memory. Whereas, data.table command uses similar amount of memory as the size of the file. we find out the the most memory economic way is to use shell commands for data import and transformation (about 100 times smaller than the file size). Model matrix for multilinear regression is much bigger than the single linear regression. For modelling we still need to use R commands. In a normal linear fitting of 3 variables, it takes about 3 times of memory comparing to the file size. For large data modelling, we can use biglm function.

At the last we do a simple modelling based on one set of data, and plot scount vs. class and scount vs. siteRef + class, results show both linear fitting models are not working well. According to RMSE, the second model is slightly better than the first one, which is predicted by scount and class only.