STAT 242
Assignment 5 Working with “Big Data”

Zhewen HU

912490355

SSH: git@bitbucket.org:zhewenhu/stat242assignment4.git

Introduction

In this assignment, we will deal with the New York taxi data. The data is stored in CSV format, organized by year and month. In each file, each row shows the record of a single taxi trip, including drop-off date time, pick-up date time, surcharge and payment type etc. The data takes up about 40 GB which covers 2014 data. Also, we should know that the dataset itself contains a large number of errors. It says that errors account for roughly 7.5% of all trips. As a matter of fact, it will affect the model results to some degree. In this assignment, we will compute the deciles of the totals amount and do the regression models.

Part one Method

One: Database + R + Bootstrap

For method, we use Google BigQuery database to solve the first question and the data pre-processing. BigQuery is based on google cloud platform, which allows us to compute on the 'cloud'. This platform largely speeds up our process of computing. What we have to do is load data into BigQuery. This process is time-consuming. Luckily, I accidentally found someone has already loaded in BigQuery (https://bigquery.cloud.google.com/table/833682135931:nyctaxi.trip_data). Of course, we have checked the validation of the data, which is identical to the dataset on http://www.andresmh.com/nyctaxitrips/ by comparing the computing results of our two methods.

By using SQL-like queries, we can solve the first question in 17 secs on average.

For the question two and three, we pre-processing the data in BigQuery. I select the variables I need: total amount less the tolls, trip time, surcharge. As we plan to use Bootstrap here, I merge the variable table based on the medallion variable, which is driver ID. After merging the table, we found that there are 1,057,934,841,265 records are matched across the two kinds of files(trip_data, trip_fare). Then we do the random sampling in BigQuery with the sample size 1,057,934,841,265^0.6 = 16393638. As time is limited, we only try one sample size. And we sampled for five times. Then we use the sample to do the bootstrap in R and get the regression model for question two and three. For bootstrapping, we use boot package in R and set bootstrap replications to 500 first. Because the process is far too slow, we simply set the bootstrap replications to 10. And we replicate the process for five samples.

Two: Shell + R + biglm Package

In this method, we use Shell command to solve the first question and pre-processing the data for the question two and three. As we plan to use all the data points in this method. We do not match the data in pairs. We do delete nine data points, because there are nine more data points for variables surcharge and (total amount – the tolls). When the data points is ready, we use the biglm package which is designed for bounded memory linear and generalized linear models.

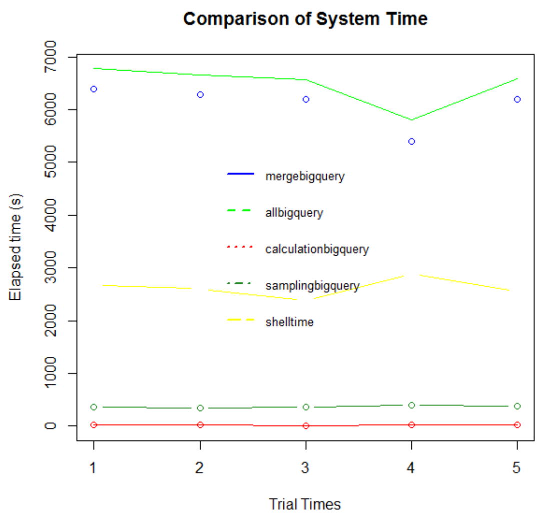
Comparison of Computing Time and Programming Time

For definition of computing time and programming time here, we define the computing time as the time to do the question one and data pre-processing. And we define the programming time as the time to model. Though we think programming time should be the time spent on designing the codes, it is not easy to measure for it is such a long time. Also, as the BigQuery only provides the elapsed time for computing. Here we just use the elapsed time to compare the different approaches.

Part Two Computing Time

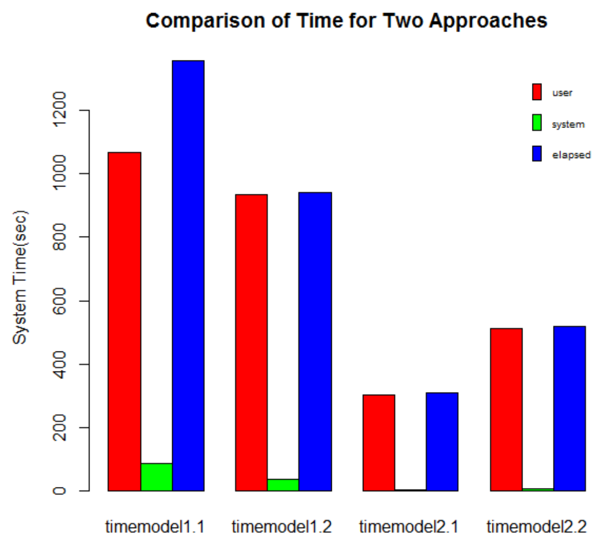
For each approach, we have tested for five times. For BigQuery we record the time by steps: do the calculation of total amount less the tolls, merge the variable into one table, random sampling. For Shell, we record the whole function process time. Here is the results. One can see that though BigQuery performs very well in calculating and gets columns of the dataset. However, it takes a long time to merge table.

If we analyze by steps, we can see that for solving problem one, it is much easier to use BigQuery if the database is ready for you. If we have to set up database for ourselves, it takes longer time. We have tried to upload one file, it takes a long time though it varies from the Internet environment. For pre-processing data part, as we use different method, it is harder to compare and judge. For the shell, as for method two, we do not need to match with driver ID and merge the table. It is much easier to pre-process.



Part Three Programming Time

In this part, we compare the time of modeling. We have experimented for five times for each method and each model, and then take the average of them. Here is the results. The groups of bars represent the method one model 1, method one model 2, method two model 1, method two model 2 respectively. Surprisingly, method two model 1 takes longer time them method two model 2. We expect method two model 2 takes longer time. Because there are two regressors in this model. Probably it is caused by the overloaded server. Two many students were using that server. From the plot below, we can see that method one takes longer time. Because bootstrapping consumes lots of time. If we take into account of merging table in BigQuery, time is much longer than the method two's time. We expect method one will take less time. Because sampling can save time from our common sense.



Part Four Comparison Results

Question One

For two methods, question one gets exactly the same answers. Here is the deciles for total amount less tolls.

	Deciles for Total Amount less Tolls								
Percentile	10%	20%	30%	40%	50%	60%	70%	80%	90%
	6	7.5	8.5	9.75	11	13	15	18.5	26.12

Question Two and Three

We have searched online the total amount less tolls is fare + surcharge + taxi. We do not have too much time doing the model validation. Also it is not reasonable to plot QQ plot, for there are too many data points.

Method 1

Question 2

We can see the figure below that if we increase the triptime by one unit, the total amount less tolls will be increased by 9.34E-05. The increasing amount is trivial. I think because the unit of trip time is secs. It is not easy to see the changes.

Bootstrap Statistics :

	original	**bias**	**std. error**
t1*	5.04E+03	-7.06E+02	2.83E+01
t2*	9.34E-05	-1.82E-05	4.36E-05

Question 3

We can see the figure below that the coefficients are unreasonable. I think it is probably caused by the sampling. Also, the linear regression assumptions are not held here.

Bootstrap Statistics :

	original	**bias**	**std. error**
t1*	5.19E+03	-7.30E+02	3.30E+01
t2*	-5.35E+02	5.50E+01	4.81E+01
t3*	-1.58E-04	1.61E-05	5.63E-05

Method 2

Question 2

We can see that the coefficient here is nearly zero, though the p value shows the coefficients are significant. I guess it is because the unit of trip time is sec.

Large data regression model: biglm(decile.new.1 ~ triptime.numeric.new1, data = modeldata)

Sample size = 173179750

	Coef	**(95%**	**CI)**	**SE**	**p**
(Intercept)	14.5233	14.5129	14.5338	0.0052	0
triptime.numeric.new1	0	0	0	0	0

Question 3

We can see that if surcharge increase by one unit, the total amount less tolls will be increased by 0.1688 units. It makes sense to us.

Large data regression model: biglm(decile.new.1 ~ triptime.numeric.new1 + surcharge.new.1,

data = modeldata)

Sample size = 173179750

	Coef	**(95%**	**CI)**	**SE**	**p**
(Intercept)	14.4694	14.4561	14.4827	0.0066	0
triptime.numeric.new1	0	0	0	0	0
surcharge.new.1	0.1688	0.1432	0.1944	0.0128	0

Comparison

One can see that the results of two methods are very different. One reason is that the data points they are using. And the other reason is the algorithms they use are different. For modelling for real problem, we should consider the efficiency and accuracy. At the same time, we should consider whether the model is easy for explaining the real problems. In short, I do not think two of my approaches are good enough. First, I have spent two weeks to model, which is time-consuming. Second, the model results are not very interpretable, which make me more confused.

Part Five Challenges

Method One:

1. I have to transfer data via different platforms. Because bigrquery package(a package allows us to use BigQuery in R) has bugs. I have to export the results from BigQuery on my local machine, then upload to the department server. The process is time-consuming and full of faults.
2. Which sample size I should use? And for Bag Little Bootstraps, which b I should choose? Of course, BLB provides a good method to deal with big data. But I have no idea which parameter is optimized for this question. What I can do is to try different parameters. However I end up with taking b as 0.6, which is mentioned in the paper. Because I have run out of time.
3. Long time to run bootstrap method. I have to choose the small replication number for bootstrapping, for it takes a lot time to process.

Method Two:

1. Shell commands are slow. But we have to admit it is faster than read.table() function.
2. If a person who is greedy like me, the regression takes a long time. If we use all the data points to do the regression. It will be extremely slow even we use biglm() package. Also, I do not really trust the results of it. Because it is so weird. I am sure about the algorithm of biglm() package is reliable or not.   Also, I do not think it is worthy to use all the data points. From one side, there are some extreme values. Also, if the sampling is scientific enough, why not just use the reasonable sample to do the regression?

Part Six Conclusion: Benefits and Drawbacks

Method 1:

Benefits:

The benefits is largely from BigQuery. Based on my personal experience, I do recommend this platform. If you want to do a project, it is convenient to construct a database on it. The computing speed is quite fast.

Drawbacks:

1. It takes a long time to export data or transfer among platform. Also, you have to pay for the service after the 30 days free trial. But you can just use Google computing which connected with BigQuery. You can use R and other software there.
2. It is not easy to wisely choose sample size for Bootstrap.

Method 2

Benefit:

The method is quite basic and easy to understand. I do have tried the parallel method. But as the server is overloaded sometimes, it is not easy to realize. By this method, you can get the answer you want in the end. However, you have to sacrifice time.

Drawbacks:

1. Not efficient. Though you will get the answer, it takes too much time to wait for the results.
2. Interpretable or not. From my common sense, the reason of regression should be very accurate. Because we have used all the data points. However, the result is so hard, at least for me, to interpret. This makes me wonder for the real problem, how much accuracy we should chase for.

Part Seven Discussion:

1. Local machine is not good for big data computing. Do make full use of other platforms or service like Amazon web service or Google Computing.
2. It is very important to learn how to sample from the big data. It is really realistic to use all the data set to analyze. From one side, there are some errors, which should be deleted in the process of data cleaning. From the other side, time is limited. It is very important to get the information we need from the big data.
3. It is not good to realize the parallel computing on the department server, which affects other students' work.
4. If I have more time, I will spend more time in validating the model and model selection so that I can check linear regression is good for this situation or not.
5. It is important to learn more and fast. When doing this assignment, it is easy to involve in the status of frustration. Though the way to solve problem is fairly simple, it is not easy to learn new technology fast to speed up the whole process. Also, I have found there are various big data products. It is not easy to choose from.
6. Big data computing involves different aspects, like data storage, data computing and data visualization. It is not easy to improve the whole process.

Appendix:

Method one:

SQL BigQuery:

```
##Step One Calculation##
SELECT hack_license, surchage, (FLOAT(total_amount)-FLOAT(tolls_amount)) AS decile
FROM [833682135931:nyctaxi.trip_fare];
####deciles
SELECT percentile_cont(0.1) OVER ( ORDER BY decile) AS one,
    percentile_cont(0.2) OVER ( ORDER BY decile) AS two,
        percentile_cont(0.3) OVER ( ORDER BY decile) AS three,
        percentile_cont(0.4) OVER ( ORDER BY decile) AS four,
        percentile_cont(0.5) OVER ( ORDER BY decile) AS five,
        percentile_cont(0.6) OVER ( ORDER BY decile) AS six,
        percentile_cont(0.7) OVER ( ORDER BY decile) AS seven,
        percentile_cont(0.8) OVER ( ORDER BY decile) AS eight,
        percentile_cont(0.9) OVER ( ORDER BY decile) AS nine,
FROM [taxi.decile]

SELECT percentile_cont(0.1) OVER ( ORDER BY decile) AS tenpercentile
FROM [taxi.decile]

SELECT percentile_cont(0.1) OVER ( ORDER BY decile) AS tenpercentile
FROM [taxi.decile]

SELECT percentile_cont(0.1) OVER ( ORDER BY decile) AS tenpercentile
FROM [taxi.decile]

SELECT percentile_cont(0.1) OVER ( ORDER BY decile) AS tenpercentile
FROM [taxi.decile]
```

```
SELECT percentile_cont(0.1) OVER ( ORDER BY decile) AS tenpercentile
FROM [taxi.decile]
##Step two triptime##
SELECT hack_license, trip_time_in_secs
FROM [833682135931:nyctaxi.trip_data];

##Step three merge table##
SELECT t1.hack_license, t1.surcharge, t1.decile, t2.trip_time_in_secs
FROM [taxi.decile2] AS t1
JOIN EACH
[taxi.triptime] AS t2
ON t1.hack_license = t2.hack_license;

##Step Four Random Sampling##
SELECT  * FROM [taxi.mergebig]
WHERE RAND()<16393638/1057934841265;

R-Bootstrap:
##Computing on server Gumbel
install.packages("boot")
library("boot")
library("read.table")

###upload the five sample files to the server take one of file as an example here
setwd("/home/zhewenhu")
sample = fread("mergesample1.csv",header = TRUE,sep = ",")

beta <- function(formula,data,indices){
  d<-data[indices,]
  fit<-lm(formula,data=d)
  return(fit$coef)
}


samplemodel <- boot(data = sample, statistic = beta, R = 10, formula = decile ~ triptime)

samplemodel2 <- boot(data = sample, statistic = beta, R = 10, formula = decile ~ triptime +
surcharge)

boot.ci(samplemodel)

print(samplemodel)

print(samplemodel2)


timemodel = system.time(boot(data = sample, statistic = beta, R = 10, formula = decile ~
triptime))

timemodel2 = system.time(boot(data = sample, statistic = beta, R = 10, formula = decile ~
triptime + surcharge))
```

Method Two:

```
##Multiple files for the first question

con11 = pipe("cut -f11 -d, *trip_fare*", open = 'r')

con10 = pipe("cut -f10 -d, *trip_fare*", open = 'r')

total = readLines(con11)

tolls = readLines(con10)

total.numeric = as.numeric(as.character(total))

tolls.numeric = as.numeric(as.character(tolls))

decile = total.numeric - tolls.numeric

decile.new = decile[!is.na(decile)] ##get rid of the header


##triptime

contime = pipe("cut -f9 -d, *trip_data*", open ='r')

triptime = readLines(contime)

triptime.new = as.numeric(as.character(triptime))
triptime.new.1 = triptime[!is.na(triptime)]

##surcharge
concharge = pipe("cut -f7 -d, *trip_fare*", open = 'r')
surcharge = readLines(concharge)
surcharge = as.numeric(as.character(surcharge))
surcharge.new = surcharge[!is.na(surcharge)]

##cbind into a data table
surcharge.new.1 = surcharge[-(1:9)]
triptime.new.1 = triptime.new[-(1:9)]
modeldata = cbind(triptime.new.1,surcharge.new.1,trip)

##Question one quantile
deciles = quantile(modeldata$decile.new.1,c(.1,.2,.3,.4,.5,.6,.7,.8,.9))

##big funtion to be measured for the system time
decilecalculation = function()(
  con11 = pipe("cut -f11 -d, *trip_fare*", open = 'r')
  con10 = pipe("cut -f10 -d, *trip_fare*", open = 'r')
  total = readLines(con11)
  tolls = readLines(con10)
  total.numeric = as.numeric(as.character(total))
  tolls.numeric = as.numeric(as.character(tolls))
  decile = total.numeric - tolls.numeric
  decile.new = decile[!is.na(decile)])

  system.time(decileculation())
##system time for data table
datatablecal = function()(
  con11 = pipe("cut -f11 -d, *trip_fare*", open = 'r')
  con10 = pipe("cut -f10 -d, *trip_fare*", open = 'r')
  total = readLines(con11)
  tolls = readLines(con10)
  total.numeric = as.numeric(as.character(total))
  tolls.numeric = as.numeric(as.character(tolls))
  decile = total.numeric - tolls.numeric
  decile.new = decile[!is.na(decile)]
  concharge = pipe("cut -f7 -d, *trip_fare*", open = 'r')
  surcharge = readLines(concharge)
  surcharge = as.numeric(as.character(surcharge))
  surcharge.new = surcharge[!is.na(surcharge)]
  surcharge.new.1 = surcharge[-(1:9)]
  decile.new.1 = triptime.new[-(1:9)]
  modeldata = cbind(decile.new.1,surcharge.new.1,triptime.new))
system.time(datatablecal())

##Step two Using biglm()
library(biglm)
model1 = biglm(decile.new ~ triptime.new, data = modeldata)
model2 = biglm(decile.new ~ triptime.new + surcharge.new, data = modeldata)
system.time(biglm(decile.new ~ triptime.new, data = modeldata))
system.time(biglm(decile.new ~ triptime.new, data = modeldata))

##Final Step Visualization
##Computing time
shelltime = c(2675.55,2600.47,2380.55,2887.63,2554.34)
calculationbigquery = c(16.6,15.3,13.3,17.2,18.5)
mergebigquery = c(6391.5,6290.4,6190.4,5391.5,6190.8)
samplingbigquery = c(364.9,347.5,363.3,398.7,384.4)
```

```
allbigquery = calculationbigquery + mergebigquery + samplingbigquery

g_range = range(0,allbigquery)

plot(mergebigquery,type = "p", col = "blue", ylim = g_range,xlab="Trial Times",ylab = "Elapsed
time (s)", main = "Comparison of System Time")

lines(allbigquery,type="l",col = "green")

lines(calculationbigquery, type = "o", col = "red")

lines(samplingbigquery, type = "b", col = "forestgreen")

lines(shelltime,type = 'c', col = "yellow")

legend("center",
c("mergebigquery","allbigquery","calculationbigquery","samplingbigquery","shelltime"),
cex=0.8, col= c("blue","green","red","forestgreen","yellow"),

    lty=1:5, lwd=2, bty="n")


##plot programming time

timemodel2.2 = c(512.662,6.872,519.825)

timemodel2.1 = c(304.823,5.765,310.759)

timemodel1.1 = c(1067.287,87.281,1355.201)

timemodel1.2 = c(934.670,38.597,940.689)

timeall = cbind(timemodel1.1,timemodel1.2,timemodel2.1,timemodel2.2)

timeall.new = as.matrix(timeall,dimnames =
list(c("user","system","elapse"),c("model1.1","model1.2","model2.1","model2.2")))

barplot(timeall,beside = TRUE,ylab = "System Time(sec)", main = "Comparison of Time for
Two Approaches",col = rainbow(3))

legend("topright", c("user","system","elapsed"), cex=0.6,

    bty="n", fill=rainbow(3))
```