# Report: Assignment5

Yibing Luo
912491862
URL:

## 1. Introduction

In this assignment, the main purpose is to work with ‘Big Data’. There are two sets of data files which are totally 50-Gigabyte. The first set is about trip data and the second one is about fare data. In fare data, what we care about are ‘total_amount’, ‘tolls_amount’, and ‘surcharge’. In trip data, what we care about is 2‘trip_time_in_secs’.

As a result, mainly, there will be three tasks in this assignment:

1. Go through all the files and extract the target variables;
2. Compute the deciles;
3. Do some regression.

Because of the ‘Big Data’, I try to use different methods to increase the efficiency in each task.

## 2. Manipulate two sets of the files

By comparing each pair of corresponding trip file and fare file with ‘medallion’ (a unique car ID), e.g. trip_data_1.csv and trip_fare_1.csv, we can make sure that the lines in these two different files are exactly matched. And, luckily, all data are clean and well-organized. So we can do our jobs without cleaning data first.

I try R, shell, C and apply parallel computing to achieve this task.

### 2.1 Utilize R

First, I have a try on read.csv() function. But it’s too slow to go through all the files. Each file will cost me almost 5 minutes.

Then I find a package names ‘data.table’. A function ‘fread’ in this package can read files really fast. I use lapply() to process the files and then obtain some lists. Then using function ‘rbindlist’ in this package can return a big dataset combined by all these lists. Using dollar sign can extract the wanted column of variables.

### 2.2 Utilize Shell Script

In this section, the shell script is used to process the files. The shell is a powerful interactive and scripting programing environment such as R. Thus it also has a simple language to process the files.

For fare files, we want to obtain the value of total amount less the tolls amount. It's easy to use shell to do that, all we need is one command:

```
awk –F ',' '{print $11 - $10}' /path/filename
```

Use loop in shell can go through all 12 fare files.

For trip files, we want to extract the value of 'trip_time_in_secs'. As I mentioned before, all data in the files are well-organized. Thus we only need to grasp the ninth filed, namely 'trip_time_in_secs', in trip files. The command is as below:

```
cut –d , -f 9 /path/filename
```

The same, we can process all the trip files with loop in shell.

### 2.3 Utilize C

Both R and shell are script languages, so they gain their advantages in expressing the computation. However, the speed to go through those files is relatively slow. Thus I decide to use C to gain a speedup.

Briefly speaking, in C, I used fopen() to open file, then use getline() to read file line by line. For each line, use strsep() to split it by delimiter, and record the specific fields we will use. Also, I will do 'the total amount less the tolls' in C.

### 2.4 Explorations Using Parallel Computing

I want to use parallel computing to reduce the time to read the files. In R, a package named 'parallel' can do it easily. However, in order to do parallel process for all above three methods in R, there should be some modifications. For shell, we need to use system() in R to call shell command. As for C routine, we can use .C to call it in R function.

Then I used function makeCluster() to creates a set of copies of R running in parallel and communicating over sockets. After that I used parLapply() to go over all the files. The usage of parLapply() is similar to lapply(), except that parLapply is a parallel version.

### 2.5 Conclusion for the methods

***(1) Result***

The results are the same relate to different methods. We can use identical() in R to identify this conclusion.

***(2) Computation Time***

Table 1 compares time consuming by different approaches. In this part, all codes were running on the same server and at the time when the server had the similar status. This can make sure that the difference of system.time is insignificant. Thus I decide to use

elapsed.time because when I use parallel, user.time is not reliable. User.time will record cumulative sum of user times for any child process. So it will give back sum of the time consuming on every core I used.

| Time(sec) / Method | Not Parallel | Parallel(in R) |
|---|---|---|
| **R('data.table')** | 870 | 412 |
| **Shell** | 1210 | 295 |
| **C** | 96 | 21 |

*Table 1 elapsed time for each methods*

From the table, we can observe that C routines with parallel is the fastest among all the methods. And the speedup of shell is the most significant after utilizing parallel.

***(3) Programming Time***

Based on my experience during this homework, writing R and shell codes is not really hard. There are only 4 to 5 rows codes to complete the coding work for both R and shell. And they can express the computing really well.

For C, it is a little hard and complicated to accomplished this work. Thanks to the clean and well- organized data, it won't be more complicated. But my C routine isn't robust enough to run for other .csv files beyond this assignment.

Programming on C cost me a lot of time I think. And the most challenge part is to figure out whether the pointers works fine. There are a lot of pointer manipulations during the file operations. It can be easy to make hidden mistakes about pointer which can lead to crash of the program.

***(4) Benefits and Drawbacks***

Both R and shell are good at expressing computation. They can use simple language to do file operations. But they are much slower comparing with C routine, especially fread() function. Since it reads all files rather than the exact columns we need.

C can be very fast as a compiled language. However it is really complicated to do file operation compared with scripting languages. The C routines I wrote in this assignment is simple and lack of robustness. So it can only be run for these specific files. It's hard to improve its robustness for me which involves a lot of details.

So it's better to use scripting languages if the size of data is not really big. But if we place speed issue at a high priority, we may want to use C/C++.

## 3. Deciles of The Total Amount Less The Tolls

When we get the data we need, namely the value of the total amount less the tolls, it is very straightforward to use R function quantile() to compute deciles. And it only cost 3~4 seconds. The result is below:

| Quantile | **10%** | **20%** | **30%** | **40%** | **50%** | **60%** | **70%** | **80%** |
|---|---|---|---|---|---|---|---|---|
| Value | 6.00 | 7.50 | 8.50 | 9.75 | 11.00 | 13.00 | 15.00 | 18.50 |

## 4. Do some linear regression

***(1) Simple Linear regression***

In this part, I will do regress predicting total amount less the tolls using trip time as the predictor. Using lm() to do regression is slow. So I chose calculate the parameters $\beta_0$ and $\beta_1$ by using formulas. The formulas are as below:

$$\begin{cases} Y = \beta_0 + \beta_1 X \\ \beta_0 = \frac{\sum y_i - \beta_1 \sum x_i}{n} \\ \beta_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \end{cases}$$

n is the number of observes.

Using formula to estimate parameter will only cost 3~4 second in this section. The result is:

$$\beta_0 = \quad 1.4523e{+}01;\ \beta_1 = \quad 2.060e{-}05;$$

***(2) Multiple regression***

In this part, I will do regression predicting total amount less the tolls using trip time and surcharge as predictors. Using lm() directly is really a waste of time here. So, still, I used formula here to solve the problem:

$$\beta = (X'\mathrm{X})^{-1}X'\mathrm{Y}$$

Let $X_1$denote trip time, $X_2$denote surcharge, and Y denote total amount less the tolls. Then the design matrix is $(1\ X_1\ X_2)$. So we can get:

$$\begin{pmatrix} 1 \\ X_1 \\ X_2 \end{pmatrix}(1\ X_1\ X_2) = \mathrm{A};$$

Let

$$A^{-1} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix};$$

As a result, we can derive formula:

$$\beta = \begin{pmatrix} a_{11}1_n + a_{12}X_1 + a_{13}X_2 \\ a_{21}1_n + a_{22}X_1 + a_{23}X_2 \\ a_{31}1_n + a_{32}X_1 + a_{33}X_2 \end{pmatrix} Y;$$

Using this formula, 6~7 seconds will be dedicated to calculate the parameters. The result is:

$$\beta_0 = 1.45\ \text{e+1}; \beta_1 = \ 2.06\text{e-05}; \beta_2 = \ 1.68\text{e-01};$$

## 5. Conclusion

In this assignment, I used some methods to work with 'Big data'. Briefly, we can say that it will be really convenient to use scripting languages to do files operations. However, if we concern more about time issue, we would better use compiled languages such as C or C++. But C/C++ can be little hard to write.

There are also some other methods I can use, such as writing R codes in distributed systems (Hadoop, spark and so on).

My C routine in this assignment lacks robustness. Also if I use C routine to do matrix multiplication, it will be more efficient.