Appendix: Code

1. Deciles of the total amount less the tolls

1.1 Method1: shell + r

```
library(parallel)

#list the file names
f = list.files("/home/data/NYCTaxis/", pattern = "fare.*\\.csv$", full.names = TRUE)

#get the frenquency table of total amount less the tolls
getFee = function(filename){

  #read the file and get total amount less the tolls
  fee = as.numeric(system2("awk", args=c(' -F"," \'NR>1{print $11-$10}\'',filename),
                           stdout = T))
  z = round(fee,digits = 2)
  return(table(z))

}

#make a cluster of 12 node and let each node read one file
cl = makeCluster(12, "FORK")
els =  clusterSplit(cl, f)

wc = clusterApply(cl, els, function(x) getFee(x))
stopCluster(cl)

#merge all informatin in the master node
wc2 = unlist(wc)
v = data.frame(id = names(wc2), freq = wc2)
library(plyr)
z = aggregate(freq ~ id, data = v, sum)

#quantile function
quantile.from.freq = function(x,freq,quant) {
  ord = order(x)
  x = x[ord]
  freq = freq[ord]
  cs = cumsum(freq)/sum(freq)
  return(x[max(which(cs<quant))+1])
}

#get all quantiles
z$id = as.numeric(as.character(z$id))
quant = seq(0.1, 0.9, 0.1)
res = sapply(quant, function(i) {quantile.from.freq(z$id,z$freq,i)})
names(res) = quant
```

1.2 Method2: c + shell

Use c to read each file and get the total amount less the tolls.

```c
/**
 * Function: getFee
 * ------------------------------------------------------------
 * Read the trip_fare file and get the value of total amount less the tolls
 * parameters:
 *             *filename: the name of the trip_fare file
 *             *fee: the column of total amount less the tolls
 */
void getFee(char **filename, double *fee){
  FILE *file = fopen(*filename, "r");

    // If the file can not be opened, print the information and return.
    if(NULL == file)
    {
        fprintf(stderr, "Cannot open file: %s\n", *filename);
        return;
    }

    // malloc a space to store each line of a file
    size_t buffer_size = 256;
    char* buffer = (char*)malloc(buffer_size);
    bzero(buffer, buffer_size);

    //the first line is the header, we do not need it
    getline(&buffer, &buffer_size, file);
    bzero(buffer, buffer_size);

    double total_amount = 0; //the last column-total amount
    double tolls_amount = 0; //the value of tolls amount
    int nl = 0; //the number of each line we will read

    // read each line
    char delims[]   = ",";
    char delims2[]  = "\n";
    while(-1 != getline(&buffer, &buffer_size, file))
    {
        char *result    = NULL;
        char *buffer2   = buffer;

        //the first 10 columns are separated by ,
        for(int j = 0; j<10; j++){
            result = strsep(&buffer2, delims);
        }
        tolls_amount = atof(result);

        //The last column is ended with line break
        result       = strsep(&buffer2,delims2);
        total_amount = atof(result);

        //record total amount minus tolls amount in fee
```

```
            fee[nl] = total_amount - tolls_amount;
            nl      = nl + 1;

            bzero(buffer, buffer_size);


        }

        fflush(stdout);
        fclose(file);
        free(buffer);

}
```

In R, the code is similar with Section 1.1 except the function getFee. It is as follows:

```
dyn.load("getFee.so")

#get the frenquency table of total amount less the tolls
getFee = function(filename){
  #get the number of lines in one file
  numLines = as.integer(system2("wc",
                                args = c("-l", filename,
                                         " | awk '{print $1}'"),
                                stdout = TRUE))

  fee = rep(0, numLines - 1)

  output = .C("getFee", as.character(filename),as.numeric(fee))

  z = round(output[[2]],digits = 2)
  return(table(z))
}
```

1.3 hadoop + r

First use the hadoop to get the frenquency table of total amount less the tolls, then use R to calculate the quntiles.

Since there is no data on the hadoop sever, first download all data and put them on hadoop hdfs. The codes are as follows:

```
for i in `seq 1 12`
do
   wget https://nyctaxitrips.blob.core.windows.net/data/trip_fare_${i}.csv.zip
    unzip trip_fare_${i}.csv.zip
    mv trip_fare_${i}.csv data/
    hadoop fs -put data/trip_fare_${i}.csv data/
done
```

Then use python to write mapper.py and reducer.py.

```
#mapper.py
import sys
for line in sys.stdin:
```

```
    line = line.strip()
      if(line.startswith('medallion') == 0):
          tmp = line.split(',')
          word = round(float(tmp[-1])- float(tmp[-2]),2)
          print "%f\t%s" % (word, 1)
```

The reducer.py is as follows(I got this reducer.py code from http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/):

```
#reducer.py

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
  line = line.strip()
    word, count = line.split('\t',1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print "%s\t%s" % (current_word, current_count)
        current_count = count
        current_word = word

if word == current_word:
    print "%s\t%s" % (current_word, current_count)
```

Then, using streamming command to calculate the frenquency table of total amount less the tolls. The command is as follows(the number of reducers is 10):

```
hadoop jar $HADOOP_HOME/contrib/streaming/hadoop-*streaming*.jar -file mapper.
    py -mapper mapper.py -file reducer.py -reducer reducer.py -numReduceTasks
    10 -input data/* -output output

hadoop fs -getmerge output final.txt
```

At last, in r, get the quantiles.

```
quantile.from.freq = function(x,freq,quant) {
  ord = order(x)
  x = x[ord]
  freq = freq[ord]
  cs = cumsum(freq)/sum(freq)
  return(x[max(which(cs<quant))+1])
}

con = file('final.txt')
```

```
data = readLines(con)
close(con)

a = sapply(data, function(i) as.numeric(unlist(strsplit(i,'\t'))))
a = unname(a)

quant = c(0.1,0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)

res = sapply(quant, function(i) {quantile.from.freq(a[1,],a[2,],i)})

names(res) = quant
```

2.Linear regression predicting total amount less the tolls using trip time as the predictor

2.1 Check whether each row of trip data file and trip fare file matched.

c part is used to compare each pair of trip_data and trip_fare files. R part is used to make cluster and run c code parallel.

```c
/**
 * Function: check
 * ----------------------------------------------------
 * Read a pair of trip_fare file and trip_data files,
 * check whether they matched line by line
 * parameters:
 *             *filename_fare: the name of the trip_fare file
 *             *filename_data: the name of the trip_data file
 *             *unequal: whether each row in the two files matched(0 means
    match)
 */
void check(char **filename_fare, char **filename_data, int *unequal){

    //open two files
    FILE *file_fare = fopen(*filename_fare, "r");
    FILE *file_data = fopen(*filename_data, "r");


    // If one or two files can not be opened, print the information and return
    if(NULL == file_fare || NULL == file_data)
    {
        fprintf(stderr, "Cannot open file");
        return;
    }


    //malloc a space to store each line of a file
    size_t buffer_size = 256;

    char* buffer1 = (char*)malloc(buffer_size);
    char* buffer2 = (char*)malloc(buffer_size);
```

```
31
32      bzero(buffer1, buffer_size);
33      bzero(buffer2, buffer_size);
34
35      //The first line is the header, we do not need them
36      getline(&buffer1, &buffer_size, file_fare);
37      getline(&buffer2, &buffer_size, file_data);
38
39      //four columns are used to see if they match
40      char *medallion1 = NULL;
41      char *hack_license1 = NULL;
42      char *vendor_id1 = NULL;
43      char *pickup_datatime1 = NULL;
44
45      char *medallion2 = NULL;
46      char *hack_license2 = NULL;
47      char *vendor_id2 = NULL;
48      char *pickup_datatime2 = NULL;
49
50
51      int nl = 0;
52
53      // read each line of the two files
54      while(-1 != getline(&buffer1, &buffer_size, file_fare) && -1 != getline(&
            buffer2, &buffer_size, file_data))
55      {
56          //from the line in trip_fare file, get value of medallion,
                hack_license, vendor_id1 and pickup time
57          char *buffer4 = buffer1;
58          medallion1 = strsep(&buffer4, ",");
59          hack_license1 = strsep(&buffer4, ",");
60          vendor_id1 = strsep(&buffer4, ",");
61          pickup_datatime1 = strsep(&buffer4, ",");
62
63          //from the line in trip_data file, get value of medallion,
                hack_license, vendor_id1 and pickup time
64          char *buffer3 = buffer2;
65          medallion2 = strsep(&buffer3, ",");
66          hack_license2 = strsep(&buffer3,",");
67          vendor_id2 = strsep(&buffer3,",");
68          pickup_datatime2 = strsep(&buffer3,",");
69          pickup_datatime2 = strsep(&buffer3,",");
70          pickup_datatime2 = strsep(&buffer3,",");
71
72          //check whether they are same in two files
73          if(strcmp(medallion1, medallion2) == 0 && strcmp(hack_license1,
                hack_license2) == 0 && strcmp(vendor_id1,vendor_id2) == 0 &&
                strcmp(pickup_datatime1, pickup_datatime2) == 0){
74              unequal[nl] = 0;
75          }else{
76              unequal[nl] = 1;
77          }
78          nl = nl + 1;
79
```

```
80          bzero(buffer1, buffer_size);
81          bzero(buffer2, buffer_size);
82
83      }
84
85      free(buffer1);
86      free(buffer2);
87
88      fflush(stdout);
89
90      fclose(file_fare);
91      fclose(file_data);
92  }
```

R part is used to run c code parallel and get the result whether each row in two kind of files matched.

```
library(parallel)

f1 = list.files("/home/data/NYCTaxis/", pattern = "fare.*\\.csv$", full.names = TRUE)

f2 = list.files("/home/data/NYCTaxis/", pattern = "data.*\\.csv$", full.names = TRUE)

dyn.load("check.so")

#check whether each line of two rows matched,0 means match
unEqual = function(filename){
  numLines = as.integer(system2("wc",
                                args = c("-l", filename,
                                         " | awk '{print $1}'"),
                                stdout = TRUE))

  checkEqual = rep(1, numLines - 1)

  filename1 = filename
  filename2 = sub("fare", "data",filename)

  output = .C("check", as.character(filename1),as.character(filename2), as.integer(checkEqual))

  return(sum(output[[3]]))

}

clzw = makeCluster(12,"FORK")
els = clusterSplit(clzw, f1)
wc = clusterApply(clzw, els, function(x) unEqual(x))
stopCluster(clzw)

#res ==0 means all lines in two kinds of files matched
res = sum(unlist(wc))
res
```

2.2 Linear Regression

Method1: In c part, get trip time and total amount less the tolls in each pair of files.

```
/**
 * Function: getXY
 * ------------------------------------------------------------
 * Read a pair of trip_fare file and trip_data files,
 * get trip_time from trip_data file and total amount less the tolls amount
    from trip_fare file
 * parameters:
 *             **filename_fare: the name of the trip_fare file
 *             **filename_data: the name of the trip_data file
 *             *trip_time: trip time from trip_data file
 *             *fee: the column of total amount less the tolls
 */
void getXY(char **filename_fare, char **filename_data, double *trip_time,
    double *fee){
  FILE *file_fare = fopen(*filename_fare, "r");
    FILE *file_data = fopen(*filename_data, "r");


    // If one or two files can not be opened, print the information and return
    if(NULL == file_fare || NULL == file_data)
    {
        fprintf(stderr, "Cannot open file");
        return;
    }


    //malloc a space to store each line of a file     size_t buffer_size = 256;
    char* buffer1 = (char*)malloc(buffer_size);
    char* buffer2 = (char*)malloc(buffer_size);

    bzero(buffer1, buffer_size);
    bzero(buffer2, buffer_size);

    getline(&buffer1, &buffer_size, file_fare);
    getline(&buffer2, &buffer_size, file_data);

    char *tripTime = NULL; //store trip_time
    double total_amount = 0; //total amount from fare file
    double tolls_amount = 0; //toll amount from fare file


    int nl = 0;

    // read each line of the two files
    while(-1 != getline(&buffer1, &buffer_size, file_fare) && -1 != getline(&
        buffer2, &buffer_size, file_data))
    {
        //get total amount - toll amount from fare file
        char *result = NULL;
        char *buffer3 = buffer1;
```

```
48
49          for(int j = 0; j<10; j++){
50                  result = strsep(&buffer3, ",");
51          }
52          tolls_amount = atof(result);
53
54          result = strsep(&buffer3,"\n");
55          total_amount = atof(result);
56          fee[nl] = total_amount - tolls_amount;
57
58
59          //get trip_time from data file
60          char *buffer4 = buffer2;
61          for(int j = 0; j<9; j++){
62                  tripTime = strsep(&buffer4, ",");
63          }
64          trip_time[nl] = atof(tripTime);
65
66          nl = nl + 1;
67
68          bzero(buffer1, buffer_size);
69          bzero(buffer2, buffer_size);
70
71      }
72
73      free(buffer1);
74      free(buffer2);
75
76      fflush(stdout);
77      fclose(file_fare);
78      fclose(file_data);
79  }
```

In r, invoke c function and get the result.

```
f1 = list.files("/home/data/NYCTaxis/", pattern = "fare.*\\.csv$", full.names = TRUE)

f2 = list.files("/home/data/NYCTaxis/", pattern = "data.*\\.csv$", full.names = TRUE)

dyn.load("getXY.so")

#read file and get sum(X), sum(Y),sum(XY),sum(X2)
lr = function(filename){
  numLines = as.integer(system2("wc",
                                args = c("-l", filename,
                                         " | awk '{print $1}'"),
                                stdout = TRUE))

  trip_time = rep(0, numLines - 1)
  fee = rep(0, numLines - 1)

  filename1 = filename
  filename2 = sub("fare", "data",filename)
```

```
  output = .C("getXY", as.character(filename1),as.character(filename2),
              as.numeric(trip_time),as.numeric(fee))
  trip_time <<-output[[3]]
  fee <<-output[[4]]
  xy = sum(output[[3]]*output[[4]])
  x2 = sum(output[[3]]^2)
  n <<- numLines - 1
  sumx = sum(output[[3]])
  sumy = sum(output[[4]])
  return(c(n = n,sumxy = xy, sumx2 = x2, sumx = sumx, sumy = sumy))

}

#get sse and ssr
anovaLR = function(filename,beta_0,beta_1,ymean){
   pred = beta_0 + beta_1*trip_time
   residual = fee - pred
   sse = sum(residual^2)
   ssr = sum((pred - ymean)^2)
   return(c(sse = sse, ssr = ssr))
}

library(parallel)

clzw = makeCluster(12,"FORK")

els = clusterSplit(clzw, f1)
wc = clusterApply(clzw, els, function(x) lr(x))

res = c(n = 0, sumxy = 0, sumx2 = 0, sumx = 0, sumy = 0)
for(i in 1:12){
   res = res + wc[[i]]
}

up1 = res[2] - res[4]*res[5]/res[1]
down1 = res[3] - res[4]*res[4]/res[1]

beta_1 = up1 / down1

beta_0 = res[5]/res[1] - beta_1*res[4]/res[1]

xmean = res[4]/res[1]
ymean = res[4] / res[1]

clusterExport(clzw,"beta_0", environment())
clusterExport(clzw,"beta_1", environment())
clusterExport(clzw,"ymean", environment())

wc2 = clusterApply(clzw, els, function(x) anovaLR(x,beta_0, beta_1, ymean))
stopCluster(clzw)

res2 = c(sse = 0, ssr=0)
for(i in 1:12){
```

```
   res2 = res2 + wc2[[i]]
}
sse = res2[1]
ssr = res2[2]
mse = sse/(res[1] - 2)
msr = ssr
fStar = msr/mse
ssto = sse + ssr
msto = ssto/(res[1]-1)
```

Method2:

C part is the same, the following is R part:

```
f1 = list.files("/home/data/NYCTaxis/", pattern = "fare.*\\.csv$", full.names = TRUE)

f2 = list.files("/home/data/NYCTaxis/", pattern = "data.*\\.csv$", full.names = TRUE)

dyn.load("getXY.so")

#read files and get trip_time and total amount less tolls
lr = function(filename){
  numLines = as.integer(system2("wc",
                                args = c("-l", filename,
                                         " | awk '{print $1}'"),
                                stdout = TRUE))

  trip_time = rep(0, numLines - 1)
  fee = rep(0, numLines - 1)

  filename1 = filename
  filename2 = sub("fare", "data",filename)


  output = .C("getXY", as.character(filename1),as.character(filename2),
              as.numeric(trip_time),as.numeric(fee))

  return(data.frame(trip_time = output[[3]], fee = output[[4]]))

}


library(parallel)

clzw = makeCluster(12,"FORK")

els = clusterSplit(clzw, f1)
wc = clusterApply(clzw, els, function(x) lr(x))
stopCluster(clzw)

library(biglm)
fit = biglm(fee~trip_time, data= wc[[1]])
for(i in 2:12){fit = update(fit, wc[[i]])}
summary(fit)$mat
```

3. Multiple regression

3.1 Method 1

C part:

```c
/**
 * Function: getX2Y
 * ----------------------------------------------------------
 * Read a pair of trip_fare file and trip_data files,
 * get trip_time from trip_data file, surcharge and total amount less the
     tolls amount from trip_fare file
 * parameters:
 *             **filename_fare: the name of the trip_fare file
 *             **filename_data: the name of the trip_data file
 *             *trip_time: trip time from trip_data file
 *             *surcharge: surcharge from trip_fare file
 *             *fee: the column of total amount less the tolls
 */
void getX2Y(char **filename_fare, char **filename_data, double *trip_time,
    double *surcharge, double *fee){
  FILE *file_fare = fopen(*filename_fare, "r");
    FILE *file_data = fopen(*filename_data, "r");


    // If one or two files can not be opened, print the information and return
    if(NULL == file_fare || NULL == file_data)
    {
        fprintf(stderr, "Cannot open file");
        return;
    }


    //malloc a space to store each line of a file     size_t buffer_size = 256;
    size_t buffer_size = 256;
    char* buffer1 = (char*)malloc(buffer_size);
    char* buffer2 = (char*)malloc(buffer_size);

    bzero(buffer1, buffer_size);
    bzero(buffer2, buffer_size);

    getline(&buffer1, &buffer_size, file_fare);
    getline(&buffer2, &buffer_size, file_data);

    char *tripTime = NULL; //store trip_time
    double total_amount = 0; //total amount from fare file
    double tolls_amount = 0; //toll amount from fare file


    int nl = 0;

    // read each line of the two files
    while(-1 != getline(&buffer1, &buffer_size, file_fare) && -1 != getline(&
        buffer2, &buffer_size, file_data))
```

```
46      {
47          //get total amount - toll amount from fare file
48          char *result = NULL;
49          char *buffer3 = buffer1;
50
51          for(int j = 0; j<7; j++){
52                  result = strsep(&buffer3, ",");
53          }
54          surcharge[nl] = atof(result);
55          result = strsep(&buffer3, ",");
56          result = strsep(&buffer3, ",");
57          result = strsep(&buffer3, ",");
58          tolls_amount = atof(result);
59
60
61          result = strsep(&buffer3,"\n");
62          total_amount = atof(result);
63          fee[nl] = total_amount - tolls_amount;
64
65
66          //get trip_time from data file
67          char *buffer4 = buffer2;
68          for(int j = 0; j<9; j++){
69                  tripTime = strsep(&buffer4, ",");
70          }
71          trip_time[nl] = atof(tripTime);
72
73          nl = nl + 1;
74
75          bzero(buffer1, buffer_size);
76          bzero(buffer2, buffer_size);
77
78      }
79
80      free(buffer1);
81      free(buffer2);
82
83      fflush(stdout);
84      fclose(file_fare);
85      fclose(file_data);
86  }
```

R part:

```
f1 = list.files("/home/data/NYCTaxis/", pattern = "fare.*\\.csv$", full.names = TRUE)

dyn.load("getX2Y.so")

#read file and get the trip_time, surcharge, total amount less tolls
#get the sum information
mlr = function(filename){
  numLines = as.integer(system2("wc",
                                args = c("-l", filename,
                                         " | awk '{print $1}'"),
```

```
                                stdout = TRUE))

  trip_time = rep(0, numLines - 1)
  fee = rep(0, numLines - 1)
  surcharge = rep(0, numLines - 1)

  filename1 = filename
  filename2 = sub("fare", "data",filename)


  output = .C("getX2Y", as.character(filename1),as.character(filename2),
              as.numeric(trip_time),as.numeric(surcharge),as.numeric(fee))

  x12 = sum(output[[3]]^2)
  x1 = sum(output[[3]])
  x22 = sum(output[[4]]^2)
  x2 = sum(output[[4]])
  x1y = sum(output[[3]]*output[[5]])
  x2y = sum(output[[4]]*output[[5]])
  x1x2 = sum(output[[3]]*output[[4]])
  y = sum(output[[5]])
  n = numLines - 1
  return(c(n = n,x1 = x1, x12 = x12, x2 = x2, x22 = x22,
           x1x2=x1x2, x1y=x1y, x2y=x2y, y=y))

}
library(parallel)

clzw = makeCluster(12,"FORK")

els = clusterSplit(clzw, f1)
wc = clusterApply(clzw, els, function(x) mlr(x))
stopCluster(clzw)

res = c(n = 0, x1 = 0, x12 = 0, x2 = 0, x22 = 0, x1x2=0,
        x1y = 0, x2y = 0, y=0)
for(i in 1:12){
   res = res + wc[[i]]
}

sumx12 = res[3] - res[2]*res[2]/res[1]
sumx22 = res[5] - res[4]*res[4]/res[1]
sumx1y = res[7] - res[2]*res[9]/res[1]
sumx2y = res[8] - res[4]*res[9]/res[1]
sumx1x2 = res[6] - res[2]*res[4]/res[1]

beta_1 = (sumx22*sumx1y - sumx1x2*sumx2y)/
  (sumx12*sumx22 - sumx1x2*sumx1x2)
beta_2 = (sumx12*sumx2y - sumx1x2*sumx1y)/
  (sumx12*sumx22 - sumx1x2*sumx1x2)
beta_0 = (res[9] - beta_1*res[2] - beta_2*res[4])/res[1]
```

3.2 Method 2

C part is the same with 3.1. R part is as follows:

```
f1 = list.files("/home/data/NYCTaxis/", pattern = "fare.*\\.csv$", full.names = TRUE)

dyn.load("getX2Y.so")

mlr = function(filename){
  numLines = as.integer(system2("wc",
                                args = c("-l", filename,
                                         " | awk '{print $1}'"),
                                stdout = TRUE))

  trip_time = rep(0, numLines - 1)
  fee = rep(0, numLines - 1)
  surcharge = rep(0, numLines - 1)
  filename1 = filename
  filename2 = sub("fare", "data",filename)


  output = .C("getX2Y", as.character(filename1),as.character(filename2),
              as.numeric(trip_time),as.numeric(surcharge),as.numeric(fee))

  return(data.frame(trip_time = output[[3]],
                    surcharge = output[[4]], fee = output[[5]]))

}
library(parallel)

clzw = makeCluster(12,"FORK")

els = clusterSplit(clzw, f1)
wc = clusterApply(clzw, els, function(x) mlr(x))
stopCluster(clzw)

library(biglm)
fit = biglm(fee~trip_time + surcharge, data= wc[[1]])
for(i in 2:12){fit = update(fit, wc[[i]])}
summary(fit)$mat
```