Appendix

This is the part where we use different methods to manipulate files

use shell to read the file

```
readVar =
  #
  #This function calls shell to extract and calculate the difference, namely
  #response variable.
  #parameter:
  #filename: It is a string, a combination of the path and name of file.
  #
  function(filename){
    dirt = "/home/data/NYCTaxis/"
    as.numeric(system(paste0("awk -F ',' '{print $11 - $10}' ",dirt, filename),
                      intern = TRUE)[-1])
  }


readPred =
  #
  #This function calls shell to extract predictor.
  #filename: It is a string, a combination of the path and name of file.
  #
  function(filename){
    dirt = "/home/data/NYCTaxis/"
    as.numeric(system(paste0("cut -d, -f 9 ",dirt,filename),
                      intern = TRUE)[-1])
  }
```

use parallel +shell to go through files

```
library("parallel")
cl = makeCluster(20)

system.time(response <- parLapply(cl, paste0("trip_fare_",1:12,".csv"), readVar))
system.time(predictor <- parLapply(cl, paste0("trip_data_",1:12,".csv"),readPred)
clusterExport(cl, "results")
```

Use R pkg to manipulate files

```
library("data.table")

clusterEvalQ(cl, library("data.table",lib = "\home\ybluo"))
dat1 =parLapply(paste0("trip_fare_", 1:12,".csv"),
            fread)
dat1  = rbindlist(dat2)
response = dat1$total_amount - dat1$tolls_amount

dat2 = parLapply(paste0("trip_data_", 1:12, ".csv"),
              fread)
dat2 = rbindlist(dat2)
predictor = dat2$trip_time_in_secs
```

use c to manipulate files

```
creadFile_resp=
  #
  #In this R function, a c routine is called to manipulate files in ord
er to get
  #value of total amount less the tolls.
  #filename: a string which is the name of the file.
  #
  function(filename){
    # get the length of file by shell command
    lenght_file = as.numeric(
      unlist(
        strsplit(
          system(paste0("wc -l ",filename),intern = TRUE)," "))[1])

    resp = as.numeric(rep(0,lenght_file))
    dyn.load("/home/ybluo/C/read.so")
    resp = .C("readFile_resp", filename, resp)[[2]]
    dyn.unload("/home/ybluo/C/read.so")
    resp
  }

creadFile2_pred=
  #
  #In this R function, a C routine is called to manipulate files in ord
er to get
  #value of trip time.
  #filename: a string which is the name of the file.
  function(filename){
    #get the length of file by shell command
    lenght_file = as.numeric(
      unlist(
        strsplit(
          system(paste0("wc -l ",filename),intern = TRUE)," "))[1])

    pred = as.numeric(rep(0,lenght_file))
    dyn.load("/home/ybluo/C/read.so")
    pred = .C("readFile_pred", filename, pred)[[2]]
    dyn.unload("/home/ybluo/C/read.so")
    pred
  }

parGety =
  #
  #This function is to go through all 12 trip_fare_*.csv files by calli
ng creadFile_pr
  #ed.
  #num_cl is the number of cluster when we use parallel computing.
  #
  function(num_cl){
    library("parallel",lib = "/home/ybluo")
    cl = makeCluster(num_cl)
    y = parLapply(cl, paste0("/home/data/NYCTaxis/trip_fare_",1:12,".cs
v"),
                  creadFile_resp)
    stopCluster(cl)
    y = unlist(y)
  }
parGetx =
  #
  #This function is to go through all 12 trip_data_*.csv files by calli
ng creadFile_re
  #sp.
  #num_cl is the number of cluster when we use parallel computing.
  #
  function(num_cl){
    library("parallel",lib = "/home/ybluo")
    cl = makeCluster(num_cl)
    x = parLapply(cl, paste0("/home/data/NYCTaxis/trip_data_",1:12,".cs
v"),
                  creadFile_pred)
    stopCluster(cl)
    x = unlist(x)
  }
```

C code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
/*function: readFile_resp
* ------------------------
* This function is to read files and get the value of total amount less
 the tolls.
* First, use fopen() to open the file. Then use getline() to read file
line by line.
* Use strsep to split the lines by delimiter, ',', and get the value fr
om the 9th and
* 10th fields. Calculate the difference.
* filename is the name of file
* fare is a pointer which leads an array
*/
void  readFile_resp(char **filename, double *fare)
{
 FILE* csvFile = fopen(*filename, "r"); /*open the file*/
  size_t buf_size = 100;
  char* buf = (char*)malloc(buf_size);
  char* token = NULL;

  int k = 0;
  double total_amount = 0, tolls_amount = 0;


  if (csvFile)   //whether the file is opend properly
  {
    getline(&buf, &buf_size, csvFile); /*skip the first line, namel
y the header*/

    while (getline(&buf, &buf_size, csvFile) != -1) /*read file lin
e by line*/
    {
      char* buf2 = buf;

      for (int i = 0; i < 10; i++)
      {
        token = strsep(&buf2, ",");
      }
      tolls_amount = atof(token);
      token = strsep(&buf2, "\n");
      total_amount = atof(token);

      fare[k++] = total_amount - tolls_amount; /*record every dif
f*/
    }
    fclose(csvFile);
    free(buf);

  }
}
```

```c
/*function: readFile_pred
* -------------------------
* This function is to read files and get the value of the trip time.
* First, use fopen() to open the file. Then use getline() to read file line by line.
* Use strsep to split the lines by delimiter, ',', and get the value from the 9th
* field.
* filename is the name of file
* sec is a pointer which leads an array
*/
void  readFile_pred(char **filename, double *sec)
{
    FILE* csvFile = fopen(*filename, "r");/*open the file*/
    size_t buf_size = 100;
    char* buf = (char*)malloc(buf_size);
    char* token = NULL;

    int k = 0;


    if (csvFile)   //whether the file is opend properly
    {
        getline(&buf, &buf_size, csvFile);/*skip the first line, namely the header*/

        while (getline(&buf, &buf_size, csvFile) != -1) /*read file line by line*/
        {
            char* buf2 = buf;

            for (int i = 0; i < 9; i++)
            {
                token = strsep(&buf2, ",");
            }
            sec[k++] = atof(token);  /*record every value*/
        }
        fclose(csvFile);
        free(buf);

    }
}
```

This is the part to calculate deciles and do some regressions.

Problem 1, use quantile() directly to calculate deciles

```r
decile =
  function(object){
    quantile(object, prob = seq(0,1,0.1))
  }
```

Problem 2, simple linear regression

method1 very slow

```r
fit = lm(y~x) #y is the response variable and x is predictor
```

method2

```r
cal_beta =
  #
  #This function is to calculate the parameter beta in linear regression.
  #The formula is in the report
  #x is the predictor
  #y is the respose variable
  #
  function(x,y){
    sumx = sum(x)
    sumy = sum(y)
    sumx_sqr = sum(x^2)
    sumxy = sum(x*y)

    if(length(x) != length(y))
      stop("x and y have different length")
    n = length(x)

    beta_1 = (n*sumxy - sumx*sumy)/(n*sumx_sqr - sumx^2)
    beta_0 = (sumy - beta_1*sumx)/n
    return(c(beta_0,beta_1))
  }
```

Problem 3, multiple regression

regression with two regressors

```r
cal_beta =
  #
  #This function is to calculate the parameter beta in regression.
  #The formula is in report
  #x1 is the first predictor and x2 is the second predictor
  #y is the response variable
  #
  function(x1,x2,y){
    if(length(x1) != length(x2)| length(x1) != length(y))
      stop("x1, x2 and y have different length")
    n = length(x1)

    sum_x1 = sum(x1);sum_x2 = sum(x2);
    sum_x2x2 = sum(x2*x2); sum_x1x2 = sum(x1*x2); sum_x1x1 = sum(x1*x
1);
    matX = matrix(c(n, sum_x1, sum_x2,
                    sum_x1, sum_x1x1, sum_x1x2,
                    sum_x2, sum_x1x2, sum_x2x2),nrow = 3, byrow =TRUE)
    rev_matX = solve(matX)
    a = rev_matX

    ele = rep(1,n)
    beta_0 = sum((a[1]*ele + a[4]*x1 +a[7]*x2)*y)
    beta_1 = sum((a[2]*ele + a[5]*x1 +a[8]*x2)*y)
    beta_2 = sum((a[3]*ele + a[6]*x1 +a[9]*x2)*y)
    return(c(beta_0,beta_1,beta_2))
  }
```