Stat243 PS2

Meng Wang, SID: 21706745

Septembre 19, 2015

1 Part(a)

The data can be downloaded with 'wget'

```
url="http://www.stat.berkeley.edu/share/paciorek/ss13hus.csv.bz2"
wget ${url}
```

Since the file is too big to unzip and read into R. We shall first have a look at the file with some bash command.bzip2 provides a command 'bzcat', which allows us access to the file and we could use this command and pipe the file to some other bash commands without actually exacting the file. For example, we could have a look at the fields of the .csv file by the following code

```
bzcat ss13hus.csv.bz2 | head -n 1
```

This provides us the fields contained in this file. We could also count how many fields there are in the file by changing the delimiter ','to '\n'and count the number of lines with command 'wc'

```
bzcat ss13hus.csv.bz2 | head -n 1 | tr ',' '\n' | wc -l
```

and the result is 205.

We are required to take only 13 fields of the data, which are 'ST', 'NP', 'BDSP', 'BLD', 'RMSP', 'TEN', 'FINCP', 'FPARC', 'HHL', 'NOC', 'MV', 'VEH', 'YBL'. We can use 'cut'command to cut the certain columns and save the selected data as a new .csv file and read the sliced data into R for random sampling. Before cutting the file, we need to find out the number of the selected columns in the file. We could do this simply by exacting the 205 field names into a file and use 'grep -n'to look for each field name.

```
for name in "ST" "NP" "BDSP" "BLD" "RMSP" "TEN" "FINCP" "FPARC" "HHL" "NOC" "MV" "VEH" "YBL"
do
  grep -wn $name field_name | sed 's/[^0-9]//g'
done
```

Then we could select the data from the .csv file and put them into a new .csv file.

```
bzcat ss13hus.csv.bz2 | cut -d','  -f8,12,17,18,32,41,49,50,53,64,63,45,47 > selected_data.csv
```

The selected data file 'selected_ data.csv'is only 252MB, which is greatly smaller than the original 5.7GB file. R is totally capable to read in the whole data file now.

Read the selected data into R with 'read.csv'and create 1000 sample of the data. Then save the file as 'sample.csv'

```r
set.seed(0);
setwd("./Stat243/Homework/PS2");
# read in the data
data<-read.csv("selected_data.csv");
# sample 1000 rows of the data
sample<-data[sample(nrow(data), 1000),];
# save it to sample.csv
write.csv(sample, file = "sample_R.csv");
```

The result is saved in file 'sample_ R.csv'.

2 Part(b)

Use 'system.time()'to get the time consumed by the command

```r
system.time(sample1<-read.csv("sample_R.csv"));

##    user  system elapsed
##   0.006   0.000   0.006

system.time(sample2<-readLines("sample_R.csv"));

##    user  system elapsed
##   0.003   0.000   0.003
```

It turns out that 'read.csv'is much slower than 'readLines'.

3 Part(c)

I have used the bash command to explicitly exact the data we are interested in and saved the interested data as 'sample.csv'. This should speed up the flow process than using R function to exact the columns. Actually the whole process can be done with bash except the random sample generating part. With the idea that we want to use as little as R code in mind, we could use R to generate the random line number from 0 to the number of rows in the data file. The number of rows in the data can be easily find out by

```
   bzcat ss13hus.csv.bz2 | wc -l
```

The result is 7219001. Then we use R to generate 1000 random integers from 1 - 721900

```r
# generate 1000 random number as line numbers to extract the data
randnum <- as.data.frame(sample(1:7219000, 1000));
write.csv(randnum, file="randnum");

Exact the second column which contains the random line numbers and put the random line numbers into file
```

```
# exact the second column of randnum
tail -n +2 randnum | cut -d',' -f2 > randLine
```

Then use a for loop to read the random line number from file 'randLine'and exact that line from 'selected$_d$ata.csv'

```
# read the line number in randLine and exact the data
cat randLine | while read line;
do
  head -n $line selected_data.csv| tail -1 >> sample_bash.csv
done
```

By using bash, we could do the same thing with R. The result is saved in file 'sample_ bash.csv'.

4 Part(d)

Check the file 'sample_ R.csv'with 'xtab'and see whether there are any interesting associations. We can check the relationship between tenure of the house 'TEN'and number of vehicles'VEH'.

```
xtabs(~ TEN + VEH, data=sample);
```

```
##    VEH
## TEN   0   1   2   3   4   5   6
##   1  11  96 164  87  18   9   5
##   2  15  76  85  23   7   1   1
##   3  42 110  46  16   5   0   0
##   4   4   5   4   5   1   0   0
```

As you can see, the average number of cars owned by people who own the house with mortgage or load ($TEN = 1$) is 2, while the number for people who rent the house ($TEN = 3$) is roughly 1. That definitely makes some sense.

** all bash scripts are saved in HW2.sh; R code saved in HW2.R; all codes have been run and tested*