Big data Statistics
Exercise Set II

Efthymios Ioannis Kavour

June 6, 2023

Contents

1 Exercise 1

(Big Data Regression: computational techniques). First install the following: *Bioconductor*

```
install.packages("biglm") # version 0.9-2.1
install.packages("fastmatch")
```

Next, download the specific versions of the packages bit 1.1-15.2, ff 2.2-14.2.tar.gz and ffbase 0.12.8. Go to the relevant pages in CRAN repository, search for the archived versions of each package and download the relevant *.tar.gz file. Then, open R and run

```
install.packages("path/bit_1.1-15.2.tar.gz", type = "source", repos
                                    = NULL) install.packages("path/
                                    ff_2.2-14.2.tar.gz", type = "
                                    source", repos = NULL) install.
                                    packages("path/ffbase_0.12.8.tar.
                                    gz", type = "source", repos =
                                    NULL)
```

by replacing path with the path to your download directory. Open R and run the following script

```
set.seed(am) # replace am with your AM p <- rpois(1, lambda = 120)
n <- 2000000
b <- rt(p, df = 5)
outFile <- "big_data_regression.csv"
zz <- file(outFile , "w")
colNames <- c("y", paste0("x", 1:(p-1))) colNames <- paste0(
                                    colNames , collapse=",") cat(
                                    colNames , "\n", file = zz)
for (i in 1:n){
x <- matrix(rnorm(p-1), nrow = 1)
y <- b[1] + x %*% b[-1] + rnorm(1)
xy <- cbind(y, x)
cat(paste0(xy, collapse = ","), file = zz, append=TRUE,"\n") if( i
                                    %% 100000 == 0){
} close(zz)
```

The previous code snippet will create a file: "big data regression.csv" and will write1 each line of the synthetic dataset. The task is to estimate a linear regression model based on this dataset, without loading the data into memory2. The header of the generated file shows the name of the variables: the response variable is y and the remaining ones are explanatory variables.

1. Use the command *bigglm.ffdf()* of the ffbase library to estimate the regression coefficients and report your results. Use the option *sandwich = FALSE*.

2. Compute $X^T X$ and $X^T y$ using a recursive approach and then use the *solve()* command in order to obtain the least squares estimate

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

For this purpose you may loop through successive chunks of the rows of the ffdf object you have already loaded previously. Compare your findings with the ones obtained previously in terms of accuracy (note: the results should be identical) and time.

3. Use a sub-sampling approach in order to estimate the regression coefficients. For each random split of the data derive an estimate and its standard error. Weight the different estimates with the inverse of the variance to report a weighted estimate. Compare your findings to the ones obtained previously.

Solution

In order to complete this exercise, we are going to use the following packages, namely:

- **biglm** : Which will help us build models.
- **data.table** : It is 4 times faster than any other library that reads data and as a result, will make our job run faster.
- **beepr** : This will notify us when a process that requires times has finish running (personal recommendation : sound = 4)
- **dplyr** : This will help us modify data
- **boot** : This package will assist us with the last part of this exercise where we are going to use bootstrap method.

Following the process described in the first part of the exercise, we end up with a dataset, with dimensions 2000000×120. In order to assure similarity accross all the times, I run the cose, I have set the seed to be

```
set.seed(3622114)
```

In order to read the data we are going to use*fread* function where apparently the time needed is only 39.88638 secs. Moving forward, given the dimensions, we are not able to run in one line linear regression as a result we stick to the following strategy:

1. Define a chunk size (10000)
2. Split the data accordingly
3. Use the first chunk in order to fit a linear model as usual
4. Finally, define a *for-loop* where in each iteration, we use the function *update* to update the coefficients of the regression model fitted earlier.

By the of the steps described earlier we end up in the model with the following summary (due to space limitation, here we provide the first 10 lines of the summary output:

```
Large data regression model: bigglm(fit, matrix_list[[1]])
Sample size =  2e+06
                Coef    (95%     CI)    SE p
(Intercept)  0.4120  0.4106  0.4134 7e-04 0
x1           0.6082  0.6068  0.6096 7e-04 0
x2          -1.5170 -1.5184 -1.5155 7e-04 0
x3           0.7838  0.7824  0.7852 7e-04 0
x4          -0.0457 -0.0471 -0.0443 7e-04 0
x5          -0.1837 -0.1851 -0.1823 7e-04 0
x6          -1.2416 -1.2430 -1.2402 7e-04 0
...
```

Given the results, provided above we can say the following:

The coefficient Estimate contains two rows; the first one is the intercept. The intercept, in our example, is essentially the expected value of the distance required for a car to stop when we consider the average speed of all cars in the dataset. In other words, we can see that all the variables have the exact same standard error (and p-values 0) as for the coefficients we can say the following:

- If every value of x is 0 the value of y is 0.4120 with CI (0.4120,0.4134)
- For $x1$ if everything remain the same and $x1$ increases by 1 value, then y is going to decrease by 0.3918 (increase by 0.6082) with CI (0.6068,0.6096)
- For $x2$ if everything remain the same and $x2$ increases by 1 value, then y is going to decrease by 1.5170 with CI (-1.5184,-1.5155)
- The interpretation is the same for the rest of the coefficients.

Moving forward, to the next question, in order to calculate the estimated betas using the formula and not the function *summary*, initially we need to create a matrix Y containing the first column and a second matrix X containing in the first column *Ones* with the rest being the *x's*.

The way that it is calculated is the following:

```
betas <- solve(t(X_mat)%*%X_mat)%*%t(X_mat)%*%Y_mat
```

and we end up with to the following results:

```
                y
V1     0.41198072
x1     0.60822588
x2    -1.51695343
x3     0.78378959
x4    -0.04570068
x5    -0.18371295
x6    -1.24157476
```

As we can see the coefficients are the same whichever methods the user uses.

In the last part of this exercise in order to complete a meta-analysis, we decided to use bootstrap method. Each times the for loop run (100 times in total) a new model produces its coefficients and they are stored in a matrix. Later on we can see that the estimates are pretty close to the ones produces in the first part of this exercise. This is done by checking the last five coefficients by using the *tail* function.

2 Exercise 2

(Big Data Regression: airlines dataset). Download the data from the airlines dataset :

http://stat-computing.org/dataexpo/2009/the-data.html

The data provides arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008. Use the data for a year with the same last digit with your AM. You want to fit a model for arrival delay, in minutes using as covariates

- The month
- The weekday
- The distance
- The departure delay
- The departure time

Describe the model you estimated and write a short report explaining what you see.

Solution

In this exercise we are going to use the following libraries:

- **R.Utils**
- **car**
- **ggplot2**
- **tidyr**
- **biglm**
- **data.table**
- **dplyr**
- **beepr**

The dimensions of the data for this exercise are 7129270×29. From those we are going to keep only the variables that are stated in the exercise's definition. Namely:

i. ArrDelay

ii. Month

iii. DayOfWeek

iv. Distance

v. DepDelay

vi. DepTime

If we take a closer look at the data provided by using *summary* function we are going to see that there exist some negative values in the variable we are trying to model which Arrival Delay. From my perspective this does not make sense as a plane can not reach its destination before the expected arrival time as a result, I decided to remove those values. The new data now have dimensions 3018442×6. In order to model the arrival time, I am going to use a generalized linear model, with **Gamma** family. This can be easily be done with the help of **MASS** package Apart from that we can also use the function *ks.test*[1] in order to check the normality of the data. From the aforementioned test, we have got the following result:

```
    Asymptotic one-sample Kolmogorov-Smirnov test

data:  ex2_dt_12\$ArrDelay
D = 0.92686, p-value < 2.2e-16
alternative hypothesis: two-sided
```

As the dimensions of the dataset is big, we need to use once again the the same strategy as with exercise 1 and the *update* function. The difference is that now instead of *lm* function we are going to use *glm* function which is used to generate a generalised linear model with the usage of Negative Binomial family.

The results, are the following

```
Call:
glm(formula = ArrDelay ~ Month + DayOfWeek + Distance +
                                    DepDelay +
    DepTime, family = Gamma(link = "log"), data = matrix_nb[[1]
                                        ])

Coefficients: (1 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.589e+00  2.082e-02 124.335  < 2e-16 ***
Month               NA         NA      NA       NA
DayOfWeek   -1.106e-02  2.508e-03  -4.411 1.03e-05 ***
Distance     6.557e-05  9.270e-06   7.073 1.55e-12 ***
DepDelay     1.945e-02  1.255e-04 154.923  < 2e-16 ***
DepTime      1.045e-05  1.153e-05   0.906    0.365
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[0]This package masks **dplyr**'s *select* function and as a result, after the usage it is recommended to detach it.

[1]I also tryied *shapiro.test* with no luck as the dimentions are above its capabilities

```
(Dispersion parameter for Gamma family taken to be 0.8240305)

    Null deviance: 47986  on 32999  degrees of freedom
Residual deviance: 24518  on 32995  degrees of freedom
AIC: 264350

Number of Fisher Scoring iterations: 8
```

We can see that the results, acquired make sense, as the variables that play an important part in the arrival delay time are the distance and the departure delay and the day of week. By the results, we have that if the distance is increased by one measurement value, then the expected log units of arrival delay will be increased by 6.557e-05 in the case that the rest of the variables remain constant, the rest of the statistically significant variables can be interpreted similarly.

3 Exercise 3

(Communities and Crime Data Set3). Find the data in :
http://archive.ics.uci.edu/ml/datasets/communities+and+crime+unnormalized#
including some description about them. The task is to find a model in order to describe the response variable

1. **murders**: number of murders in 1995

Note that you need some pre-processing of the data to remove some variables that are not useful. Be as detailed as possible so as your report to be self-explained. Explain why you selected the specific model, how good you think it is and any limitations that may apply.

Solution

In this exercise we are going to use the following packages:

- **biglm**
- **data.table**
- **dplyr**
- **beepr**
- **glmnet**

Initially, we load the data, and we see that the dimensions are 2215×147. One can also notice that there is a character value **'?'** in the dataset which needs to be replaced by NA in order to be recognised as such. We also set the column with the ones provided in the dataset source. Next we decide to remove the null values and use the Lasso algorithm for different values of lambda (0.05,0.1,0.2,0.3,0.4,0.5,1.0,1.5,2.0,2.5) in order to select the most important of the 146 variables in order to model variable *murders*. The results acquired are the following:

λ	Var.Selected
0.05	12
0.1	12
0.2	12
0.3	12
0.4	12
0.5	12
1.0	10
1.5	8
2.0	7
2.5	5

By running the models with the selected models provided we see that the best among them is the one with 12 selected variables, namely:

i. communityCode
ii. blackPerCap
iii. OtherPerCap
iv. HispPerCap
v. PctNotSpeakEnglWell
vi. NumInShelters
vii. LemasSwornFT
viii. RacialMatchCommPol
ix. PolicAveOTWorked
x. murdPerPop
xi. robberies
xii. assaults

The model generated using those variables is a poisson (as we have a count variable) generalised linear model built as follows:

```
glm(data = dt3, murders ~ communityCode + blackPerCap +
                                  OtherPerCap + HispPerCap +
                                  PctNotSpeakEnglWell +
                                  NumInShelters + LemasSwornFT
                                  + RacialMatchCommPol +
                                  PolicAveOTWorked + murdPerPop
                                   + robberies + assaults,
                                  family = poisson(link = 'log'
                                  ))
```

with the following summary:

```
Call:
glm(formula = murders ~ communityCode + blackPerCap +
                                  OtherPerCap +
    HispPerCap + PctNotSpeakEnglWell + NumInShelters +
                                     LemasSwornFT +
    RacialMatchCommPol + PolicAveOTWorked + murdPerPop +
                                     robberies +
    assaults, family = poisson(link = "log"), data = dt3)

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)          4.399e+00  2.300e-01  19.124  < 2e-16 ***
communityCode       -6.740e-06  9.479e-07  -7.110 1.16e-12 ***
blackPerCap         -7.754e-05  1.075e-05  -7.212 5.52e-13 ***
```

```
OtherPerCap           3.349e-05  1.490e-05   2.247 0.024664 *
HispPerCap           -1.234e-04  1.370e-05  -9.008  < 2e-16 ***
PctNotSpeakEnglWell -7.970e-02  8.301e-03  -9.602  < 2e-16 ***
NumInShelters         1.780e-04  2.928e-05   6.080 1.20e-09 ***
LemasSwornFT          1.635e-04  3.582e-05   4.564 5.02e-06 ***
RacialMatchCommPol   -7.648e-03  2.323e-03  -3.292 0.000995 ***
PolicAveOTWorked      2.437e-03  2.467e-04   9.879  < 2e-16 ***
murdPerPop            3.783e-02  9.066e-04  41.727  < 2e-16 ***
robberies             2.333e-04  1.669e-05  13.972  < 2e-16 ***
assaults             -1.844e-04  8.832e-06 -20.878  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 17075.2  on 140  degrees of freedom
Residual deviance:  1113.8  on 128  degrees of freedom
  (2074 observations deleted due to missingness)
AIC: 1633.7

Number of Fisher Scoring iterations: 5
```

The coefficient here can be interpreted as the models produced earlier. For example, if everything remain constant and the assaults are raised by one unit on the measurement scale then the murders will by affected negatively (a.k.a. decreased) by -1.844e-04. The rest of the variables included in the model can be interpreted in the exact same way.