CHALLENGE B

Gabriel SAIVE & Florence LAURENS

30 novembre 2017

https://www.github.com/gabrielsaive/ChallengeB

TASK 1B - Predicting house prices in Ames, Iowa (continued)

Step 1

Choose a ML technique : non-parametric kernel estimation, random forests, etc… Give a brief intuition of how it works. (1 points)

We choose the following ML technique : **randomForest**. Let us explain how it works. As the name suggest, this algorithm creates the forest with a number of trees.

In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results.

The principle : to average the forecasts of several independent models to reduce the variance and therefore the forecast error. To build these different models, we select several bootstrap samples, that is to say prints with discounts.

Step 2

Train the chosen technique on the training data. Hint : packages np for non-parametric regressions, randomForest for random forests. Don't use the variable Id as a feature. (2 points)

Let us import the data base **train** and then thee **test** data base :

```
train <- read.csv(file = "data/train.csv")
test <- read.csv(file = "data/test.csv")
```

We install some libraries needed for our code : tidyverse, np, randomForest and dplyr.

```
##    tidyverse           np      markdown randomForest        dplyr
##         TRUE         TRUE         TRUE         TRUE         TRUE
```

We remove the missing variables from our **train** database :

```
remove.vars <- train %>% summarise_all(.funs = funs(sum(is.na(.)))) %>%
  gather(key = "feature", value = "missing.observations") %>%
  filter(missing.observations > 100) %>% select(feature) %>% unlist
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.3
```

```
train <- train %>% select(- one_of(remove.vars))
train <- train %>% filter(is.na(GarageType) == FALSE,
                          is.na(MasVnrType) == FALSE,
                          is.na(BsmtFinType2) == FALSE,
                          is.na(BsmtExposure) == FALSE,
                          is.na(Electrical) == FALSE)
```

Let us compute our machine learning method **random Forest** to our database.

Firstly we load the package "randomForest".

Now, we are able to regress SalePrice on our explanatory variables without Id using the function randomForest :

```
model<-randomForest(SalePrice~.-Id,data=train)
model
```

```
##
## Call:
##  randomForest(formula = SalePrice ~ . - Id, data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 24
##
##           Mean of squared residuals: 804678754
##                     % Var explained: 87.07
```

We obtain a model with an explained variance around 87.44%. To compare with the linear regression model we had a model explaining 91.41% of the variance of the residuals.

```
summary(lm(SalePrice~.-Id,data=train))$adj.r.squared
```

```
## [1] 0.9141072
```

Step 3

Make predictions on the test data, and compare them to the predictions of a linear regression of your choice. (2 points)

For some variables, the levels (meaning the values that a factor can take) of some factors are differents. To run the prediction we have to equalize the levels of the **train** database to the levels of the **test** database. This is needed for the following variables : Utilities, Condition2, HouseStyle, RoofMatl,Exterior1st,Exterior2nd,Heating,Electrical,GarageQual.

Let us run a prediction of the sale price from our test sample with the **random forest** regression, and do the same with a linear model:

```
prediction.Forest<-predict(model,test)

lm<-lm(SalePrice ~ MSZoning + LotArea + Neighborhood  + YearBuilt + OverallQual
                    , data = train)
prediction.lm<-predict.lm(lm,test)
```

We can now compare the result of the prediction with the **random forest** regression with the result of the linear regression:

```
data.frame<-data.frame(Id=test$Id,
                       "Prediction with Random Forest regression"=prediction.Forest,
                       "Prediction with Linear regression"=prediction.lm,
                    Differences=abs(prediction.Forest-prediction.lm),
                    Rapport=(prediction.Forest-prediction.lm)/prediction.lm)
head(data.frame)
```

```
##     Id Prediction.with.Random.Forest.regression
## 1 1461                                 125715.7
## 2 1462                                 157076.4
## 3 1463                                 182691.6
```

```
## 4 1464                              184976.6
## 5 1465                              195518.4
## 6 1466                              182648.9
##   Prediction.with.Linear.regression Differences     Rapport
## 1                          110073.2    15642.48  0.14210984
## 2                          171382.6    14306.19 -0.08347513
## 3                          142314.0    40377.64  0.28372225
## 4                          171336.5    13640.17  0.07961043
## 5                          295944.8   100426.34 -0.33934150
## 6                          169593.8    13055.07  0.07697846
```

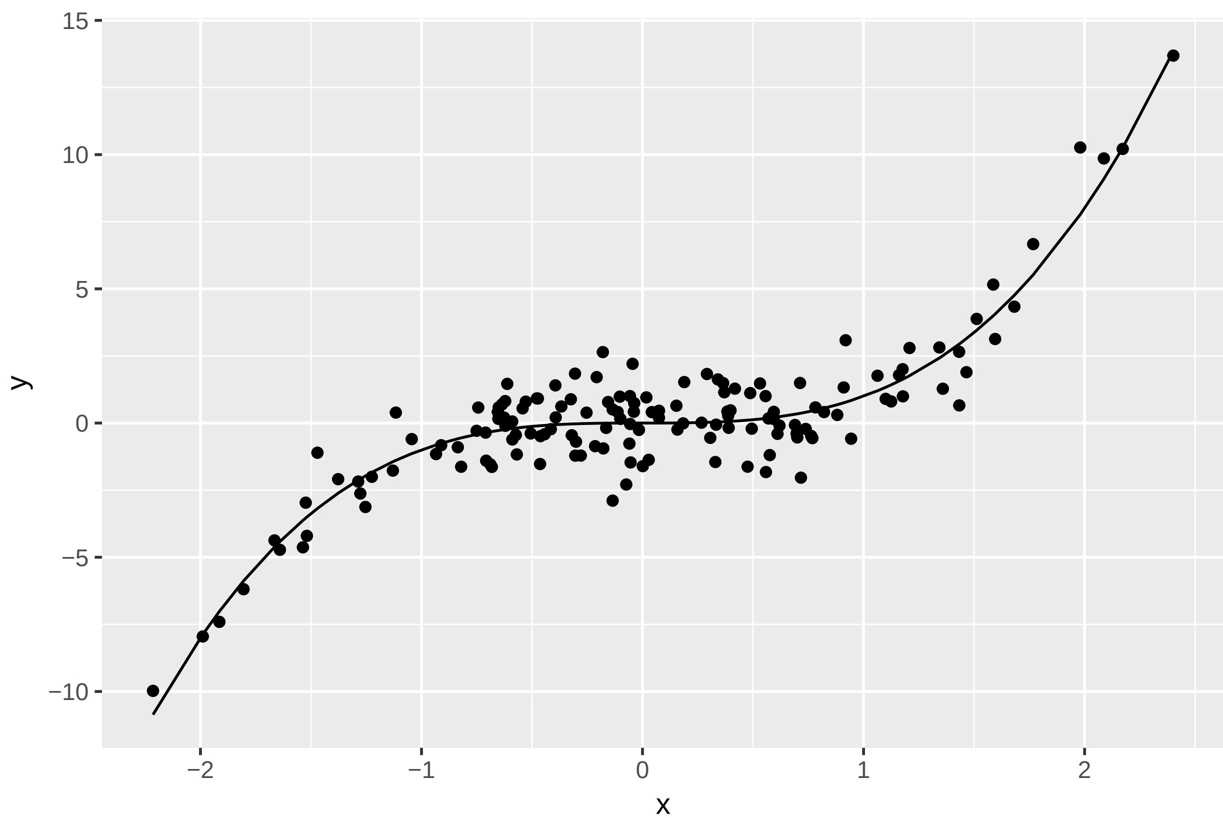
Task 2B - Overfitting in Machine Learning (continued)

Step 1

Estimate a low-flexibility local linear model on the training data. For that, you can use function npreg the package np. Choose ll for the method (local linear), and a bandwidth of 0.5; Call this model ll.fit.lowflex

We simulate 150 independant draws of x and y and put them in a table. We create our training and test dataset by slicing our table (respectively 80% and 20%).

This is how it looks like on a graph:



Step 1

We estimate a low-flexibility local linear model on our training data with a bandwidth of 0.5.

```
ll.fit.lowflex <- npreg(y ~ x, data = training, method = "ll", bws = 0.5)
summary(ll.fit.lowflex)
```

```
##
## Regression Data: 120 training points, in 1 variable(s)
##                 x
## Bandwidth(s): 0.5
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
## Residual standard error: 1.459384
## R-squared: 0.849058
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

Step 2

Estimate a high-flexibility local linear model on the training data. For that, you can use function npreg the package np. Choose ll for the method (local linear), and a bandwidth of 0.01; Call this model ll.fit.highflex

We estimate a high-flexibility local linear model on the training data with a bandwidth of 0.01:

```r
ll.fit.highflex <- npreg(y ~ x, data = training, method = "ll", bws = 0.01)
summary(ll.fit.highflex)
```

```
##
## Regression Data: 120 training points, in 1 variable(s)
##                  x
## Bandwidth(s): 0.01
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
## Residual standard error: 0.5589014
## R-squared: 0.9568912
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

Step 3

Plot the scatterplot of x-y, along with the predictions of ll.fit.lowflex and ll.fit.highflex, on only the training data. See Figure 1.

First we create predictions of y using the low and the high flexibility local linear model.

We can plot it : the blue line correspond to the prediction of y using the high flexibility model, the red one the prediction of y using the low flexibility model, and the black one is the true regression line.

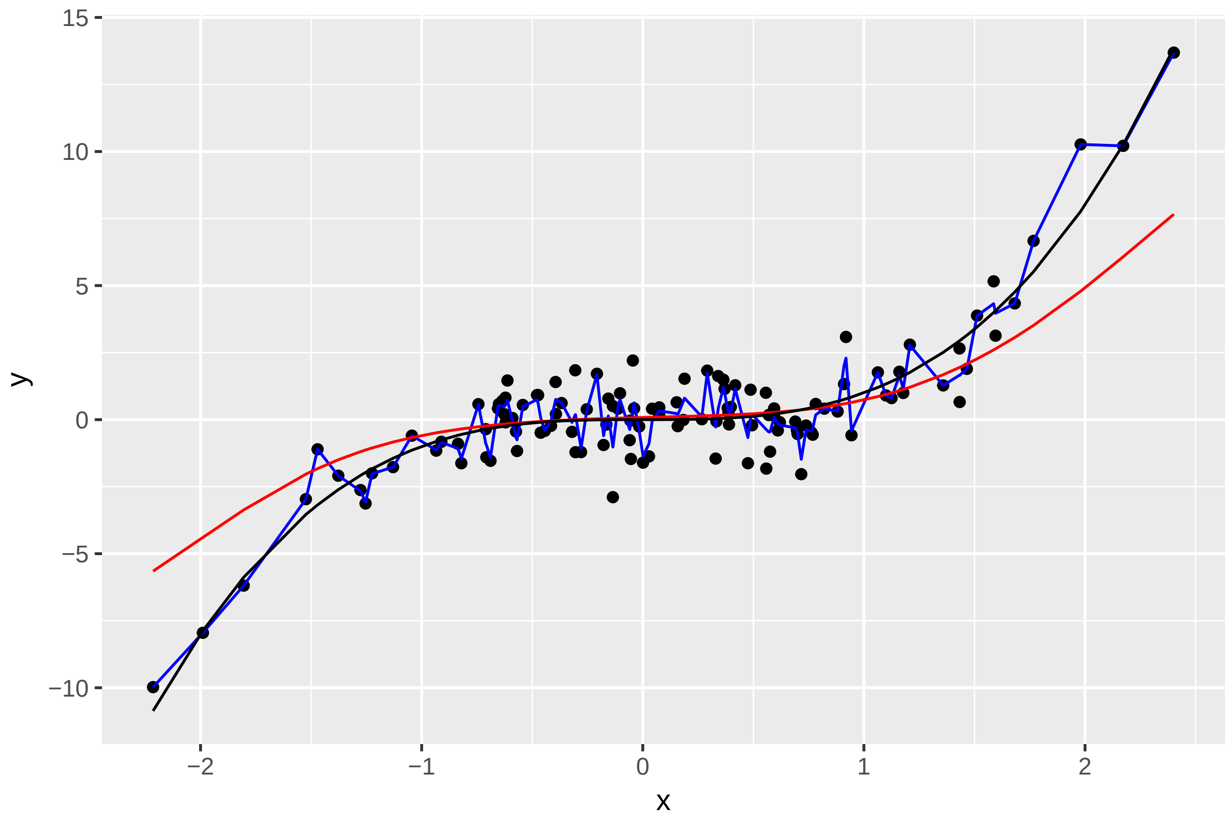


Figure 1: Step 3 − Predictions of ll.fit.lowflex and ll.fit.highflex on training data

Step4

Between the two models, which predictions are more variable? Which predictions have the least bias?

We compare the variance of y.ll.low.flex with the variance of y.ll.high.flex and the one of y.

```r
var(training$y.ll.lowflex)
```

```
## [1] 2.144139
```

```r
var(training$y.ll.highflex)
```

```
## [1] 6.846302
```

```r
var(training$y)
```

```
## [1] 7.292423
```

The y.ll.lowflex as the smallest variance, but this variance is too far from this one of y, so low flex prediction have the highest bias, in contrary to the high flex prediction, which has the greatest variance, and is the least biais. We can see this on the plot, its line is going through many points of y.

Step5

Plot the scatterplot of x-y, along with the predictions of ll.fit.lowflex and ll.fit.highflex now using the test data. Which predictions are more variable? What happened to the bias of the least biased model?

We create new predictions with the test dataset.

Let us plote these predictions:

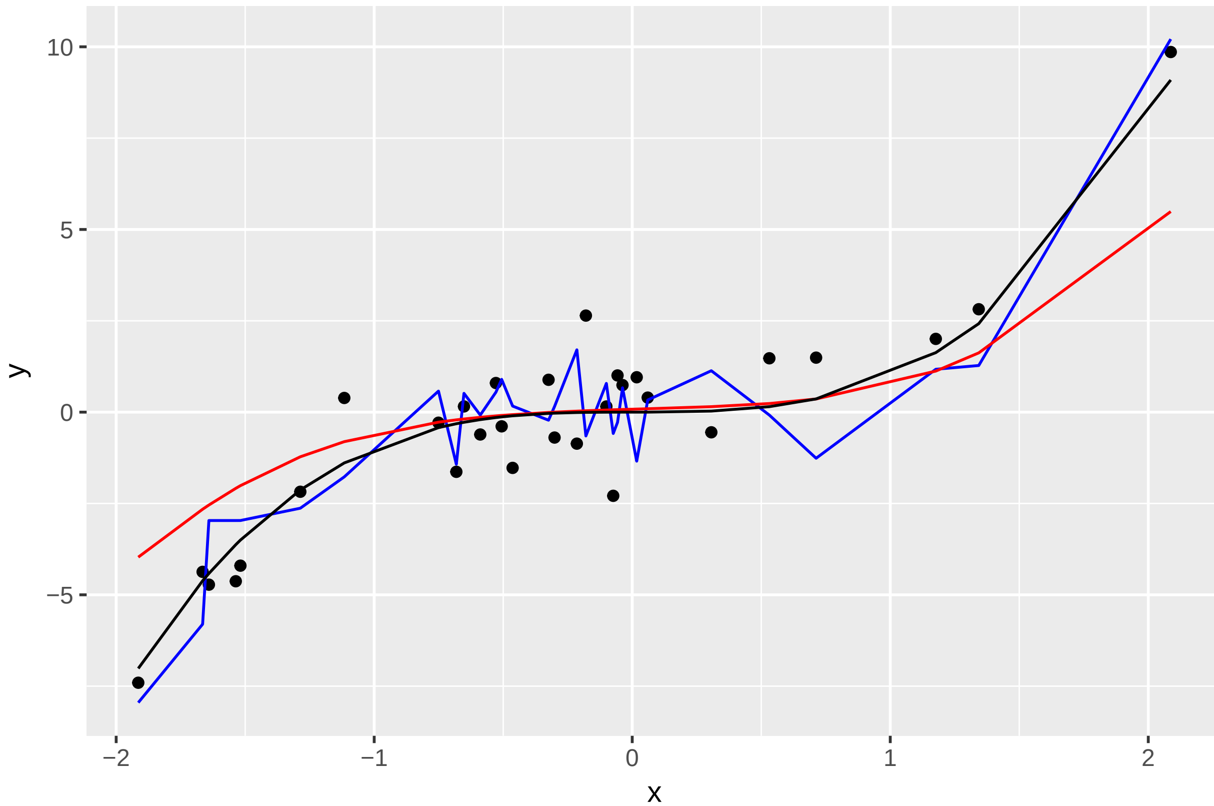


Figure 2: Step 5 − Predictions of ll.fit.lowflex and ll.fit.highflex on test data

```r
var(test$y.ll.lowflex)
```

```
## [1] 2.507889
```

```r
var(test$y.ll.highflex)
```

```
## [1] 8.571279
```

```r
var(test$y)
```

```
## [1] 9.430587
```

The y.ll.lowflex as the smallest variance, but this variance is too far from this one of y, so low flex prediction have the highest bias. The prediction is more biased than with the training data, the difference between the variance of the high flex model and y has increased.

Step 6

Create a vector of bandwidth going from 0.01 to 0.5 with a step of 0.001

Let us create vector of several bandwidths:

```r
bw <- seq(0.01, 0.5, by = 0.001)
```

Step 7

Train local linear model y ~ x on training with each bandwidth

We create a function doing a non parametric regression for every bandwidth. We apply in this function each bandwidth from **bw**, using **lapply**.

```
f<-function(bw) {npreg(y ~ x, data = training, method = "ll", bws = bw)}

train.bwd <- lapply(X = bw, f)
```

Step 8

Compute for each bandwidth the MSE-training

We create a function that predict y for every bandwidth using our training data. Then We get our mean square error by doing the squared difference between y and our predictions.

```
f2<-function(fit.model){
  predictions <- predict(object = fit.model, newdata = training)
  training %>% mutate(squared.error = (y - predictions)^2) %>% summarize(mse = mean(squared.error))
}
MSE.train<- unlist(lapply(train.bwd,f2))
```

Step 9

Compute for each bandwidth the MSE-test

We create a function that predict y for every bandwidth using our test data. Then We get our mean square error by doing the squared difference between y and our predictions.

```
f3 <- function(fit.model){
  predictions <- predict(object = fit.model, newdata = test)
  test %>% mutate(squared.error = (y - predictions)^2) %>% summarize(mse = mean(squared.error))
}
MSE.test <- unlist(lapply(X = train.bwd, f3))
```

Step 10

Draw on the same plot how the MSE on training data, and test data, change when the bandwidth increases. Conclude

First, we create a table with the bandwidth and our predictions for the training and the test data set.

```
mse.df <- data.frame(bw, MSE.train, MSE.test)
```

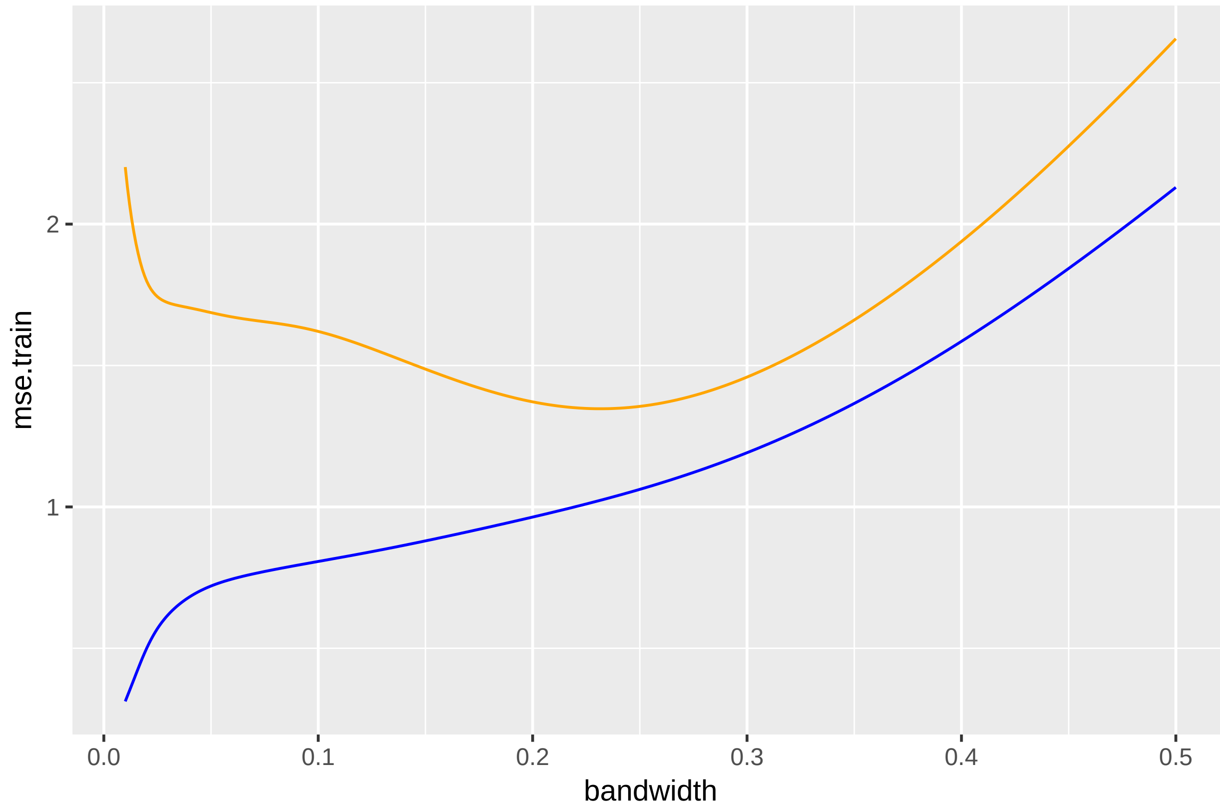


Figure 3: Step 10 − MSE on training and test data for different bandwidth − local linear regression

Task 3B - Privacy regulation compliance in France

Step 1

Import the CNIL dataset from the Open Data Portal. (1 point)

Let us import the CNIL data set :

```
CIL <- read.csv(file = "data/CNIL.csv",sep=";",header = T)
summary(CIL)
```

```
##     ï..Siren
##  Min.   :        0
##  1st Qu.:328921047
##  Median :413669820
##  Mean   :456909330
##  3rd Qu.:519605918
##  Max.   :999999999
##  NA's   :302
##                                            Responsable
##  N.F.L. DISTRIBUTION                              :   16
##  CENTRE COMMUNAL D'ACTION SOCIALE                 :    7
##  CHAMBRE DEPARTEMENTALE DES HUISSIERS DE JUSTICE:    7
##  BAYER S.A.S.                                     :    6
##  CAISSE PRIMAIRE D'ASSURANCE MALADIE              :    6
##  MAIRIE                                           :    6
```

```
##  (Other)                                         :18496
##                 Adresse       Code_Postal          Ville
##  11 RUE EMILE BRAULT:  206   75008  :  347   PARIS     : 1871
##  MAIRIE             :  195   75009  :  272   LYON      :  226
##  115 RUE DE LA SANTE:   92   53000  :  222   LAVAL     :  222
##                     :   61   75013  :  147   MARSEILLE :  180
##  8 AVENUE DELCASSE  :   38   75017  :  134   TOULOUSE  :  108
##  LE BOURG           :   29   75015  :  127   STRASBOURG:   96
##  (Other)            :17923   (Other):17295   (Other)   :15841
##                                                                                     NAF
##  6910Z ActivitÃ©s juridiques                                                          :2689
##  741A ActivitÃ©s juridiques, comptables et de conseil de gestion                      :2379
##  8810A Action sociale sans hÃ©bergement pour personnes Ã¢gÃ©es et pour personnes handicapÃ©es:1794
##  8411Z Administration gÃ©nÃ©rale, Ã©conomique et sociale                               : 746
##                                                                                       : 613
##  4711D Commerce de dÃ©tail en magasin non spÃ©cialisÃ©                                 : 453
##  (Other)                                                                              :9870
##          TypeCIL           Portee
##              : 119   Etendue   :16928
##  EXTERNE     : 901   GÃ©nÃ©rale: 1316
##  INTERNE     :4359   Partielle :  300
##  MUTUALISE   :3757
##  PROFESSIONNEL:8918
##  SALARIE     : 490
##
```

Step 2

Show a (nice) table with the number of organizations that has nominated a CNIL per department.

Firstly, we clean the data base: we have to delete all the missing variables and then we create a new table “departement” by selection ONLY the two first number of *Code Postal*. In the **table** we have for each department the number of organization that has nomitated a CNIL.

```r
sum(is.na(CIL))
```

```
## [1] 302
```

```r
CIL.NA<-na.omit(CIL)

departement<-substr(CIL.NA$Code_Postal,start = 1,stop = 2)
CIL.NA$departement<-departement

table<-as.data.frame(table(departement))
colnames(table)<-c("Department","Number of CNIL")
table
```

```
##     Department Number of CNIL
## 1                          56
## 2            .              1
## 3           01            132
## 4           02            104
## 5           03             68
## 6           04             72
## 7           05             52
```

```
## 8          06          256
## 9          07           61
## 10         08           82
## 11         09           20
## 12         10          103
## 13         11           92
## 14         12           85
## 15         13          454
## 16         14          255
## 17         15           53
## 18         16          122
## 19         17          149
## 20         18           78
## 21         19           52
## 22         20           92
## 23         21          147
## 24         22          112
## 25         23           31
## 26         24           82
## 27         25          144
## 28         26          132
## 29         27          111
## 30         28           96
## 31         29          176
## 32         30          132
## 33         31          311
## 34         32           81
## 35         33          364
## 36         34          285
## 37         35          280
## 38         36           52
## 39         37          181
## 40         38          416
## 41         39           67
## 42         40          176
## 43         41           95
## 44         42          217
## 45         43          100
## 46         44          337
## 47         45          180
## 48         46           59
## 49         47          106
## 50         48           11
## 51         49          210
## 52         50          131
## 53         51          168
## 54         52           51
## 55         53          316
## 56         54          198
## 57         55           65
## 58         56          177
## 59         57          244
## 60         58           44
## 61         59          530
```

```
## 62              60              210
## 63              61               74
## 64              62              219
## 65              63              139
## 66              64              159
## 67              65               69
## 68              66              109
## 69              67              273
## 70              68              166
## 71              69              596
## 72              70               69
## 73              71              122
## 74              72              132
## 75              73              102
## 76              74              187
## 77              75             2054
## 78              76              287
## 79              77              223
## 80              78              283
## 81              79              133
## 82              80              155
## 83              81              117
## 84              82               65
## 85              83              196
## 86              84              129
## 87              85              195
## 88              86              157
## 89              87              113
## 90              88              126
## 91              89               90
## 92              90               22
## 93              91              223
## 94              92              932
## 95              93              310
## 96              94              291
## 97              95              176
## 98              97              247
## 99              98               26
## 100             BP                2
## 101             CE                1
## 102             CS                1
## 103             EC                1
## 104             F3                1
## 105             LI                1
## 106             LU                1
## 107             PA                2
## 108             W1                1
## 109             WC                1
```

We deleted 302 observations because of the NA.

Step 3

Merge the information from the SIREN dataset into the CNIL data. Explain the method you use

For this question, we combine the two data base. The SIREN dataset is very large , and long to download so we only dowload the columns which are interesting to solve the problem, which are the SIREN and the size of the compagny. It take this long to import the dataset:

```
timer <- system.time("tabAll"<-read.csv(file=file.choose(),
sep=";",dec=".",header = T, colClasses = c(NA,rep("NULL",78))))
timer
```

```
##    user  system elapsed
##  528.44    7.47  826.62
```

Which is around 9 min. We remove the duplicated rows:

```
tabAll<-tabAll[!duplicated(tabAll$SIREN),]
```

With the command "gather" we merge the information from the SIREN dataset into the CNIL data.

```
library(dplyr)
data2<-CIL.NA[( CIL.NA$"ï..Siren" %in% tabAll$SIREN),]
head(data2)
```

```
##    ï..Siren     Responsable                     Adresse Code_Postal
## 1 788349926 "LA RIVE BLEUE"              3/5 RUE BOILEAU       49100
## 2 421715731       01 DIRECT      58 AVENUE DE RIVESALTES       66240
## 3 409869708    01DB-METRAVIB       200 CHEMIN DES ORMEAUX       69760
## 4 444600464      1.2.3. SAS 57-59 -61 RUE HENRI BARBUSSE       92110
## 6 429621311      1000MERCIS         28 RUE DE CHATEAUDUN       75009
## 7 429621311      1000MERCIS         28 RUE DE CHATEAUDUN       75009
##          Ville
## 1       ANGERS
## 2 SAINT ESTEVE
## 3     LIMONEST
## 4       CLICHY
## 6        PARIS
## 7        PARIS
##                                                                              NAF
## 1                                            8790A Autres activitÃ©s d'hÃ©bergement social
## 2                                                    526B Commerce de dÃ©tail hors magasin
## 3                                         7120B ActivitÃ©s de contrÃ´le et analyses techniques
## 4                                     524C Autres commerces de dÃ©tail en magasin spÃ©cialisÃ©
## 6                            6201Z Programmation, conseil et autres activitÃ©s informatiques
## 7 6311Z Traitement de donnÃ©es, hÃ©bergement et activitÃ©s connexes ; portails internet
##         TypeCIL     Portee departement
## 1       INTERNE    Etendue          49
## 2       EXTERNE GÃ©nÃ©rale          66
## 3 PROFESSIONNEL    Etendue          69
## 4       EXTERNE    Etendue          92
## 6       INTERNE    Etendue          75
## 7     MUTUALISE    Etendue          75
```

Step 4

Plot the histogram of the size of the companies that nominated a CIL. Comment.

Let us plot the histogram of the size of the companies that nominated a CIL.

```
data3<-tabAll[(tabAll$SIREN %in% CIL.NA$"ï..Siren"),]
plot(factor(data3$LIBTEFEN),las=2,cex.names=0.5)
```