Challenge B

Vincent Larrieu - Joel Brehin

December 8, 2017

Link to gihub repository : https://github.com/VincentLarr/ChallengeB-

DISCLAIMER: Some of the data files are too big to be uploaded on github, this work was realized using a R project file created from a repository containing this document as well as all the datasets used. Through the course of running the code three prompts will ask for data input.
In the first one, one must choose "train" file from challenge A, the second is "test" file from challenge A and the third is the CIL Database named "OpenCNIL_Organismes_avec_CIL_VD_20171115". Finally in last exercise, step 3, the line number 406 must contain the name of the SIREN database on the user PC (in our case we kept the default "sirc-17804_9075_14209_201710_L_M_20171101_030132835.csv"). We assume that to work this file needs to be in the same repository than this RMD file.

TASK 1B

Step 1

Random forest is a learning method for regression, classification and other tasks. The goal is to create a model that predicts the value of a target variable based on several independent variables. It constructs a multitude of decision tree at training time and outputting the class that is the mean prediction of the individual tree. In case of a regression, each interior node represents one of the independent variables such as "lotsize" for instance. Each leaf represents a value of the target variable, which "price" in our case, given the value of all the independent variables. Finally, the different paths represents different combination of values of the independent variables.

Step2

Now we train the machine learning "randomForest" on the data train.

We have kept the model that we used in the Challenge A and we run the RandomForest machine learning on the data train.

```
RFtrain <- randomForest(SalePrice~ LotArea + OverallQual + OverallCond + YearBuilt
                        +X1stFlrSF+ X2ndFlrSF +GarageArea+ ScreenPorch , data=train)
```

step3

```
prediction.lr <- predict(model4, test)
```

I predict the Saleprice with the randomForest function:

```
predict.rf <- predict(RFtrain, test)
```

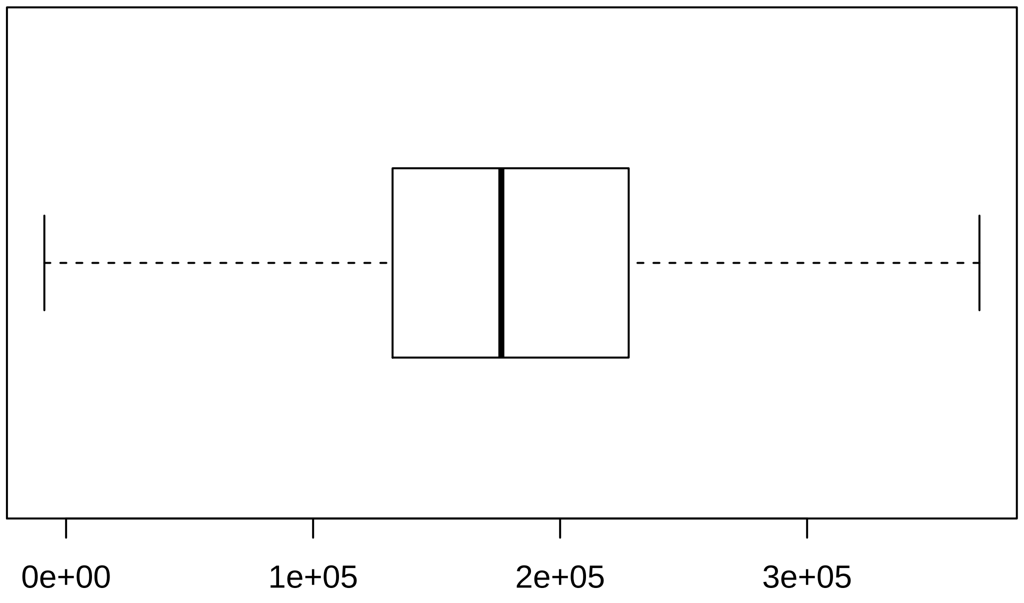
We can compare first with the two summaries :

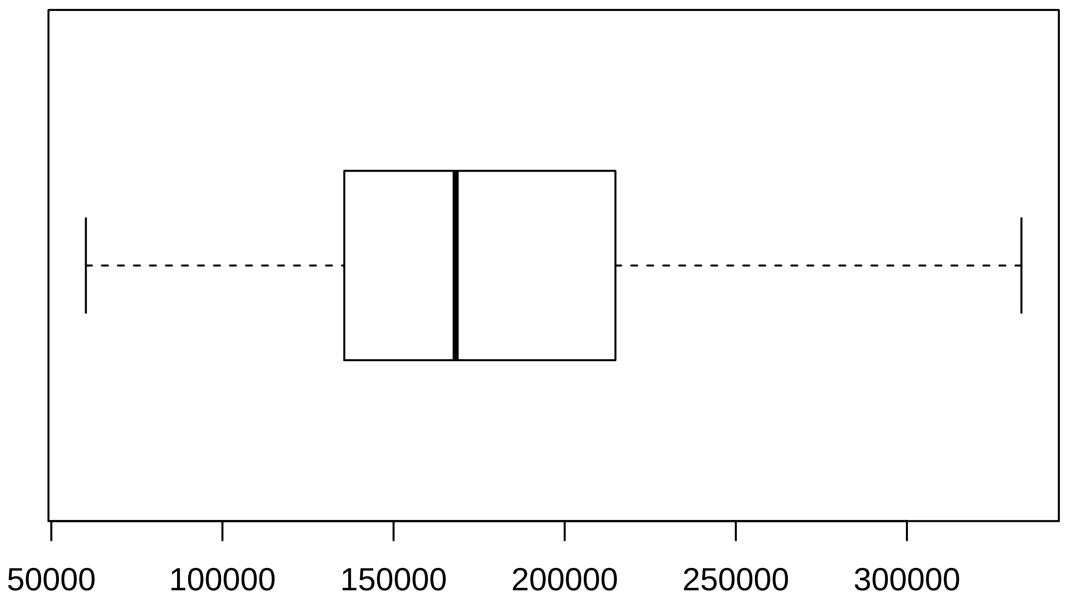
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   60107  135604  168137  185172  214832  513540
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -35497  132177  176215  183942  227767  605884
```

The prediction with the regression is more generous. Values are on average higher than the prediction with random forest. Moreover, the prediction with random forest doesn't give negative values compare to the prediction with the linear prediction, which is an enormous advantage because we try to predict a price.

We can also compare with box plot:





TASK 2B

Step1

We simply use the function npreg from the package np with the following arguments:

```
set.seed(1)
ll.fit.lowflex<-npreg(y~x,bws = 0.5,regtype="ll",data=training,newdata=training)
```

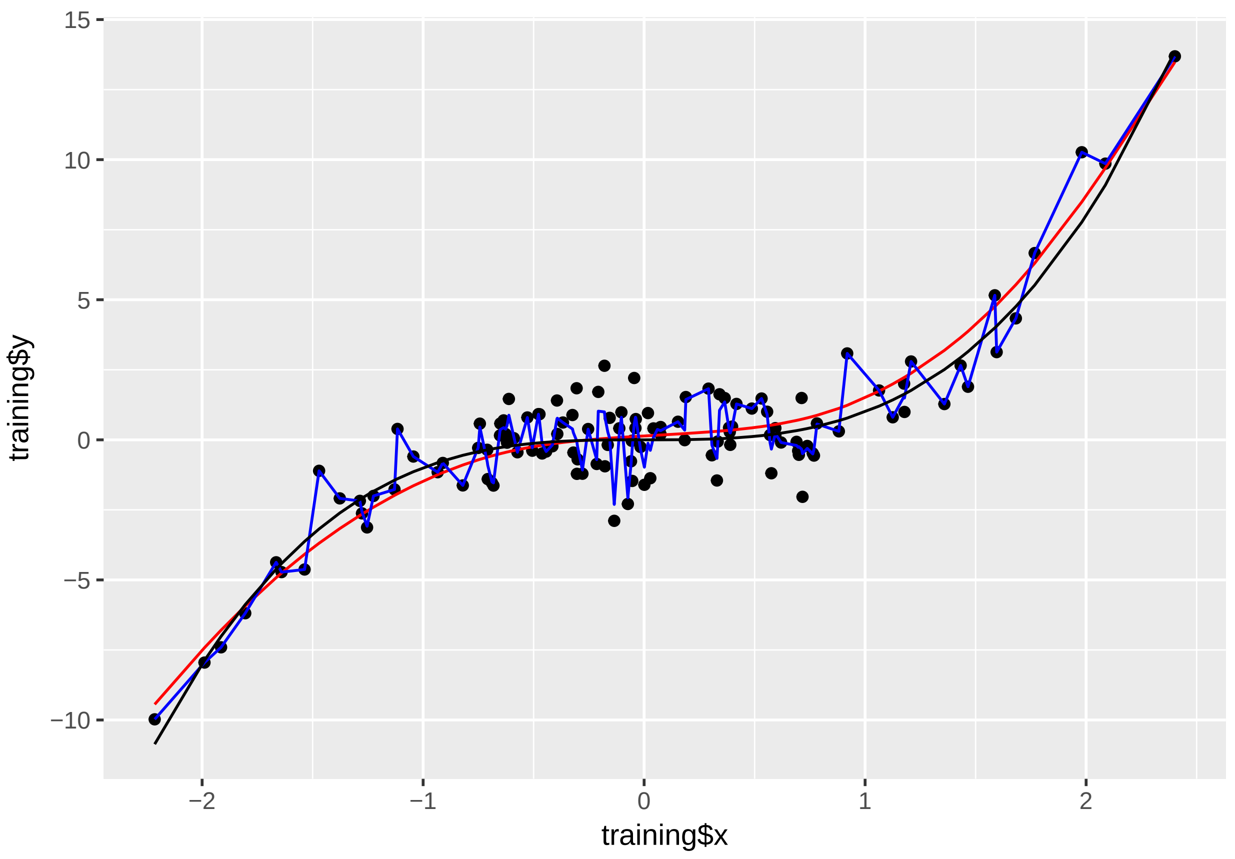
Step2

Same than previously, changing the bandwith

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

Step 3

We add columns to our training data frame containing the predictions using both regressions from the previous questions. Then we simply plot them as well as the points and the theoretical value x^3. In blue is the high flexibility line, in red the low flexibility one



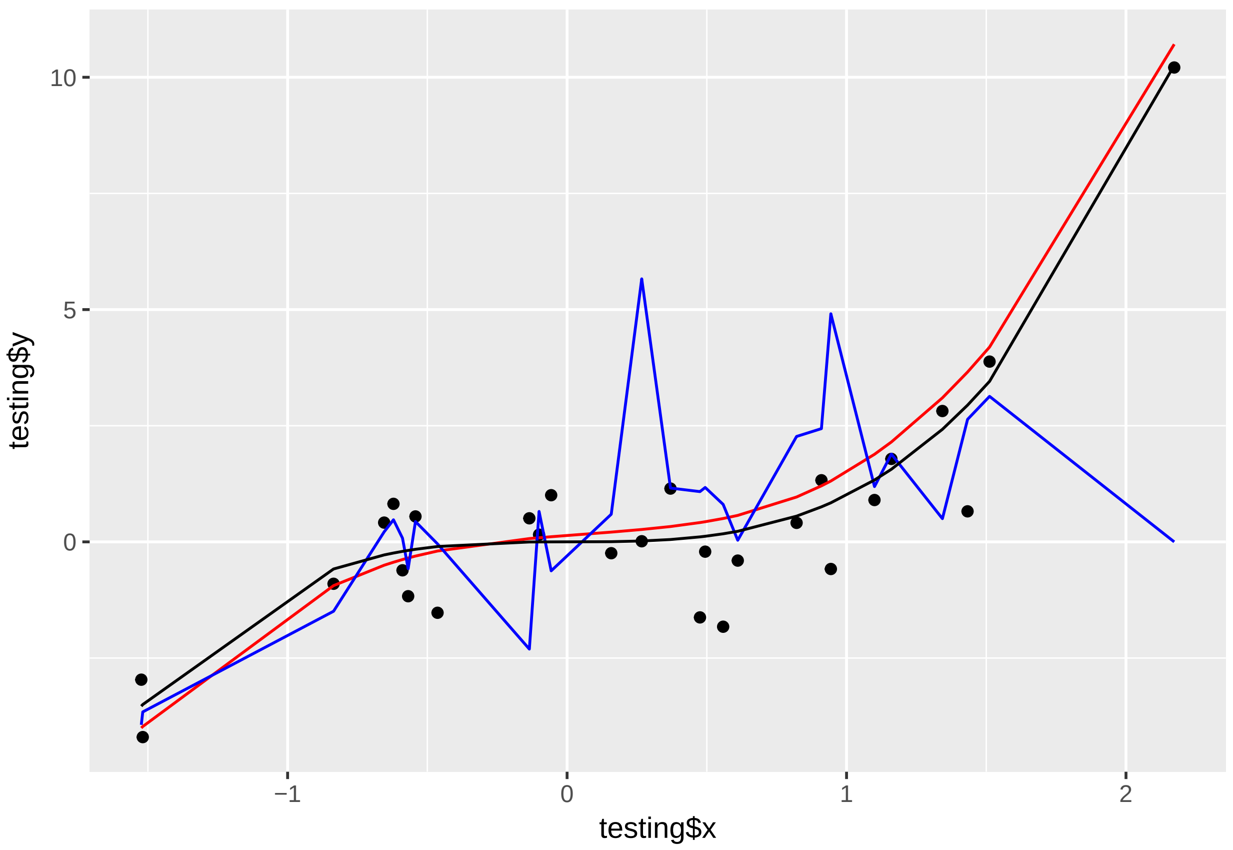
Step 4

The predictions from the high flexibility model have less bias, indeed the blue line is closer from the points for every x.However this blue line is also the more variable as it doesn't follow a trend close to the cube function, as is the case with the red line

Step 5

We proceed like in question 3 to build the plot, using the predictions of the old model on the new data(not making a new model with the new data)

```
testing$ll.fit.lowflex<-predict(object = ll.fit.lowflex, newdata=testing)
testing$ll.fit.highflex<-predict(object = ll.fit.highflex, newdata=testing)
```



We also see that the high-flexibility model is no longer the least-biased. For most points, the red line is closer from the dots than the blue one. Using a highly-flexible model on new data doesn't seem to be a good way to get a good prediction.

Step 6

We do this using a simple sequence command

```
bwsvec<-seq(0.01,0.5,0.001)
```

Step7

For each element in our bandwith vector we want to apply a npregfunction so we can do this using a lapply.

```
npreglist<-lapply(X=bwsvec, FUN=function(bwsvec){npreg(y~x, method="ll",
                                                       bws=bwsvec, data=training)})
```

Step8

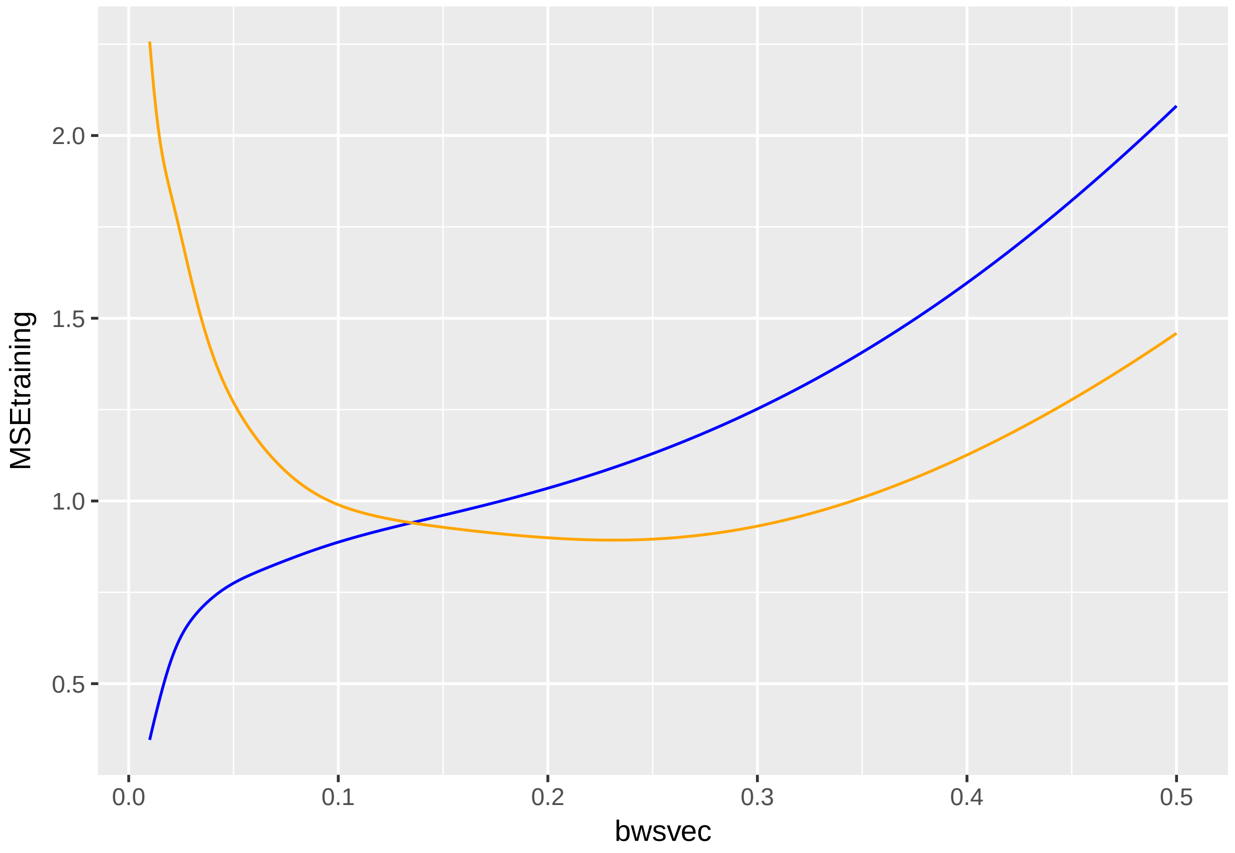
We first create a function which from a model input gives the MSE of this model on new data(here training),we name this function get MSE, from this we can just do a sapply using our list of regression as the object and the new function getMSE

Step9

We proceed similarly, except this time our "get MSE" function inputs the testing data as the new data.

Step10

We plot the MSE according to the bandwith. The orange line is the MSE in the testing data(the new data) and the blue one is the one on the training data (the one on which the model was fitted). MSE is a measure of bias, the lower the MSE, the lower the bias.
As we've seen before a higher flexibility(a higher bandwith) will reduce the bias with the training data, however this effect will stop after a certain bandwith and be replaced by an increasing bias. For the testing data, theeffect is the opposite, the higher the bandwith the morebiased the model, this is explained by the fact that fitting closely a model with some data doesn't mean that this relationship will hold with another data.



TASK 3B

Step1

We import the CIL dataset as usual being careful that the separator is the good one (here a semi column).

Step2

First we reduce the Postcodes to the first three digits(we need three to separate between mainland France and overseas territories).Indeed, overseas territories have a three digit code, for simplicity reasons, we focus on mainland France. We create a data partition for metropolitan France, excluding overseas territories with 3digits department numbers and French companies abroad (above post code 959).

We then reduce mainland postcodes to the two first digits

We then simply finish by having a loop coutning for each element of the department vectors, how many lines have this value in their departement column, and we get the following table:

```
##
## Departement    01  02 03 04 05  06 07 08 09  10 11 12  13  14 15  16  17
## Number of CIL 134 106 71 74 54 259 61 82 21 103 93 89 468 259 54 123 151
##
## Departement   18 19 20  21  22 23 24  25  26  27 28  29  30  31 32  33  34
## Number of CIL 85 54 96 149 114 32 84 145 137 114 96 183 136 317 83 371 291
##
## Departement    35 36  37  38 39  40 41  42  43  44  45 46  47 48  49  50
## Number of CIL 283 53 186 421 69 177 97 218 102 340 182 60 109 12 213 134
##
## Departement    51 52  53  54 55  56  57 58  59  60 61  62  63  64 65  66
## Number of CIL 172 51 316 207 65 179 246 45 543 212 75 220 142 162 71 110
##
## Departement    67  68  69 70  71  72  73  74   75  76  77  78  79  80  81
## Number of CIL 281 171 598 70 123 133 103 188 2089 299 224 287 135 158 118
##
## Departement   82  83  84  85  86  87  88 89 90  91  92  93  94  95
## Number of CIL 65 199 130 196 160 117 127 90 24 224 939 310 291 171
```

Step3

The dataframe is too large to store it in the RAM so we need to read it by chunck of 100000 observations.To do that we use a repeat function that:
* removes the oldest lines for duplicates SIREN numbers(first order command by date,then only move to a new dataset those who are not duplicates)
* Select the rows whose SIREN number is also is the CIL database

*Pastes this trimmed sample to an object call DataFinal that will contain the results of all sets. The size of the chunck can be changed if it is a too large number to be stored on the RAM.

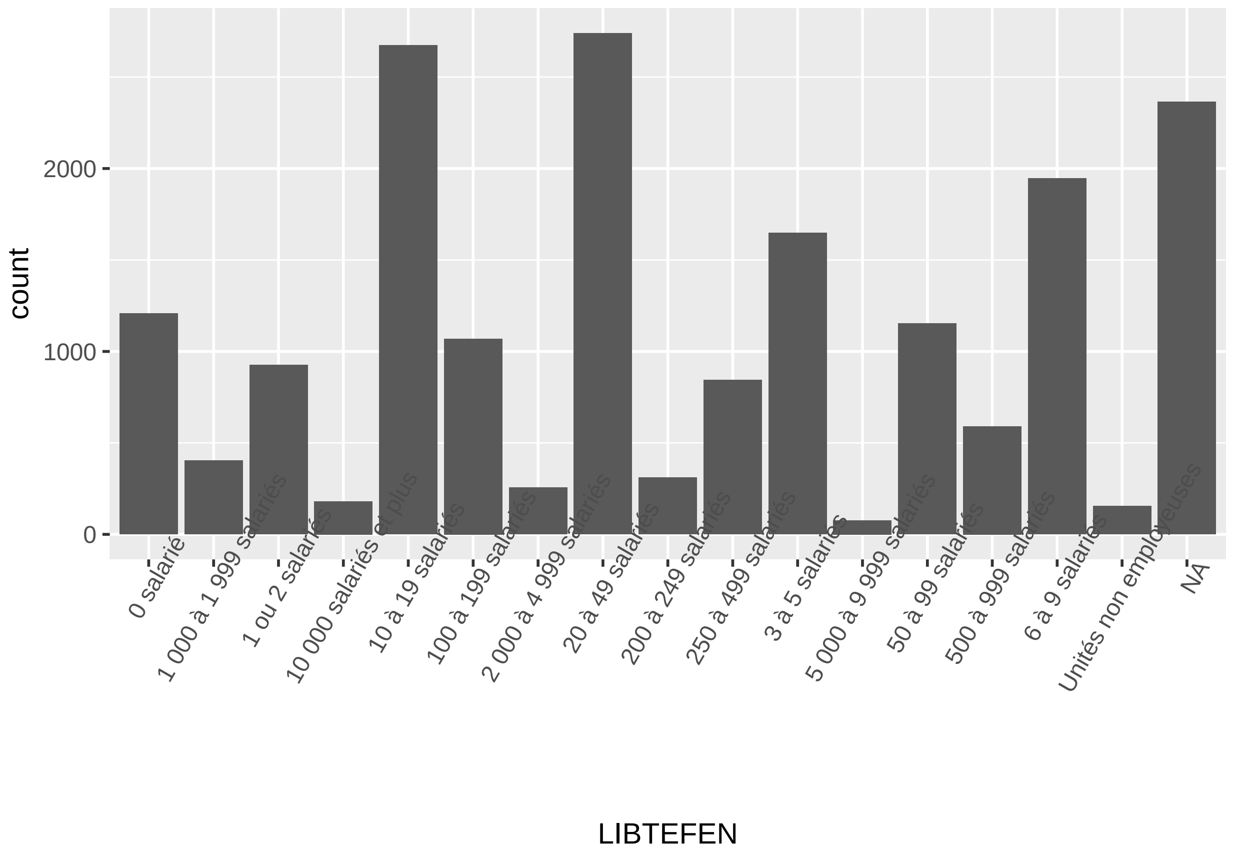
This takes this amount of time:

```
## Time difference of 13.56432 mins
```

We just have to bind the two datasets using the left_join command rom dplyr packages

Step4

We plot the histogram with ggplot 2, however we cannot order the bins as they are a character variable than cannot be ordered.


count
0 salarié
1 000 à 1 999 salariés
1 ou 2 salariés
10 000 salariés et plus
10 à 19 salariés
100 à 199 salariés
2 000 à 4 999 salariés
20 à 49 salariés
200 à 249 salariés
250 à 499 salariés
3 à 5 salariés
5 000 à 9 999 salariés
50 à 99 salariés
500 à 999 salariés
6 à 9 salariés
Unités non employeuses
NA
LIBTEFEN