Challenge B

Maxime GRANDMAITRE Gauderic THIETART Margaux SINCEUX

06/12/2017

Link :

https://github.com/Margauxsinceux/Challenge-B (https://github.com/Margauxsinceux/Challenge-B)

TASK 1B : Predicting house prices in Ames, Iowa

We clean our training database as we did in Challenge A : - we omit the NA - we convert all character variables into factors

Question 1: choose a ML technique

Random Forest is a Machine Learning algorithm wich is efficient to spot the links between one independent variable and some explanatory ones. Random Forest will classify the explanatory variables in function of their links with the variable we have to explain. It consists of doing the predictions average of multiple independent models to reduce the variance and so the prediction error.

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Question 2: Train the chosen technique on the training data

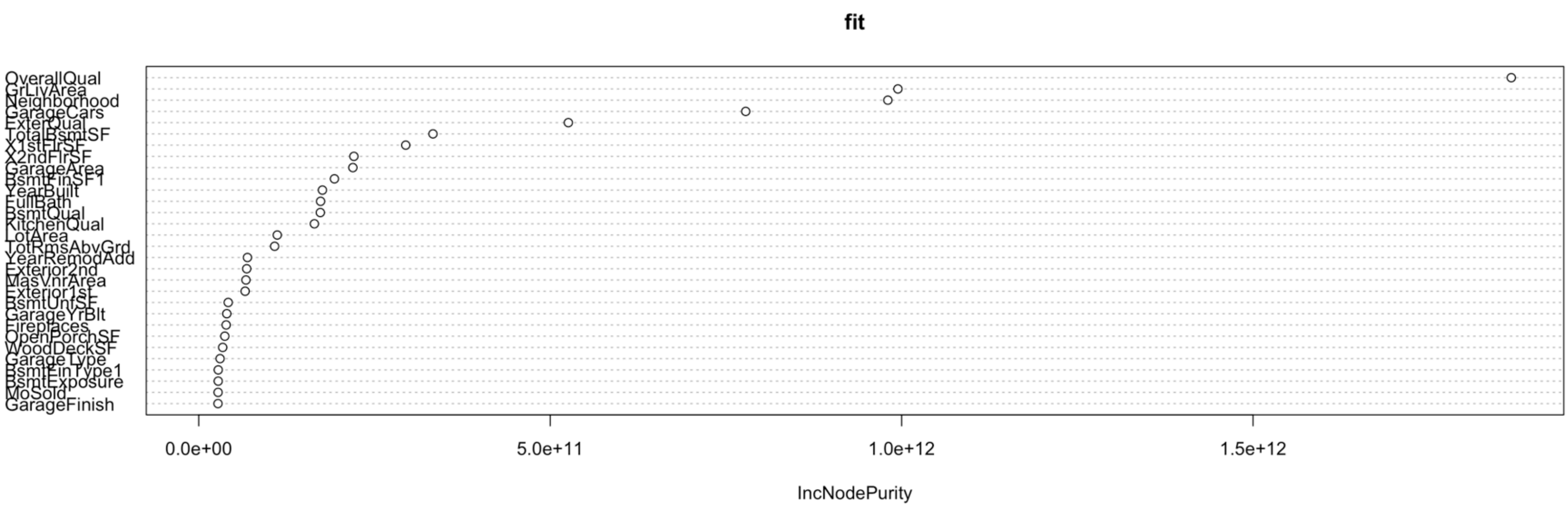
We use the random forest method by doing a regression, and deleting the “Id”.

```
set.seed(123)
fit <- randomForest(SalePrice~.-Id, data = train)
print(fit)
```

```
##
## 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: 792923183
##                     % Var explained: 87.26
```

We are doing varImpPlot to analyze which variables have more impact on the Sale Price.

```
varImpPlot(fit)
```



It is the same thing but in a table format.

```
fit$importance[order(fit$importance[, 1], decreasing = TRUE), ]
```

```
##   OverallQual     GrLivArea  Neighborhood    GarageCars     ExterQual 
##  1.867376e+12  9.947095e+11  9.804184e+11  7.781664e+11  5.258180e+11 
##   TotalBsmtSF     X1stFlrSF     X2ndFlrSF    GarageArea    BsmtFinSF1 
##  3.332303e+11  2.944926e+11  2.206979e+11  2.192564e+11  1.930225e+11 
##     YearBuilt      FullBath      BsmtQual   KitchenQual       LotArea 
##  1.759303e+11  1.733255e+11  1.729592e+11  1.645110e+11  1.116813e+11 
##  TotRmsAbvGrd  YearRemodAdd   Exterior2nd    MasVnrArea   Exterior1st 
##  1.079355e+11  6.927770e+10  6.823789e+10  6.712568e+10  6.596879e+10 
##     BsmtUnfSF   GarageYrBlt    Fireplaces   OpenPorchSF    WoodDeckSF 
##  4.190521e+10  3.988652e+10  3.900062e+10  3.706281e+10  3.403791e+10 
##    GarageType  BsmtFinType1  BsmtExposure        MoSold  GarageFinish 
##  3.032681e+10  2.771064e+10  2.766030e+10  2.747499e+10  2.735182e+10 
##   OverallCond      MSZoning  BedroomAbvGr    HouseStyle    MSSubClass 
##  2.451285e+10  1.941212e+10  1.902986e+10  1.714502e+10  1.664387e+10 
## SaleCondition   LandContour  BsmtFullBath     LotConfig        YrSold 
##  1.544624e+10  1.474222e+10  1.359481e+10  1.288784e+10  1.243108e+10 
##      LotShape     RoofStyle    MasVnrType      HalfBath    Condition1 
##  1.151920e+10  1.126130e+10  1.107057e+10  9.932382e+09  9.598063e+09 
##   ScreenPorch    Foundation      SaleType     HeatingQC    Functional 
##  9.253386e+09  9.039261e+09  8.946245e+09  8.555804e+09  6.498640e+09 
##    CentralAir      PoolArea     LandSlope EnclosedPorch      RoofMatl 
##  6.424842e+09  6.094925e+09  5.951381e+09  5.878999e+09  5.142328e+09 
##      BldgType  BsmtFinType2     ExterCond    BsmtFinSF2    GarageQual 
##  5.097134e+09  4.528857e+09  4.459573e+09  4.451311e+09  3.763106e+09 
##      BsmtCond  BsmtHalfBath  KitchenAbvGr    PavedDrive    Condition2 
##  3.723160e+09  3.464209e+09  2.973711e+09  2.005044e+09  1.923166e+09 
##    Electrical    GarageCond    X3SsnPorch  LowQualFinSF       Heating 
##  1.844362e+09  1.555179e+09  1.537852e+09  1.301425e+09  1.043248e+09 
##       MiscVal        Street     Utilities 
##  8.762859e+08  1.363551e+08  8.634947e+06
```

Question 3: Make predictions on the test data, and compare them to the predictions of a linear regression of your choice

We had to transform the factor variables because they have not the same levels in the data train and the data test. So we modified this to have the same levels.

```
levels(test$Utilities) <- levels(train$Utilities)
levels(test$Condition2) <- levels(train$Condition2)
levels(test$HouseStyle) <- levels(train$HouseStyle)
levels(test$RoofMatl) <- levels(train$RoofMatl)
levels(test$Exterior2nd) <- levels(train$Exterior2nd)
levels(test$Electrical) <- levels(train$Electrical)
levels(test$GarageQual) <- levels(train$GarageQual)
levels(test$Exterior1st) <- levels(train$Exterior1st)
levels(test$Heating) <- levels(train$Heating)


prediction <- data.frame(Id= test$Id, SalePrice_predict = predict(fit, test, type=
"response"))
```

We took the linear regression of challenge A to compare it with our new predicitons.

We compare thanks to a plot, and to check that it is not the same we did the average of the difference between them.

```
library(ggplot2)

predict2 <- data.frame(prediction, prediction_lm)
pred <- predict2[,-3]

plot <- ggplot(pred, aes (x = Id, y = SalePrice_predict)) +
  geom_point(data = prediction, aes(color="prediction")) +
  geom_point(data = prediction_lm, aes(color="prediction_lm"))

difference <- prediction_lm - prediction
summary(abs(difference[,2]))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##     13.99   7758.65  17260.04  22369.78  30767.40 112947.02
```

Task 2B : Overfitting in Machine Learning

```
rm(list = ls())

library(tidyverse)
library(np)
```

```
## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-3)
## [vignette("np_faq",package="np") provides answers to frequently asked questions
]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
# True model : y = x^3 + epsilon
set.seed(1)
Nsim <- 150
b <- c(0,1)
x0 <- rep(1, Nsim)
x1 <- rnorm(n = Nsim)

X <- cbind(x0, x1^3)
y.true <- X %*% b

eps <- rnorm(n = Nsim)
y <- X %*% b + eps

df <- tbl_df(y[,1]) %>% rename(y = value) %>% bind_cols(tbl_df(x1)) %>% rename(x =
value) %>% bind_cols(tbl_df(y.true[,1])) %>% rename(y.true = value)
```

We split sample into training and testing

```
training.index <- createDataPartition(y = y, times = 1, p = 0.8)
df <- df %>% mutate(which.data = ifelse(1:n() %in% training.index$Resample1, "trai
ning", "test"))

training <- df %>% filter(which.data == "training")
test <- df %>% filter(which.data == "test")
```

We do a linear regression

```
lm.fit <- lm(y ~ x, data = training)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = y ~ x, data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7575 -1.0695  0.0419  1.0229  7.6216
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1950     0.1649   1.183    0.239
## x             2.4446     0.1846  13.241   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.82 on 120 degrees of freedom
## Multiple R-squared:  0.5937, Adjusted R-squared:  0.5903
## F-statistic: 175.3 on 1 and 120 DF,  p-value: < 2.2e-16
```

```
df <- df %>% mutate(y.lm = predict(object = lm.fit, newdata = df))
training <- training %>% mutate(y.lm = predict(object = lm.fit))
```

Step 1 : lowflex

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

```
##
## Regression Data: 122 training points, in 1 variable(s)
##                 x
## Bandwidth(s): 0.5
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
## Residual standard error: 1.442574
## R-squared: 0.8569977
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

Step 2 : highflex

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

```
##
## Regression Data: 122 training points, in 1 variable(s)
##                  x
## Bandwidth(s): 0.01
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
## Residual standard error: 0.5882872
## R-squared: 0.9569811
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

Step 3 : plot highflex and lowflex in the training data

```
df <- df %>% mutate(y.ll.lowflex = predict(object = ll.fit.lowflex, newdata = df),
y.ll.highflex = predict(object = ll.fit.highflex, newdata = df))


training <- training %>% mutate(y.ll.lowflex = predict(object = ll.fit.lowflex, ne
wdata = training), y.ll.highflex = predict(object = ll.fit.highflex, newdata = tra
ining))

training
```

```
## # A tibble: 122 x 7
##              y          x       y.true which.data       y.lm y.ll.lowflex
##          <dbl>      <dbl>        <dbl>      <chr>      <dbl>        <dbl>
##  1  0.20433883 -0.6264538 -0.245848275   training -1.3363726 -0.200217871
##  2 -0.01236649  0.1836433  0.006193347   training  0.6439806  0.182352673
##  3  3.13050121  1.5952808  4.059863355   training  4.0948517  3.023714572
##  4 -1.45168388  0.3295078  0.035776429   training  1.0005590  0.236226060
##  5 -1.62750566 -0.8204684 -0.552313364   training -1.8106582 -0.425804071
##  6  1.11583565  0.4874291  0.115806846   training  1.3866113  0.310101940
##  7 -0.21878864  0.7383247  0.402478052   training  1.9999477  0.498338173
##  8 -1.19354142  0.5757814  0.190885432   training  1.6025962  0.363540869
##  9  1.84080947 -0.3053884 -0.028481152   training -0.5515002  0.007813978
## 10 -0.17939960  0.3898432  0.059247498   training  1.1480543  0.261909431
## # ... with 112 more rows, and 1 more variables: y.ll.highflex <dbl>
```

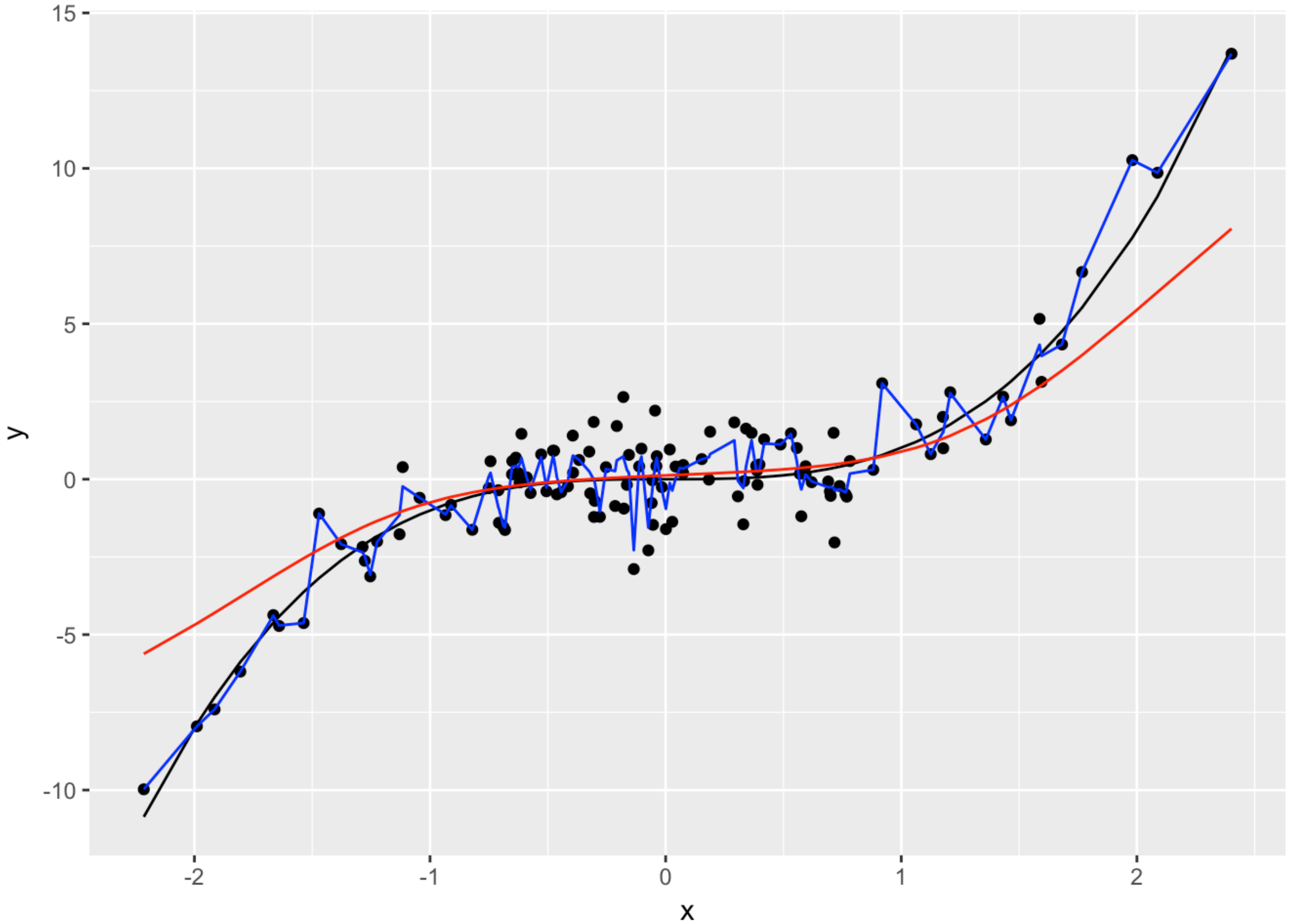
```
test <- test %>% mutate(y.ll.lowflex = predict(object = ll.fit.lowflex, newdata =
test), y.ll.highflex = predict(object = ll.fit.highflex, newdata = test))

test
```

```
## # A tibble: 28 x 6
##             y          x       y.true which.data y.ll.lowflex
##         <dbl>      <dbl>        <dbl>      <chr>        <dbl>
##  1 -0.9015671 -0.8356286 -0.583498718       test  -0.44815580
##  2  3.8802495  1.5117812  3.455149104       test   2.60289580
##  3  0.8187215 -0.6212406 -0.239761502       test  -0.19548025
##  4 -0.5837001  0.9438362  0.840794584       test   0.77420576
##  5  0.4094355  0.8212212  0.553835065       test   0.59241185
##  6  0.9008803  1.1000254  1.331092102       test   1.09962838
##  7  0.6575602  1.4330237  2.942795753       test   2.23958632
##  8  0.5076451 -0.1350546 -0.002463362       test   0.07595184
##  9 10.2105426  2.1726117 10.255251705       test   6.57995968
## 10 -1.6257013  0.4755095  0.107517132       test   0.30368108
## # ... with 18 more rows, and 1 more variables: y.ll.highflex <dbl>
```

We simulate Nsim = 100 points of (x,y)

```
ggplot(training) + geom_point(mapping = aes(x = x, y = y)) +
  geom_line(mapping = aes(x = x, y = y.true)) +
  geom_line(mapping = aes(x = x, y = y.ll.highflex), col="blue") +
  geom_line(mapping = aes(x = x, y = y.ll.lowflex), col="red")
```

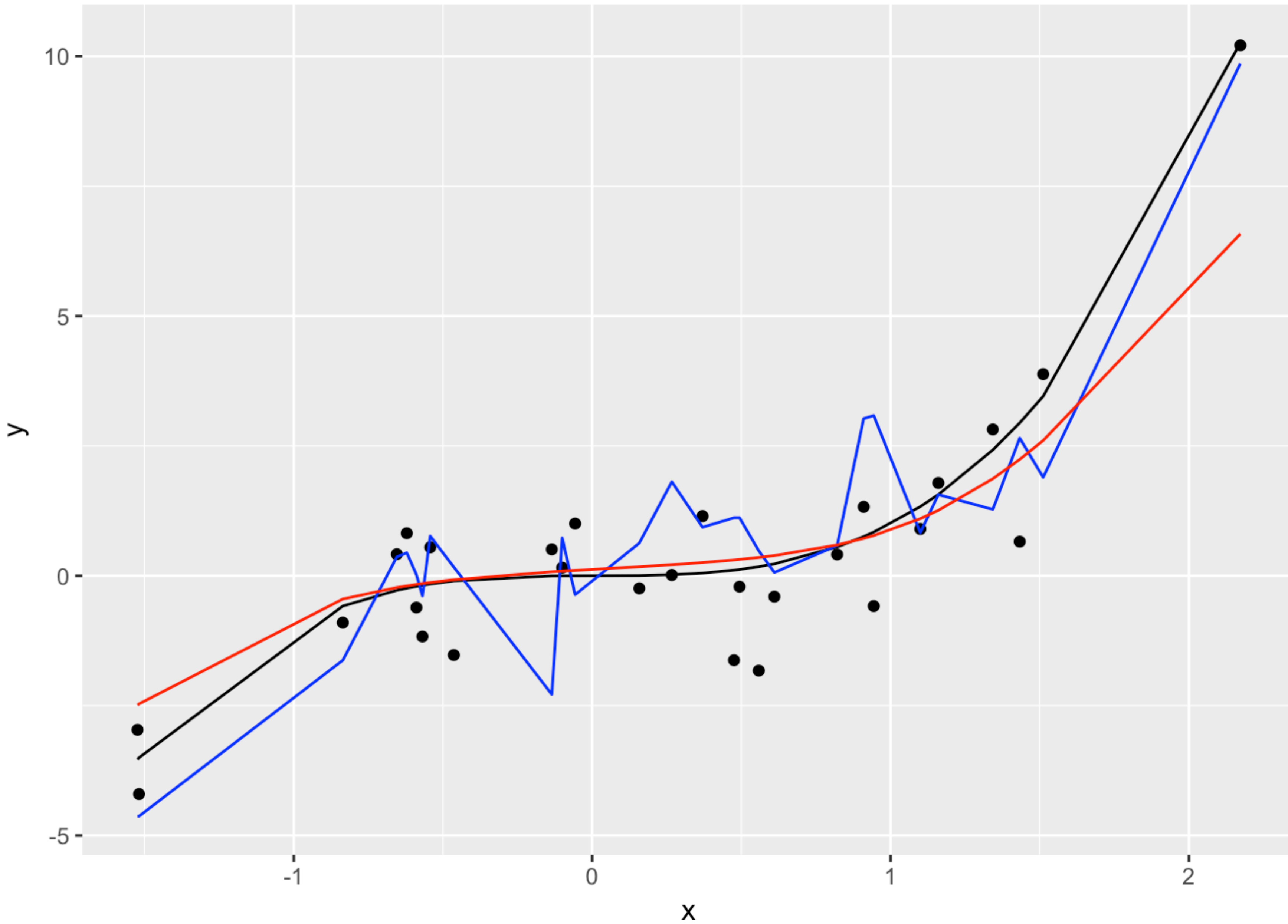


Step 4 : analysis

According to the plot, we can conclude that the predictions from ll.fit.highflex are more variable than the ones from ll.fit.lowflex but he has also least bias.

Step 5 : plot highflex and lowflex in the test data

```
ggplot(test) + geom_point(mapping = aes(x = x, y = y)) +
  geom_line(mapping = aes(x = x, y = y.true)) +
  geom_line(mapping = aes(x = x, y = y.ll.highflex), col="blue") +
  geom_line(mapping = aes(x = x, y = y.ll.lowflex), col="red")
```



We can see graphicaly than the predictions from ll.fit.highflex are more variable than the ones from ll.fit.lowflex when we are on the limit of the graph, but he has also least bias.

Step 6 : Create vector of several bandwidth

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

Step 7 : Train local linear model y ~ x on training with each bandwidth

```r
llbw.fit <- lapply(X = bw, FUN = function(bw) {npreg(y ~ x, data = training, metho
d = "ll", bws = bw)})
head (llbw.fit)
```

```
## [[1]]
##
## Regression Data: 122 training points, in 1 variable(s)
##                  x
## Bandwidth(s): 0.01
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
##
##
## [[2]]
##
## Regression Data: 122 training points, in 1 variable(s)
##                   x
## Bandwidth(s): 0.011
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
##
##
## [[3]]
##
## Regression Data: 122 training points, in 1 variable(s)
##                   x
## Bandwidth(s): 0.012
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
##
##
## [[4]]
##
## Regression Data: 122 training points, in 1 variable(s)
##                   x
## Bandwidth(s): 0.013
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
##
```

```
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
##
##
## [[5]]
##
## Regression Data: 122 training points, in 1 variable(s)
##                  x
## Bandwidth(s): 0.014
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
##
##
## [[6]]
##
## Regression Data: 122 training points, in 1 variable(s)
##                  x
## Bandwidth(s): 0.015
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

Step 8 : Compute for each bandwidth the MSE-training

```
mse.training <- function(fit.model){
  predictions <- predict(object = fit.model, newdata = training)
  training %>% mutate(squared.error = (y - predictions)^2) %>% summarize(mse = mea
n(squared.error))
}
mse.train.results <- unlist(lapply(X = llbw.fit, FUN = mse.training))
head (mse.train.results)
```

```
##       mse       mse       mse       mse       mse       mse
## 0.3460818 0.3708529 0.3947976 0.4181274 0.4408973 0.4630605
```

Step 9: Compute for each bandwidth the MSE-test

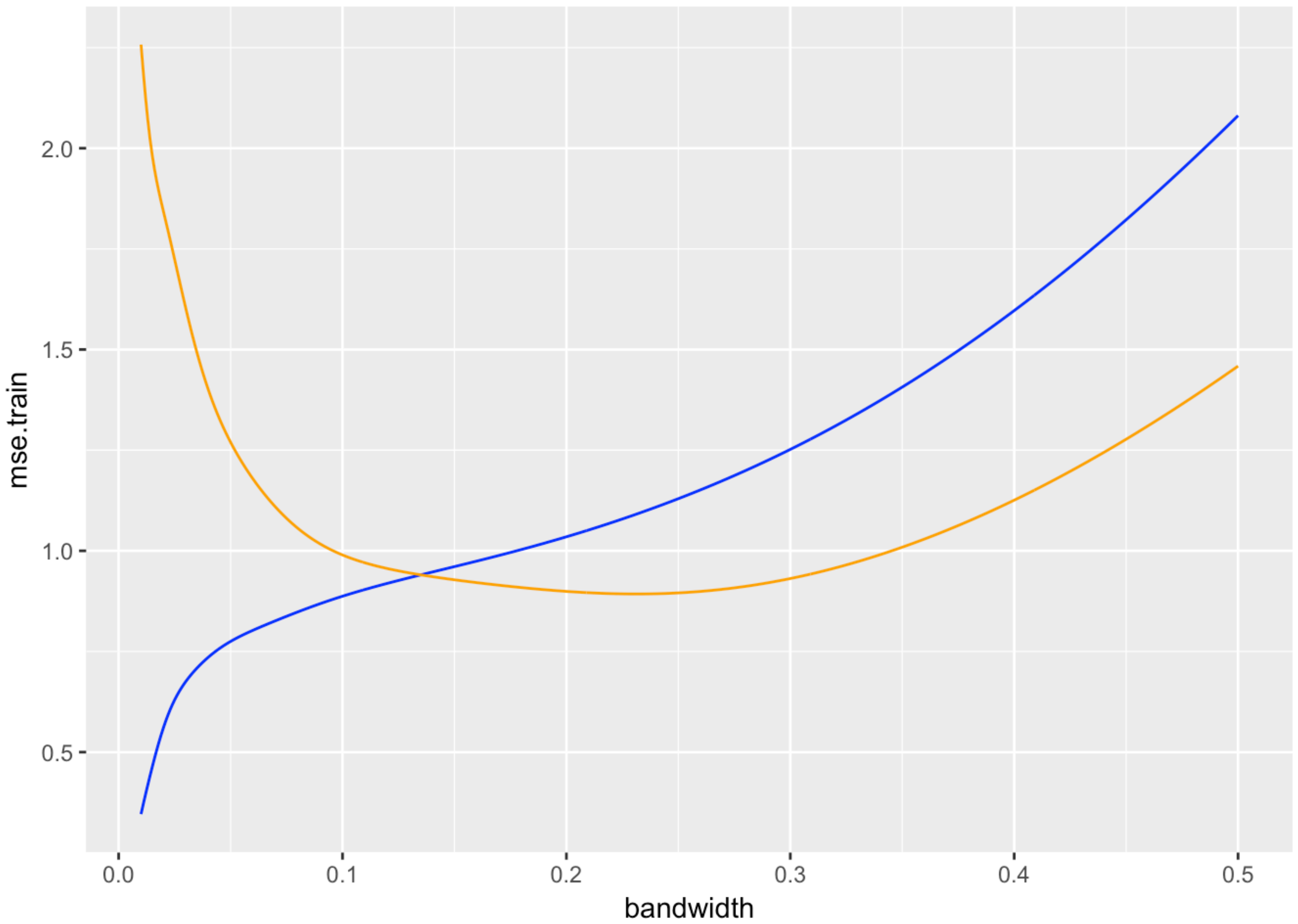
```
mse.test <- function(fit.model){
  predictions <- predict(object = fit.model, newdata = test)
  test %>% mutate(squared.error = (y - predictions)^2) %>% summarize(mse = mean(sq
uared.error))
}
mse.test.results <- unlist(lapply(X = llbw.fit, FUN = mse.test))
head (mse.test.results)
```

```
##      mse      mse      mse      mse      mse      mse 
## 2.257278 2.189654 2.126863 2.071231 2.023309 1.982573
```

Step 10 : Plot

```
## # A tibble: 491 x 3
##    bandwidth mse.train mse.test
##        <dbl>     <dbl>    <dbl>
##  1     0.010 0.3460818 2.257278
##  2     0.011 0.3708529 2.189654
##  3     0.012 0.3947976 2.126863
##  4     0.013 0.4181274 2.071231
##  5     0.014 0.4408973 2.023309
##  6     0.015 0.4630605 1.982573
##  7     0.016 0.4845110 1.947912
##  8     0.017 0.5051165 1.917947
##  9     0.018 0.5247438 1.891276
## 10     0.019 0.5432788 1.866645
## # ... with 481 more rows
```

```
ggplot(mse.df) +
  geom_line(mapping = aes(x = bandwidth, y = mse.train), color = "blue") +
  geom_line(mapping = aes(x = bandwidth, y = mse.test), color = "orange")
```



Task 3B : Privacy regulation compliance in France

Step 1 : Import Data

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.4.2
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
system.time(mat <- fread(file = "/Users/Margaux/Desktop/sirc-17804_9075_14209_2017
10_L_M_20171101_030132835.csv",sep =";", dec = ".", header = T, select = c("SIREN"
, "LIBTEFEN", "DATEMAJ")))
```

```
##
Read 0.0% of 10831176 rows
Read 1.8% of 10831176 rows
Read 4.2% of 10831176 rows
Read 6.2% of 10831176 rows
Read 8.1% of 10831176 rows
Read 9.8% of 10831176 rows
Read 11.7% of 10831176 rows
Read 13.8% of 10831176 rows
Read 14.8% of 10831176 rows
Read 16.6% of 10831176 rows
Read 18.6% of 10831176 rows
Read 20.4% of 10831176 rows
Read 22.5% of 10831176 rows
Read 23.1% of 10831176 rows
Read 24.7% of 10831176 rows
Read 26.3% of 10831176 rows
Read 28.2% of 10831176 rows
Read 29.7% of 10831176 rows
Read 30.7% of 10831176 rows
Read 32.8% of 10831176 rows
Read 33.0% of 10831176 rows
Read 35.0% of 10831176 rows
Read 37.0% of 10831176 rows
Read 39.1% of 10831176 rows
Read 41.1% of 10831176 rows
Read 43.1% of 10831176 rows
Read 44.6% of 10831176 rows
Read 46.3% of 10831176 rows
Read 48.0% of 10831176 rows
Read 49.7% of 10831176 rows
Read 51.6% of 10831176 rows
Read 53.5% of 10831176 rows
Read 55.3% of 10831176 rows
Read 57.1% of 10831176 rows
Read 58.3% of 10831176 rows
Read 60.1% of 10831176 rows
Read 62.0% of 10831176 rows
Read 62.9% of 10831176 rows
Read 64.9% of 10831176 rows
Read 66.9% of 10831176 rows
Read 68.9% of 10831176 rows
Read 70.8% of 10831176 rows
Read 72.8% of 10831176 rows
Read 74.7% of 10831176 rows
Read 75.4% of 10831176 rows
Read 77.5% of 10831176 rows
Read 79.5% of 10831176 rows
```

```
Read 81.5% of 10831176 rows
Read 83.6% of 10831176 rows
Read 85.5% of 10831176 rows
Read 87.5% of 10831176 rows
Read 89.6% of 10831176 rows
Read 91.6% of 10831176 rows
Read 93.6% of 10831176 rows
Read 94.4% of 10831176 rows
Read 96.5% of 10831176 rows
Read 98.4% of 10831176 rows
Read 99.3% of 10831176 rows
Read 10831176 rows and 3 (of 100) columns from 8.068 GB file in 00:01:37
```

```
##    user  system elapsed
##  63.311  33.452 200.088
```

```
head (mat)
```

```
##        SIREN               LIBTEFEN             DATEMAJ
## 1: 000325175            0 salari\xe9 2014-01-08T00:00:00
## 2: 005420021 10 \xe0 19 salari\xe9s 2011-07-19T00:00:00
## 3: 005420120 10 \xe0 19 salari\xe9s 1997-09-02T00:00:00
## 4: 005420120 10 \xe0 19 salari\xe9s 2014-08-28T00:00:00
## 5: 005420120 10 \xe0 19 salari\xe9s 2017-03-03T00:00:00
## 6: 005420120 10 \xe0 19 salari\xe9s 2010-12-24T00:00:00
```

Step 2 : Nice table

```
system.time(cil <- read.csv(file = "/Users/Margaux/Desktop/OpenCNIL_Organismes_ave
c_CIL_VD_20171204-2.csv",header = T, dec = ".",sep = ";"))
```

```
##    user  system elapsed
##   0.515   0.012   0.545
```

```
attach(cil)
```

We change the “code postal” by taking the first two digits.

```
departement <- substr (cil$Code_Postal, 1,2)
departement = as.factor(departement)
dep_clear <- data.frame(summary(departement))
dep_clear
```

```
##         summary.departement.
## 75                      2114
## 92                       943
## 69                       601
```

```
## 59                                   550
## 13                                   471
## 38                                   422
## 33                                   371
## 44                                   342
## 31                                   318
## 53                                   316
## 93                                   313
## 76                                   299
## 94                                   293
## 34                                   292
## 78                                   288
## 35                                   287
## 67                                   280
## 06                                   260
## 14                                   258
## 97                                   252
## 57                                   246
## 91                                   225
## 77                                   224
## 62                                   221
## 42                                   219
## 49                                   213
## 60                                   213
## 54                                   206
## 83                                   199
## 85                                   196
## 74                                   188
## 37                                   186
## 45                                   184
## 29                                   183
## 40                                   180
## 56                                   179
## 95                                   178
## 51                                   174
## 68                                   172
## 64                                   161
## 86                                   160
## 80                                   158
## 17                                   151
## 21                                   149
## 25                                   146
## 63                                   145
## 26                                   137
## 30                                   136
## 01                                   135
## 50                                   135
## 72                                   135
## 79                                   135
## 84                                   135
## 88                                   127
## 16                                   125
## 71                                   124
## 81                                   118
```

```
## 87                      117
## 22                      114
## 27                      114
## 66                      110
## 47                      109
## 02                      106
## 10                      103
## 73                      103
## 43                      102
## 28                       97
## 41                       97
## 20                       96
## 11                       93
## 89                       90
## 12                       89
## 18                       85
## 24                       84
## 32                       83
## 08                       82
## 61                       75
## 04                       74
## 65                       72
## 70                       72
## 03                       71
## 39                       69
## 55                       65
## 82                       65
## 07                       61
## 46                       60
##                          59
## 05                       54
## 15                       54
## 19                       54
## 36                       53
## 52                       50
## 58                       45
## 23                       32
## 98                       32
## 90                       24
## 09                       21
## 48                       12
## BP                        2
## (Other)                  11
```

Step 3 : SIREN - CNIL

We start by rename the variable ï..siren by SIREN for obtained the same name of the variable siren between matrice and cil.

After that we merge the SIREN dataset into the CNIL data with the merge function.

```
## [1] "2017-01-29" "2000-05-05" "2017-01-29" "2002-02-28" "2001-11-14"
## [6] "2006-01-09"
```

Step 4 : histogram

```
clean <- last[order(last$LIBTEFEN, decreasing=TRUE),]

clean$EFFECTIF <- factor(clean$LIBTEFEN, labels= c("0 salari?", "1 ou 2 salari?s",
"3 ? 5 salari?s", "6 ? 9 salari?s", "10 ? 19 salari?s", "20 ? 49 salari?s", "50 ?
99 salari?s", "100 ? 199 salari?s", "200 ? 249 salari?s", "250 ? 499 salari?s", "5
00 ? 999 salari?s", "1000 ? 1 999 salari?s", "2 000 ? 4 999 salari?s", "5 000 ? 9
999 salari?s", "10 000 salari?s et plus", "Unit?s non employeuses"))


library(ggplot2)
ggplot(clean, aes(EFFECTIF)) +
  geom_histogram(stat="count", col="blue")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```