Challenge B

Aymeric Maillard & Théo Le Zoualc'h

06/12/2017

https://github.com/aymericmrd/Challenge-B.git We didn't succeed to import files on repository

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

Step 1:

We decide to choose the algorithm random forest. Random forest relies on decision trees which are gathering in order to form a stronger machine learning method. Its operations is based on the principle that the more trees, the more robust forest is.

Step 2:

```
#We use the randomForest function replacing missing observations by median value. We use the model from
fit.rf <- randomForest(SalePrice ~ MSZoning + LotArea + Neighborhood  + YearBuilt + OverallQual, data =
summary(fit.rf)
```

```
##                 Length Class  Mode
## call               4   -none- call
## type               1   -none- character
## predicted       1460   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       1460   -none- numeric
## importance         5   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               1460   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

Step 3 :

```
#We make predictions with the data test and the two regressions
forest.pred <- data.frame(SalePrice_predict = predict(fit.rf, data = test, type="response"))
fit.lm <- lm(SalePrice ~ MSZoning + LotArea + Neighborhood  + YearBuilt + OverallQual, data = train, na
lm.pred <- data.frame(SalePrice_predict = predict(fit.lm, data = test, type="response"))
```

```r
summary(abs(lm.pred - forest.pred))
```

```
##  SalePrice_predict
##  Min.   :     3.68
##  1st Qu.:  5163.38
##  Median : 11193.28
##  Mean   : 15289.65
##  3rd Qu.: 20674.44
##  Max.   :227402.81
```

```r
summary(abs((lm.pred - forest.pred)/forest.pred))
```

```
##  SalePrice_predict
##  Min.   :0.0000179
##  1st Qu.:0.0293014
##  Median :0.0627446
##  Mean   :0.0927940
##  3rd Qu.:0.1195133
##  Max.   :1.1191774
```

There is on average a difference in absolute value of about 15365$. We can also see there is a difference of about 9.3% between these 2 predictions.

Task 2B - Overfitting in Machine Learning (continued)

Step 1 - Estimate a low-flexibility local linear model on the training data.

Train local linear model y ~ x on training, using default low flexibility (high bandwidth)

```r
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 - Estimate a high-flexibility local linear model on the training data.

Train local linear model y ~ x on training, using default low flexibility (high bandwidth)

```r
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)
```

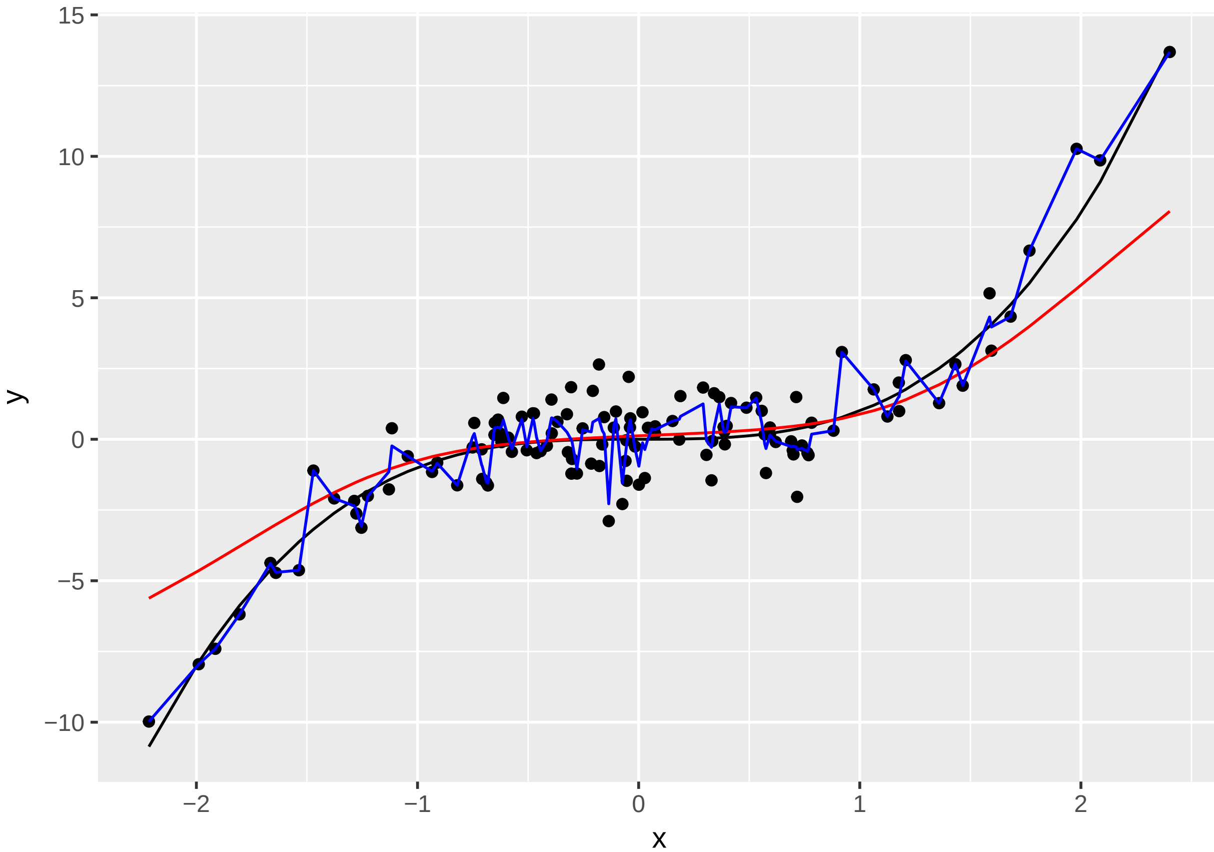
```
##                   x
## Bandwidth(s): 0.01
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
## Residual standard error: 0.5882872
## R-squared: 0.9569811
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

from Challenge A

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.

from Challenge A

```
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.lowflex), color = "red") +
  geom_line(mapping = aes(x = x, y = y.ll.highflex), color = "blue")
```



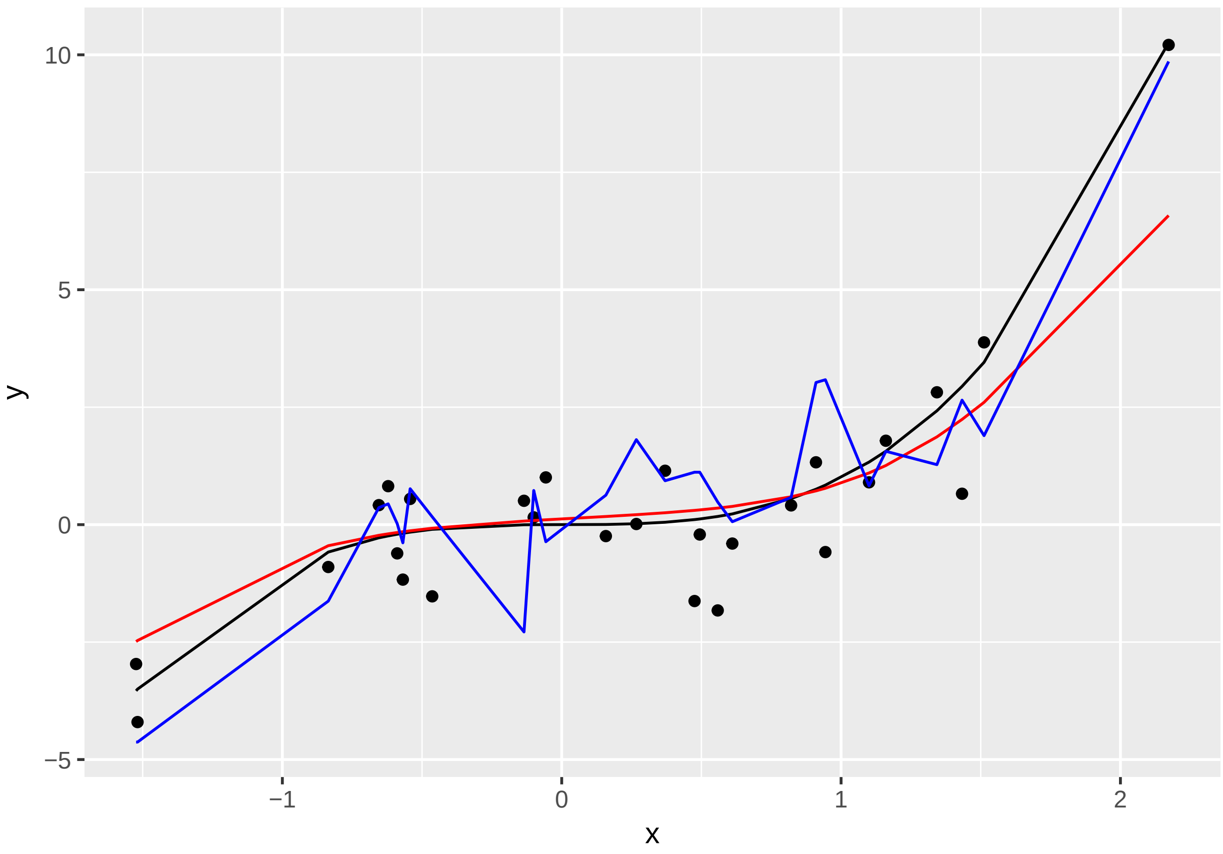
Step 4 - Between the two models, which predictions are more variable? Which predictions have the least bias?

In the highflex model, we observe that the predictions are more variable/fluctuate. Moreover we observe that it's still the highflex model which has the smallest bias.

Step 5 - Plot the scatterplot of x-y, along with the predictions of ll.fit.lowflex and ll.fit.highflex now using the test data.

from Challenge A

```
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.lowflex), color = "red") +
  geom_line(mapping = aes(x = x, y = y.ll.highflex), color = "blue")
```



One more time, predictions are more variables in the highflex model.. Moreover, we observe that's the lowflex which has the smallest bias now.

Step 6 - Create a vector of bandwidth going from 0.01 to 0.5 with a step of 0.001

from Challenge A

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

Step 7 - Estimate a local linear model y ~ x on the training data with each bandwidth.

from Challenge A

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

Step 8 - Compute for each bandwidth the MSE on the training data.

from Challenge A

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

Step 9 - Compute for each bandwidth the MSE on the test data.

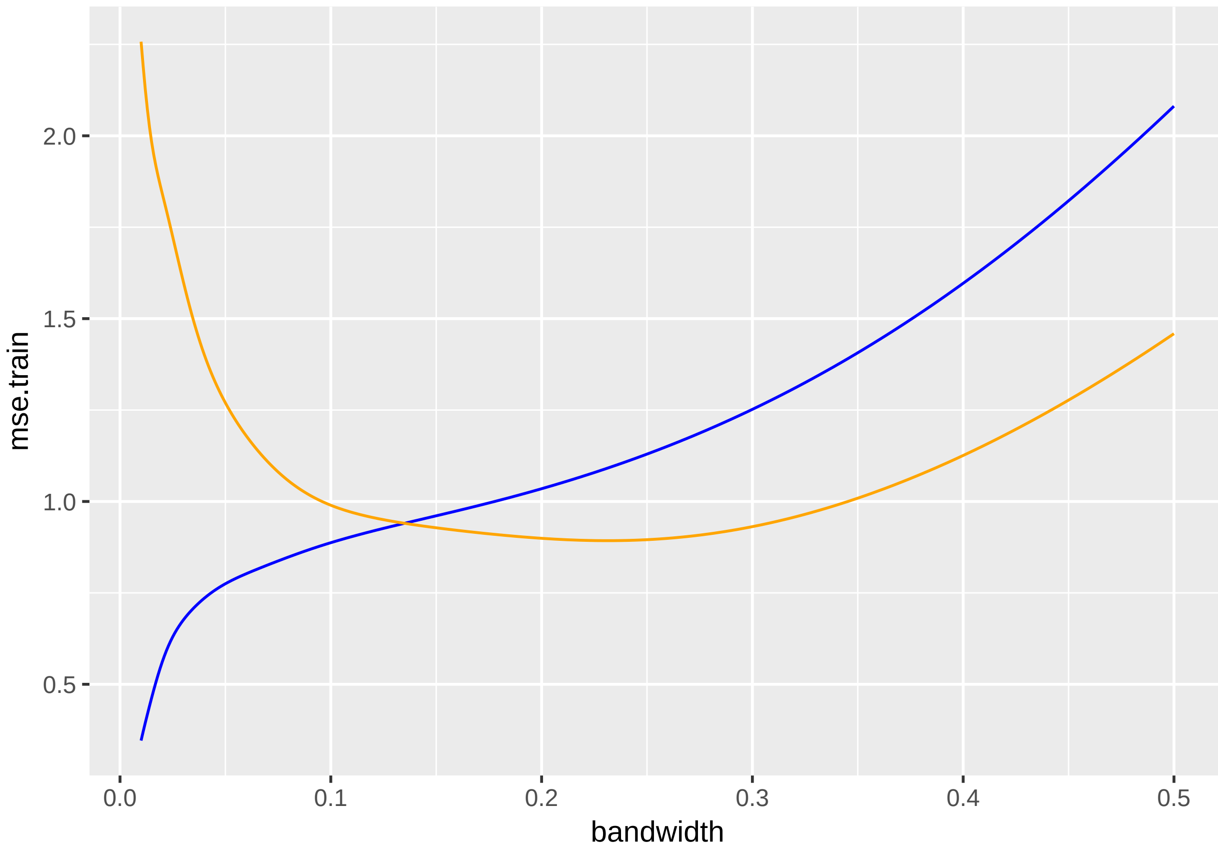
from Challenge A

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

Step 10 - Step 10 - Draw on the same plot how the MSE on training data, and test data

from Challenge A

```
mse.df <- tbl_df(data.frame(bandwidth = bw, mse.train = mse.train.results, mse.test = mse.test.results))
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

My computer is too slow to export task 3 in pdf ##Step 1 - Import the CNIL dataset from the Open Data Portal.

```
CNIL <- read.csv("/Users/aymericmrd/Documents/rprog/Challenge/OpenCNIL_Organismes_avec_CIL_VD_20171204.
```

Step 2 - Show a (nice) table with the number of organizations that has nominated a CNIL per department.

```
# We select the postal code
Dep <- data.frame(Dep = CNIL$Code_Postal)
#Then we select the department code
Dep <- str_sub(Dep, 1,2)
#We compute a table to know frequencies
Dep.companies <- as.data.frame(table(Dep))
colnames(Dep.companies) <- c("Department","Number of companies")
```

Step 3 - Merge the information from the SIREN dataset into the CNIL data.

```
#system.time(data.siren <- fread("/Users/aymericmrd/Documents/rprog/Challenge/sirc-17804_9075_14209_201
```

```
#We sort dates with decreasing order
#data.siren <- data.siren[order(DATEMAJ , decreasing = TRUE ),]

#We delete duplicated SIREN number and keep the most update
#data.siren <- subset(data.siren, FUN = !duplicated(data.siren[,1]))
#anyDuplicated(data.siren)

#We try to merge the two dataset by completing informations from the CNIL data with the "data.siren" da
#CNIL.SIREN <- merge(x=CNIL, y=data.siren, by.x = "Siren", by.y = "SIREN" , all.x=TRUE, sort = FALSE)
#attach(CNIL.SIREN)
```

Step 4 - Plot the histogram of the size of the companies that nominated a CIL.

```
#We select the data about the size of companies
#size.cil <- data.frame(CNIL.SIREN$LIBTEFEN)

#We create a table to know frequencies
#size.cil <- as.data.frame(table(size.cil))
#colnames(size.cil) <- c("Size","Number of companies")
```

We arrange the look of the data

```
#substi1 <- function(x) {gsub("[\xe9]","é",x) }
#substi2 <- function(x) {gsub("[\xe0]","à",x) }
#size.cil$Size <- substi1(size.cil$Size)
#size.cil$Size <- substi2(size.cil$Size)
```

We try to compute the histogramm

```
#barplot(size.cil$`Number of companies`, col = "green",
       # xlab="Size",
       # ylab="Number of companies",
       # main="Size of the companies that nominated a CIL",
       # ylim=c(0,20000))
```