Challenge B - R Prog

Charlotte Chemarin, Clarisse Allier

7 dÃfÆ’Ã,Â©cembre 2017

# CHALLENGE B

<https://github.com/charlottechemarin/Allier-Chemarin-ChallengeB.git> This is the link to our repository on GitHub.

# TASB 1B

# QUESTION 1:

We choose to use Random Forest. We read that it is a very efficient algorithm of machine learning to check links between the different variables. It classifies the explicative variables regarding their links to the dependent variable. It produces a set of classifications on a random fraction of the data, then he makes them vote and deduces the order and the importance of the different variables.

This is the link to the website we were inpired by. <http://mehdikhaneboubi.free.fr/random_forest_r.html#importation-des-donnees>

# QUESTION 2:

From the previous challenge, we take the datasets. We use the same code in order to clean the data (removing the missing values and changing the variables for factor ones). We choose to take Rossi’s code in order to be sure there will be no mistake. We also load all the libraries we will use.

library(tidyverse)

library(knitr)

library(randomForest)

train <- read.csv(file.choose("train.csv"),header=T)  
attach(train)  
  
remove.vars <- train %>% summarise\_all(.funs = funs(sum(is.na(.)))) %>% gather(key = "feature", value = "missing.observations") %>% filter(missing.observations > 100) %>% select(feature) %>% unlist  
  
train %>% summarise\_all(.funs = funs(sum(is.na(.)))) %>% gather(key = "feature", value = "missing.observations") %>% filter(missing.observations > 0)

## feature missing.observations  
## 1 LotFrontage 259  
## 2 Alley 1369  
## 3 MasVnrType 8  
## 4 MasVnrArea 8  
## 5 BsmtQual 37  
## 6 BsmtCond 37  
## 7 BsmtExposure 38  
## 8 BsmtFinType1 37  
## 9 BsmtFinType2 38  
## 10 Electrical 1  
## 11 FireplaceQu 690  
## 12 GarageType 81  
## 13 GarageYrBlt 81  
## 14 GarageFinish 81  
## 15 GarageQual 81  
## 16 GarageCond 81  
## 17 PoolQC 1453  
## 18 Fence 1179  
## 19 MiscFeature 1406

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, is.na(LotFrontage) == FALSE, is.na(Alley) == FALSE, is.na(MasVnrType) == FALSE, is.na(MasVnrArea) == FALSE, is.na(BsmtQual) == FALSE, is.na(BsmtCond) == FALSE, is.na(BsmtExposure) == FALSE, is.na(BsmtFinType1) == FALSE, is.na(BsmtFinType2) == FALSE, is.na(Electrical) == FALSE)  
  
#We finally remove the Id column.  
train <- train[,-1]

Then, we chose to use the Random Forest method. We will here follw a process we found on the Internet.

fit <- randomForest(train$SalePrice~., data = train)  
print(fit)

##   
## Call:  
## randomForest(formula = train$SalePrice ~ ., data = train)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 24  
##   
## Mean of squared residuals: 684670684  
## % Var explained: 59.85

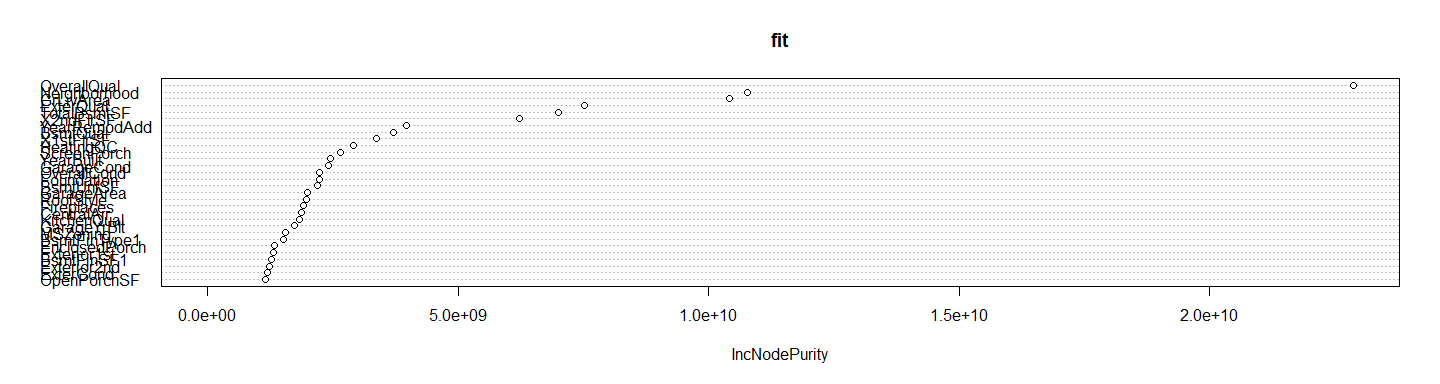
Then, we ask for the confusion matrix. In lines, we have the observations, in columns the data the model predicts.

summary(table(train$SalePrice, fit$predicted))

## Number of cases in table: 77   
## Number of factors: 2   
## Test for independence of all factors:  
## Chisq = 5236, df = 5168, p-value = 0.2506  
## Chi-squared approximation may be incorrect

We draw a graph to classify the explanatory variables according to their weight on the prediction.

varImpPlot(fit)



T

he command below give us the importance of each variable in an increasing way. We can see that the explanatory variables that matter the most are : utilities, street, heating, miscval, garagecond…

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

## OverallQual Neighborhood GrLivArea ExterQual TotalBsmtSF   
## 22879303096 10781099638 10413866445 7515706074 7003384426   
## X2ndFlrSF   
## 6231780939

# QUESTION 3:

We have to make predictions on the test data and compare them with the prediction we did for the last challenge. Once more, we use Rossi’s code to get the first prediction of our data.

First, the code for the prediction by the linear regression:

test <- read.csv(file.choose("test.csv"),header=T)  
attach(test)

test <- test %>% filter(is.na(GarageType) == FALSE, is.na(MasVnrType) == FALSE, is.na(BsmtFinType2) == FALSE, is.na(BsmtExposure) == FALSE, is.na(Electrical) == FALSE, is.na(LotFrontage) == FALSE, is.na(Alley) == FALSE, is.na(MasVnrType) == FALSE, is.na(MasVnrArea) == FALSE, is.na(BsmtQual) == FALSE, is.na(BsmtCond) == FALSE, is.na(BsmtExposure) == FALSE, is.na(BsmtFinType1) == FALSE, is.na(BsmtFinType2) == FALSE, is.na(Electrical) == FALSE)  
  
lm\_model <- lm(SalePrice ~ MSZoning + LotArea + Neighborhood + YearBuilt + OverallQual, data = train)  
summary(lm\_model)

##   
## Call:  
## lm(formula = SalePrice ~ MSZoning + LotArea + Neighborhood +   
## YearBuilt + OverallQual, data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -54123 -12726 -4308 13475 103461   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.366e+05 4.254e+05 -0.321 0.74919   
## MSZoningFV 1.772e+04 3.931e+04 0.451 0.65368   
## MSZoningRH 5.442e+04 4.916e+04 1.107 0.27248   
## MSZoningRL 2.066e+04 2.893e+04 0.714 0.47781   
## MSZoningRM 2.631e+04 2.498e+04 1.053 0.29622   
## LotArea 3.828e+00 1.422e+00 2.692 0.00908 \*\*   
## NeighborhoodCrawfor -7.101e+03 3.277e+04 -0.217 0.82914   
## NeighborhoodEdwards -4.829e+04 2.160e+04 -2.235 0.02896 \*   
## NeighborhoodIDOTRR -4.050e+04 2.753e+04 -1.471 0.14627   
## NeighborhoodNAmes -2.305e+04 3.289e+04 -0.701 0.48595   
## NeighborhoodOldTown -3.152e+04 2.285e+04 -1.380 0.17259   
## NeighborhoodSomerst NA NA NA NA   
## NeighborhoodSWISU -1.182e+04 3.410e+04 -0.347 0.73008   
## YearBuilt 7.294e+01 2.173e+02 0.336 0.73828   
## OverallQual 1.905e+04 3.851e+03 4.948 5.89e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 27970 on 63 degrees of freedom  
## Multiple R-squared: 0.6247, Adjusted R-squared: 0.5472   
## F-statistic: 8.066 on 13 and 63 DF, p-value: 3.381e-09

predict1 <- data.frame(Id = test$Id, SalePrice\_predict = predict(lm\_model, test, type="response"))

head(predict1)

## Id SalePrice\_predict  
## 1 1497 174140.6  
## 2 1498 172992.2  
## 3 1499 170423.6  
## 4 1500 151442.1  
## 5 1503 229379.3  
## 6 1533 118440.6

Then, with Random Forest:

#First, we correct the levels of the variables in order to make the two datasets match.  
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)  
  
#Then, we predict  
predict2 <- data.frame(Id = test$Id, SalePrice\_predict = predict(fit, test, type="response"))  
head(predict2)

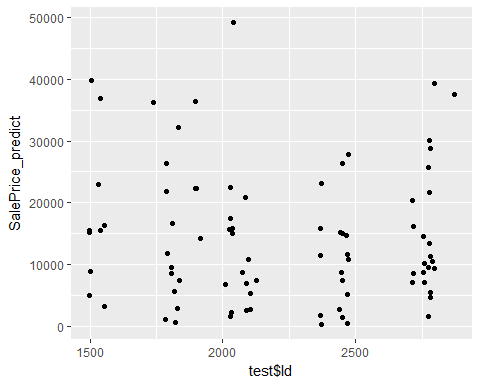
## Id SalePrice\_predict  
## 1 1497 169052.3  
## 2 1498 157502.1  
## 3 1499 155245.2  
## 4 1500 160368.2  
## 5 1503 189545.1  
## 6 1533 141489.2

To compare the two predictions, we make plots.

#First, we can compute the difference between the two predictions.  
diff <- data.frame(SalePrice\_predict = abs(predict1$SalePrice\_predict - predict2$SalePrice\_predict))  
head(diff)

## SalePrice\_predict  
## 1 5088.343  
## 2 15490.101  
## 3 15178.386  
## 4 8926.103  
## 5 39834.212  
## 6 23048.572

ggplot(diff, aes(x=test$Id, y=SalePrice\_predict)) + geom\_point()



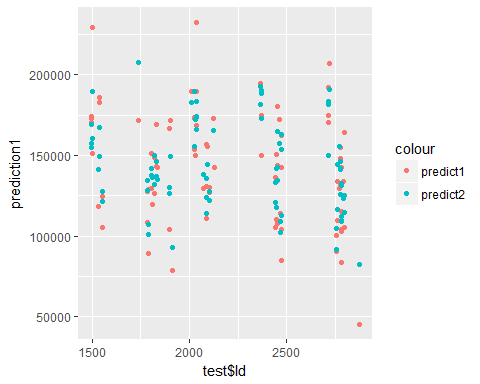
#We can see that the matching between the two predictions is unequal among the observations.

Then, we can compare the two predictions.

prediction1 <- predict1$SalePrice\_predict  
prediction2 <- predict2$SalePrice\_predict  
  
predictions <- data.frame(test$Id, prediction1 = predict1$SalePrice\_predict, prediction2 = predict2$SalePrice\_predict)  
head(predictions)

## test.Id prediction1 prediction2  
## 1 1497 174140.6 169052.3  
## 2 1498 172992.2 157502.1  
## 3 1499 170423.6 155245.2  
## 4 1500 151442.1 160368.2  
## 5 1503 229379.3 189545.1  
## 6 1533 118440.6 141489.2

ggplot(data= predictions) + geom\_point(data = predict1, aes(x= test$Id, y=prediction1, color = "predict1")) + geom\_point(data = predict2, aes(x=Id, y=prediction2, color = "predict2"))



# TASK 2B

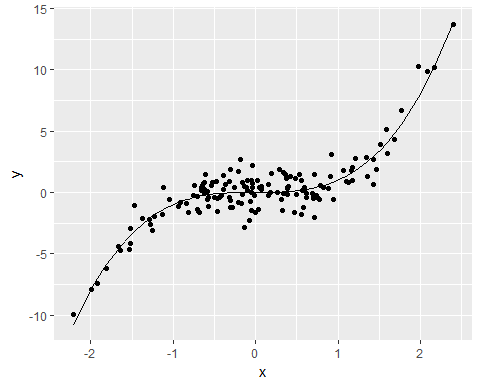
We have chosen to take the correct code from the Challenge A written by Rossi Abi-Rafeh to be sure that there will be no mistakes:

Indeed, here is the code of the Challenge A:

# We simulate an overfit:  
library(tidyverse)  
library(np)

library(caret)

# We compute the 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 simulate Nsim = 100 points of (y,x)  
ggplot(df) + geom\_point(mapping = aes(x = x, y = y)) +   
 geom\_line(mapping = aes(x = x, y = y.true))



# We split the 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, "training", "test"))  
  
training <- df %>% filter(which.data == "training")  
test <- df %>% filter(which.data == "test")  
  
# We create the model lm.fit:  
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))

Now, we can compute all the steps of the Task 2B:

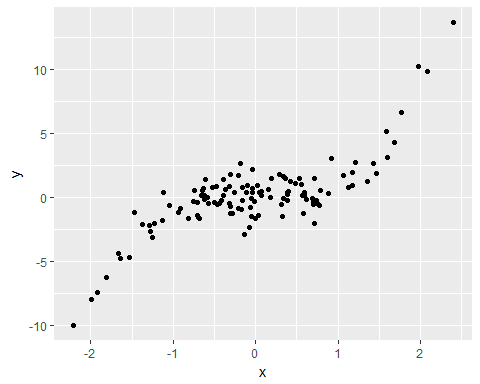
#Step 1:  
  
# We estimate a low-flexibility local linear model on the training data:  
ll.fit.lowflex <- npreg(y ~ x, data = training, method = "ll", bws = 0.5)  
summary(ll.fit.lowflex)

##   
## 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

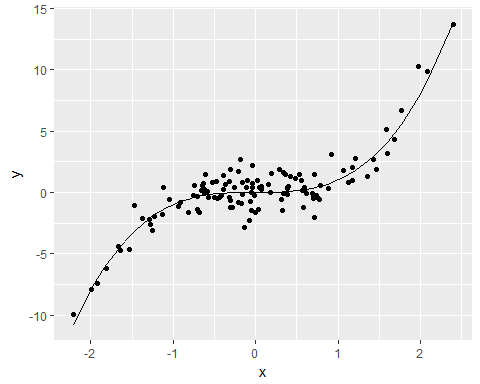
#Step 2:  
  
# We estimate a high-flexibility local linear model on the training data:  
ll.fit.highflex <- npreg(y ~ x, data = training, method = "ll", bws = 0.01)  
summary(ll.fit.highflex)

##   
## 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

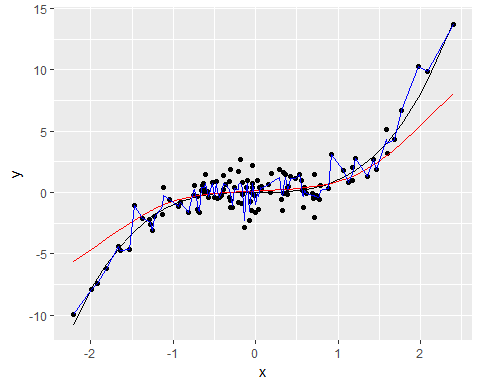
#Step 3:  
  
#We create the predictions of the two models above:  
  
training <- training %>% mutate(y.ll.lowflex = predict(object = ll.fit.lowflex, newdata = training), y.ll.highflex = predict(object = ll.fit.highflex, newdata = training))  
  
#Then, we plot the scatterplot of x-y on training data:  
  
#First, we get the simple plot of training:  
  
ggplot(training) + geom\_point(mapping = aes(x = x, y = y))



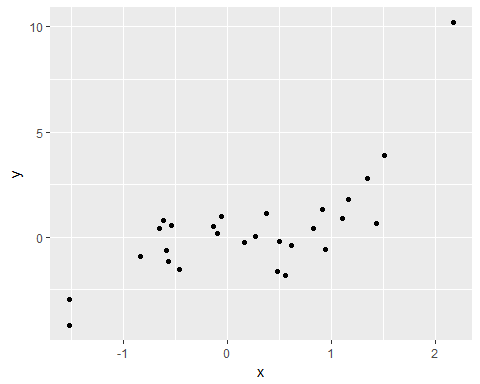
#Then, we add the true regression line:  
  
ggplot(training) + geom\_point(mapping = aes(x = x, y = y)) +   
 geom\_line(mapping = aes(x = x, y = y.true))



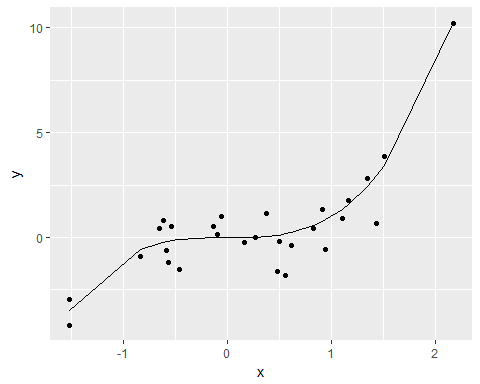
#And we can get the scatterplot of x-y with the predictions of the 2 models:  
  
ggplot(training) + geom\_point(mapping = aes(x = x, y = y)) +   
 geom\_line(mapping = aes(x = x, y = y.true)) +   
 geom\_line(mapping = aes(x = x, y = y.ll.lowflex), color = "red") +   
 geom\_line(mapping = aes(x = x, y = y.ll.highflex), color = "blue")



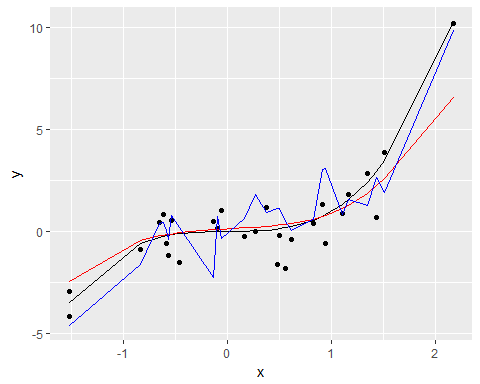
#Step 4:  
  
#The predictions of fit.highflex are more variables than the predictions of fit.lowflex, indeed, we observe more variations from the fit.highflex line than from the fit.lowflex line. The predictions of fit.lowflex have then the least bias.  
  
#Step 5:  
  
#First, we create the predictions of the two models on the test data:  
  
test <- test %>% mutate(y.ll.lowflex = predict(object = ll.fit.lowflex, newdata = test), y.ll.highflex = predict(object = ll.fit.highflex, newdata = test))  
  
#Then, we plot the scatterplot of x-y on test data:  
  
#First, we get the simple plot of test:  
  
ggplot(test) + geom\_point(mapping = aes(x = x, y = y))



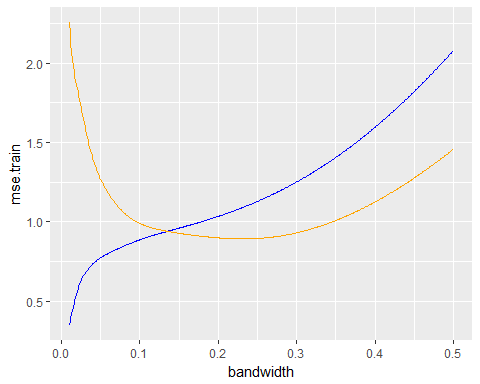
#Then, we add the true regression line:  
  
ggplot(test) + geom\_point(mapping = aes(x = x, y = y)) +   
 geom\_line(mapping = aes(x = x, y = y.true))



#And we can get the scatterplot of x-y with the predictions of the 2 models:  
  
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")



#We see that the predictions of fit.highflex are more variables than the predictions of fit.lowflex too because we observe more variations from the fit.highflex line than from the fit.lowflex line. The predictions of fit.lowflex have then the least bias.  
  
#Step 6:  
  
# We create a vector of bandwidth going from 0.01 to 0.5 with a step of 0.001:  
  
bw <- seq(0.01, 0.5, by = 0.001)  
  
#Step 7:  
  
# We estimate a local linear model on the training data with each bandwidth:  
  
llbw.fit <- lapply(X = bw, FUN = function(bw) {npreg(y ~ x, data = training, method = "ll", bws = bw)})  
  
#Step 8:  
  
#We compute for each bandwidth the MSE on the training data:  
  
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:  
  
#We compute for each bandwidth the MSE on the test data:  
  
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:  
  
#We draw on the same plot how the MSE on training data and test data changes when the bandwidth increases:  
  
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:

#Step 1:  
  
#We import the SIREN dataset:  
library(data.table)

siren <- system.time(table1 <- fread(file.choose(),sep=";",dec=".",header = T, select = c("SIREN", "LIBTEFEN")))

#Step 2:  
library(tidyverse)  
  
#We import the CNIL data set:  
library(readr)  
CNIL <- read\_delim("D:/Documents/M1 Economics/rporg/Challenges/Challenge B/OpenCNIL\_Organismes\_avec\_CIL\_VD\_20171115.csv",";", escape\_double = FALSE, trim\_ws = TRUE)

#First, we clean the table by removing all the NA present in it:  
CNIL2<-na.omit(CNIL)  
sum(is.na(CNIL2))

#Then, we create a new column called "Departement" with the first 2 numbers of the codepost:  
department <- substr(CNIL2$Code\_Postal, start =1, stop=2)  
CNIL2$Departement <- department  
  
#The new column is added in the table:  
head(CNIL2)

## # A tibble: 6 x 9  
## Siren Responsable Adresse  
## <chr> <chr> <chr>  
## 1 788349926 "\"\"\"LA RIVE BLEUE\"\"\"" 3/5 RUE BOILEAU  
## 2 421715731 01 DIRECT 58 AVENUE DE RIVESALTES  
## 3 409869708 01DB-METRAVIB 200 CHEMIN DES ORMEAUX  
## 4 444600464 1.2.3. SAS 57-59 -61 RUE HENRI BARBUSSE  
## 5 922002968 100 % ASNIERES 70 AVENUE D'ARGENTEUIL  
## 6 429621311 1000MERCIS 28 RUE DE CHATEAUDUN  
## # ... with 6 more variables: Code\_Postal <chr>, Ville <chr>, NAF <chr>,  
## # TypeCIL <chr>, Portee <chr>, Departement <chr>

#We create the table with the number of CNIL by department:  
table <- as.data.frame(table(department))  
colnames(table) <- c("Department", "Number of CNIL")  
head(table)

## Department Number of CNIL  
## 1 01 125  
## 2 02 100  
## 3 03 66  
## 4 04 70  
## 5 05 51  
## 6 06 244

#Step 3  
  
head(table1)

## SIREN LIBTEFEN  
## 1: 000325175 0 salarié  
## 2: 005420021 10 à 19 salariés  
## 3: 005420120 10 à 19 salariés  
## 4: 005420120 10 à 19 salariés  
## 5: 005420120 10 à 19 salariés  
## 6: 005420120 10 à 19 salariés

#We remove the duplicated rows:  
table1 <- table1[!duplicated(table1$SIREN),]  
  
#We need to rename the variable Siren in SIREN in the CNIL dataset otherwise we would not be able to merge the two datasets:  
colnames(CNIL2)[colnames(CNIL2) == 'Siren'] <-'SIREN'  
head(CNIL2)

## # A tibble: 6 x 9  
## SIREN Responsable Adresse  
## <chr> <chr> <chr>  
## 1 788349926 "\"\"\"LA RIVE BLEUE\"\"\"" 3/5 RUE BOILEAU  
## 2 421715731 01 DIRECT 58 AVENUE DE RIVESALTES  
## 3 409869708 01DB-METRAVIB 200 CHEMIN DES ORMEAUX  
## 4 444600464 1.2.3. SAS 57-59 -61 RUE HENRI BARBUSSE  
## 5 922002968 100 % ASNIERES 70 AVENUE D'ARGENTEUIL  
## 6 429621311 1000MERCIS 28 RUE DE CHATEAUDUN  
## # ... with 6 more variables: Code\_Postal <chr>, Ville <chr>, NAF <chr>,  
## # TypeCIL <chr>, Portee <chr>, Departement <chr>

#We can now merge the information from the SIREN dataset into the CNIL dataset using the merge function:  
library(dplyr)  
  
total <- merge(CNIL2, table1, by="SIREN")  
head(total)

## SIREN Responsable  
## 1 005520176 HERNAS CARTONNAGE  
## 2 005820378 DEMOUSELLE SAS  
## 3 006380158 ESPACE DOMICILE  
## 4 007080195 ENTREPRISES LANG  
## 5 015650617 DURUPT  
## 6 016250029 SOCIETE DES AUTOROUTES PARIS RHIN RHONE  
## Adresse Code\_Postal Ville  
## 1 50 RUE PASTEUR 80210 FEUQUIERES EN VIMEU  
## 2 140 RUE DU CHATEAU D'EAU 80100 ABBEVILLE  
## 3 13 AVENUE BARBARA 44570 TRIGNAC  
## 4 7 RUE EUGENE CORNET 44604 SAINT NAZAIRE  
## 5 4, AVENUE DE L'EUROPE 21600 LONGVIC  
## 6 36 RUE DU DOCTEUR SCHMITT 21850 SAINT APOLLINAIRE  
## NAF  
## 1 1721A Fabrication d'articles en papier ou en carton  
## 2 4321A Travaux d'installation électrique, plomberie et autres travaux d'installation  
## 3 6820A Location et exploitation de biens immobiliers propres ou loués  
## 4 4399C Autres travaux de construction spécialisés  
## 5 4674A Autres commerces de gros spécialisés  
## 6 5222Z Services auxiliaires des transports  
## TypeCIL Portee Departement LIBTEFEN  
## 1 INTERNE Etendue 80 50 à 99 salariés  
## 2 SALARIE Etendue 80 200 à 249 salariés  
## 3 EXTERNE Etendue 44 20 à 49 salariés  
## 4 MUTUALISE Générale 44 50 à 99 salariés  
## 5 SALARIE Etendue 21 20 à 49 salariés  
## 6 MUTUALISE Etendue 21 2 000 à 4 999 salariés

#Step 4:  
  
#We plot the histogram of the size of the companies that nominated a CNIL:  
  
#We first gather the variables we need in one table:  
data3 <-table1[(table1$SIREN %in% CNIL2$SIREN),]  
  
#And then we plot the histogram:  
plot(factor(data3$LIBTEFEN), las=2, cex.names= 0.5)

