Challenge B

Julia Lohse & Cristina Artero

7 December 2017

Task 1B: Fitting a random forest to predict housing prices

Step 1

Random Forest is a ML algorithm particularly effective in identifying links between explanatory variables and variables to be explained as it classifies the explanatory variables according to their links with the variable to be explained.

Step 2: Train the chosen technique on the training data

We re-use the revised training data from challenge A and we train the model using the ML algorithm Random Forest. The recommended default is of 500 trees per forest and terminal nodes which size is 5 and the defaut value of m=p/3: this is what we will be using.

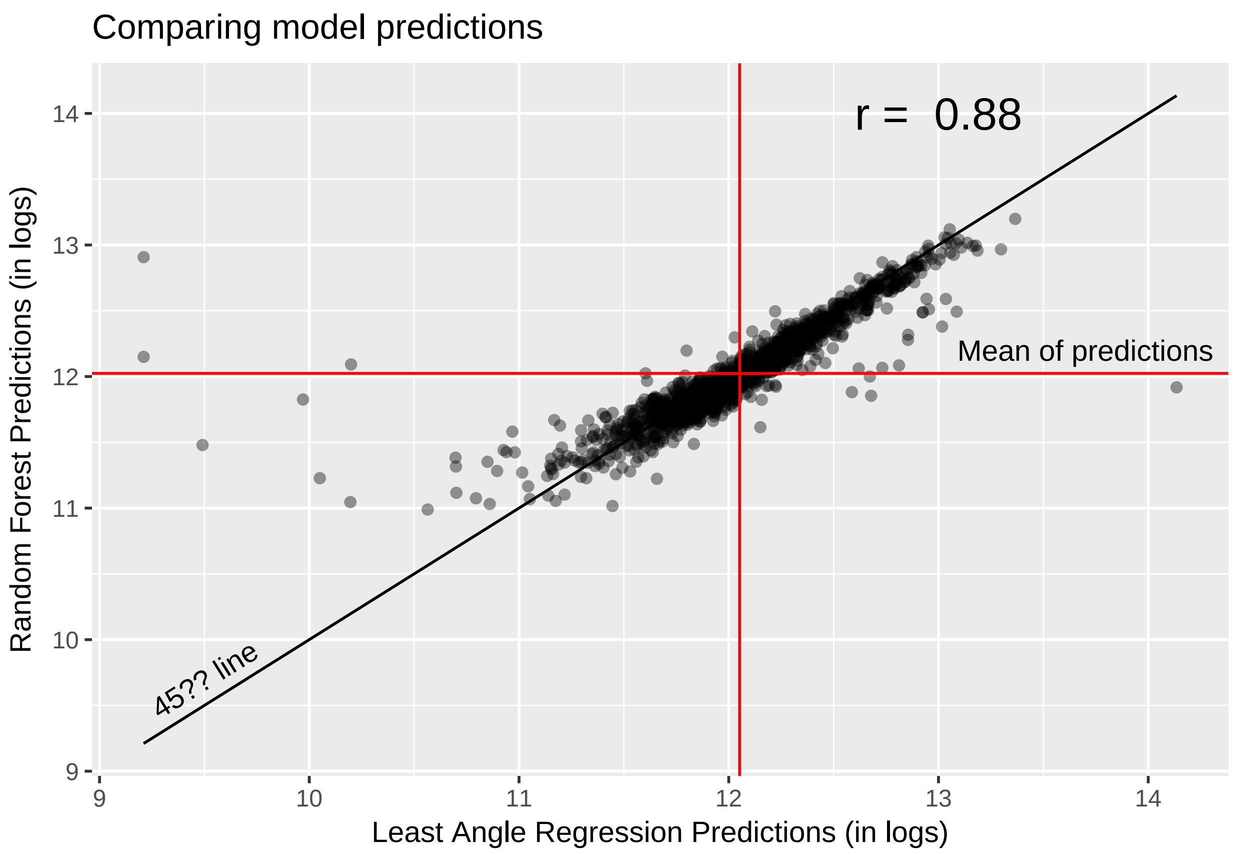
```r
training <- read.csv(file = "training_final_ChallengeA.csv", header = T)
random_forest_model1 <- randomForest(data=training, SalePrice~.-Id, ntree = 500, nodesize = 5)
```

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

We us the trained model we presented before and we compare the found predictions to the predictions found with the revised test dataset from challenge A.

```r
test <- read.csv(file = "test_final_ChallengeA.csv", header = TRUE)
rf_predictions <- predict(random_forest_model1, test)
lar_predictions <- read.csv(file = "final_predictions_ChallengeA.csv", header = TRUE)
compare_predictions <- cbind(lar_predictions,rf_predictions)
```

We plot the predictions against each other in order to compare them. According to the figure, the predictions are pretty close with r=0.88. Extreme observations in both estimations may explain why R-squared is not 1 as with these extreme observations both models predict lower or higher prices on average, as we are using the Least Angle Regression.



Task 2B - Overfitting in Machine Learning (continued) - 1 point for each step

We create the data we need for this exercise in the following manner.

```r
set.seed(1234)
nsims <- 150 # Simulations
e <- rnorm(n = nsims, mean = 0, sd = 1) # Draw 150 errors from a normal distribution
x <- rnorm(n = nsims, mean = 0, sd = 1) # Draw 150 x obs. from a normal distribution
y <- x^3+e # generate y following (T)
df <- data.frame(y,x)

df$ID <- c(1:150)
training2 <- df[df$ID %in% sample(df$ID, size = 120, replace = F), ] #training set, size 120
test2 <- df[!(df$ID %in% training2$ID), ] # remaining test dataset
df$training <- (df$ID %in% training2$ID) # Create variable specifying whether obs.
```

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

```r
ll.fit.lowflex <- npreg(training2, formula = y ~ x, method = "ll", bws = 0.5)
```

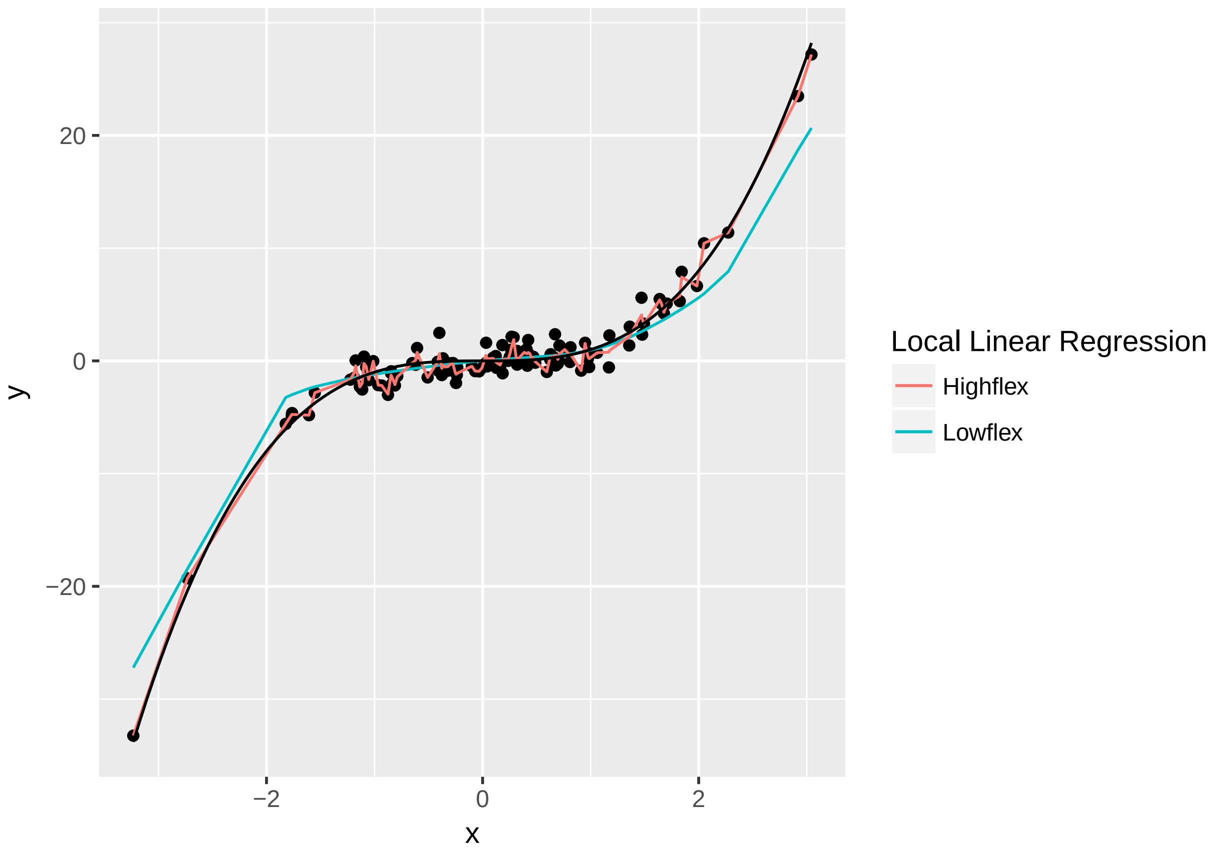
Step 2: Estimate a high-flexibility local linear model on the training data.

```r
ll.fit.highflex <- npreg(training2, formula = y ~ x, method = "ll", bws = 0.01)
```

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.

```
# We get estimates of the two models for training2 data
lowflex_estimates <- data.frame(y_estimates_lowflex = ll.fit.lowflex$mean, y = training2$y, x = ll.fit.l
highflex_estimates <- data.frame(y_estimates_highflex = ll.fit.highflex$mean, y = training2$y, x = ll.fi
combined_estimates <- merge(lowflex_estimates, highflex_estimates)

ggplot(data = combined_estimates) + geom_point(aes(x = x, y = y)) +
  geom_line(aes(x = x, y = y_estimates_lowflex, color = "red")) +
  geom_line(aes(x = x, y = y_estimates_highflex, color = "darkblue")) +
  stat_function(fun = function(x) x^3) +
  scale_color_discrete(name = "Local Linear Regression", labels = c("Highflex", "Lowflex"))
```



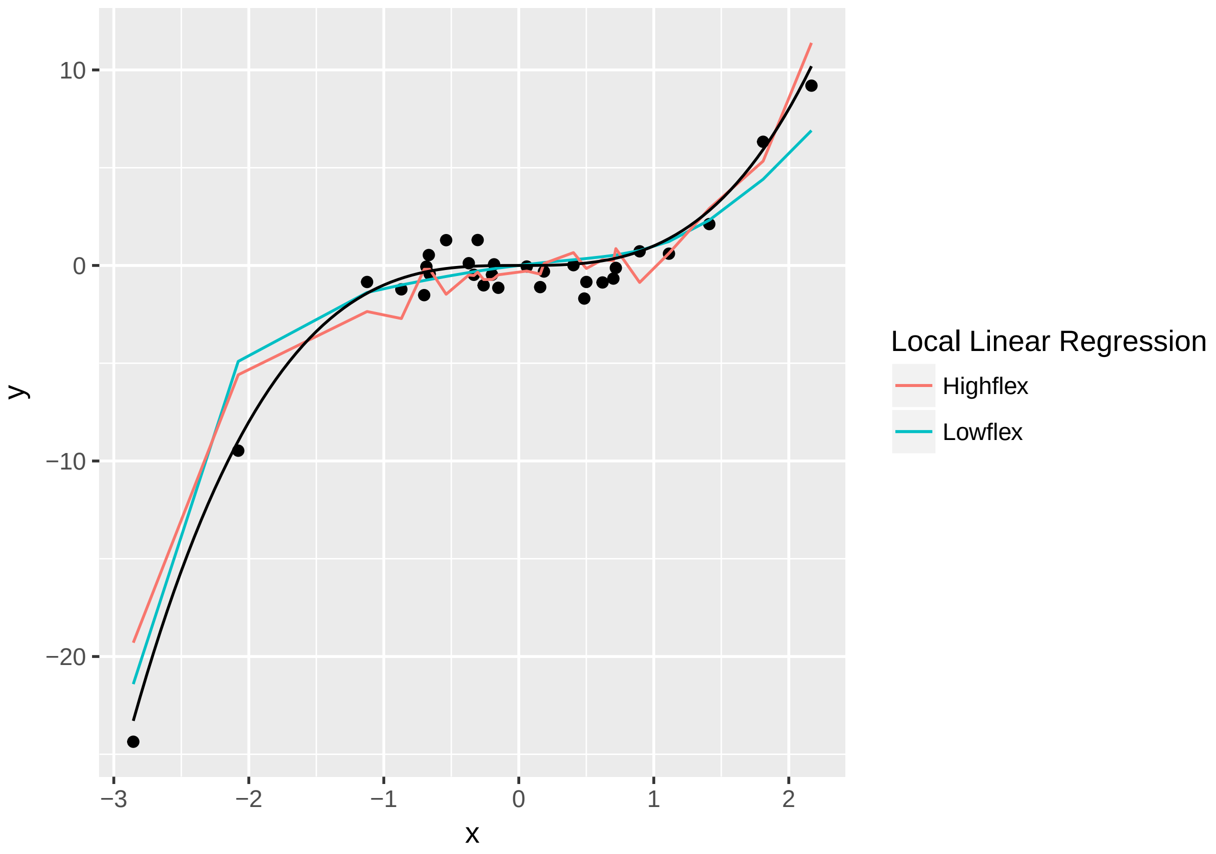
Step 4 - Between the two models, which predictions are more variable? Which predictions have the least bias?

According to the predictions on the training dataset, the high-flexibility local linear model has a better performance than the low-flexibility-model as to bias and variance is concerned. The high-flexbility model has a low bias as was expected since we only face a bias-variance trade-off when applying the model to the test data.

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. Which predictions are more variable? What happened to the bias of the least biased model?

```r
# Getting predictions of both models for test2 data
lowflex_predictions <- predict(ll.fit.lowflex, newdata = test2)
highflex_predictions <- predict(ll.fit.highflex, newdata = test2)
combined_predictions <- cbind(test2, lowflex_predictions, highflex_predictions)

ggplot(data = combined_predictions) + geom_point(aes(x = x, y = y)) +
  geom_line(aes(x = x, y = lowflex_predictions, color = "red")) + # Plotting predictions from low flexi
  geom_line(aes(x = x, y = highflex_predictions, color = "darkblue")) + # Plotting predictions from hig
  stat_function(fun = function(x) x^3) +
  scale_color_discrete(name = "Local Linear Regression", labels = c("Highflex", "Lowflex"))
```



Compare bias and variance.

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

```r
bandwidth_vector <- seq(0.01,0.5,0.001)
```

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

We have two options to do it: a loop or a vector. As said in class, doing a vector is much more efficient. Indeed, if we apply the same function to each of the vector's elements, then estimating with a vector is less time consuming.

```
run_ll <- function(bandwidth){
  npreg(training2, formula = y ~ x, method = "ll", bws = bandwidth)
}
ll_models <- lapply(X = bandwidth_vector, FUN = run_ll)
```

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

Now we extract the computed MSE from our previous model output. Vectorizing or looping take almost the same amount of time.

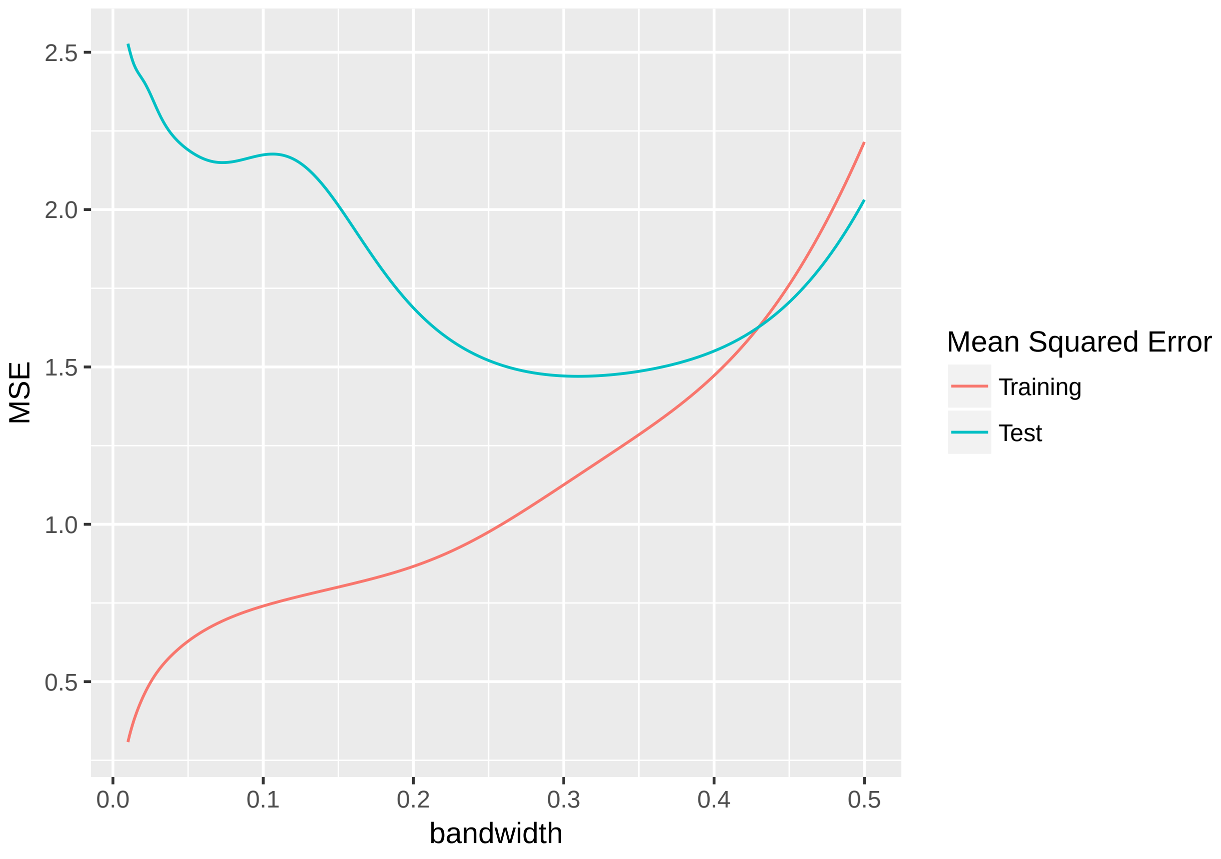
```
MSE_training <- sapply(c(1:length(bandwidth_vector)), FUN = function(i) ll_models[[i]]$MSE)
```

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

```
MSE_test <- c()
for(i in 1:length(bandwidth_vector)){
  ll_model_predictions <- predict(ll_models[[i]], newdata = test2) # Predicting for a given  bandwidth
  MSE_test[i] <- mean((test2$y-ll_model_predictions)^2) # Compute MSE for a given bandwidth
}
```

Step 10 - Draw on the same plot how the MSE on training data, and test data, change when the bandwidth increases. Conclude.

```
MSE <- data.frame(bandwidth = bandwidth_vector, MSE_training, MSE_test) # Combine in one dataset
MSE_long <- melt(data = MSE, id.vars = c("bandwidth"), value.name = "MSE") # Long format
ggplot() +
  geom_line(data = MSE_long, aes(x = bandwidth, y = MSE, group = variable, color = variable)) +
  scale_color_discrete(name = "Mean Squared Error", labels = c("Training","Test"))
```



Task 3B - Privacy regulation compliance in France

We use Sys.time to check the time that step 3B will take to run.

Step 1

First of all, we import the data set `CNIL.csv` which lists all the companies that nominated a CIL.

```
head(CNILdata)
```

```
##       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
## 5 922002968  100 % ASNIERES       70 AVENUE D'ARGENTEUIL       92600
## 6 429621311      1000MERCIS         28 RUE DE CHATEAUDUN       75009
##           Ville
## 1        ANGERS
## 2 SAINT ESTEVE
## 3     LIMONEST
## 4        CLICHY
## 5     ASNIERES
## 6         PARIS
##                                                              NAF
## 1                       8790A Autres activités d'hébergement social
## 2                          526B Commerce de détail hors magasin
## 3              7120B Activités de contrôle et analyses techniques
## 4           524C Autres commerces de détail en magasin spécialisé
## 5                        913C Autres organisations associatives
## 6 6201Z Programmation, conseil et autres activités informatiques
##         TypeCIL   Portee
## 1       INTERNE  Etendue
## 2       EXTERNE Générale
## 3 PROFESSIONNEL  Etendue
## 4       EXTERNE  Etendue
## 5       INTERNE  Etendue
## 6       INTERNE  Etendue
```

Step 2

The following table indicates how many CIL were numerated per department.

Here is the first 50 departments

Department	Number of organizations	Department	Number of organizations
01	134	51	172
02	106	52	51
03	71	53	316
04	74	54	207
05	54	55	65
06	259	56	179
07	61	57	246
08	82	58	45
09	21	59	543
10	103	60	212
11	93	61	75
12	89	62	220
13	468	63	142
14	259	64	162
15	54	65	71
16	123	66	110
17	151	67	281
18	85	68	171
19	54	69	598
20	96	70	70
21	149	71	123
22	114	72	133
23	32	73	103
24	84	74	188
25	145	75	2089
26	137	76	299
27	114	77	224
28	96	78	287
29	183	79	135
30	136	80	158
31	317	81	118
32	83	82	65
33	371	83	199
34	291	84	130
35	283	85	196
36	53	86	160
37	186	87	117
38	421	88	127
39	69	89	90
40	177	90	24
41	97	91	224
42	218	92	939
43	102	93	310
44	340	94	291
45	182	95	177
46	60	97	252
47	109	98	32
48	12	NA	73
49	213		
50	134		

Step 3

First of all we are importing the dataset `SIREN.csv`. We will only keep the first ten characters of the variable “date” which are year, month and day. We will then classify the data from the more recent to the oldest. After we will take off the duplicates: given that we sorted out the data we can delete all the duplicates that come next. Finally, we will merge the datasets by SIREN number and we will only have the informations about the companies which nominated a CIL.

```r
system.time(SIRENdata <- fread("SIREN.csv", sep = ';', header = TRUE))

# (About 11 minutes to run this step).
```

```r
# Here we have the most up to date information about each company:

SIRENdata$DATEMAJ <- str_sub(SIRENdata$DATEMAJ, 1,10)
# only the first 10 characters of the variable "date" : year, month and day

system.time(SIRENdata <- SIRENdata[ order(SIRENdata$DATEMAJ , decreasing = TRUE ),])
#more recent data to the oldest

SIRENdata <- subset(SIRENdata, !duplicated(SIRENdata[,1]))

sum(duplicated(SIRENdata))

system.time(DataCNILInfo2<-merge(x=CNILdata,y=SIRENdata,by.x ="Siren",by.y = "SIREN",all.x=TRUE, sort =
write.csv(DataCNILInfo2,file= "DataStep3.csv")

# by.x et by.y because the names of variables are differents in the two datasets
#Next we merge the datasets by SIREN number and now we have all the informations about only
#the companies which nominated a CIL.
```

Data is saved as “DataStep3.csv”

Step 4

We take the size of companies from the dataset created in Step 3 in order to plot the histogram. We use the table to obtain the frequence.

```r
DataCNILInfo2 <- fread("DataStep3.csv", sep = ",", header = TRUE)

size <- data.frame(size = DataCNILInfo2$LIBTEFET)
#First we collect the number of salaries by company, this is the size of the organization.

#We create a data frame
size <- as.data.frame(table(size))
colnames(Q3.2) <- c("Size","Number of organisations")
size$Freq <-  as.numeric(size$Freq)
```

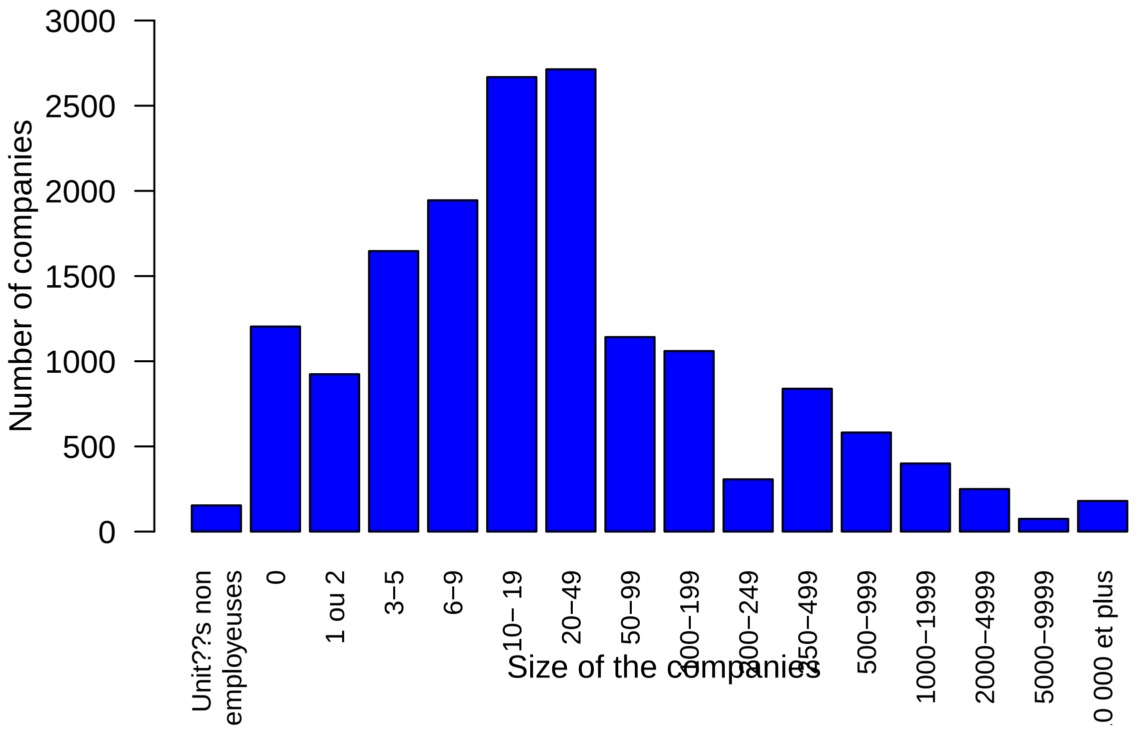


Figure 1: Size of the companies that nominated a CIL

According to the histogram, most companies that nominated a CIL have between 10 and 49 salaries. After this figure, the number of companies is decreasing in size, which reminds us to a normal law.

The time needed to run the Task 3 B is the following

```
## [1] "time to Knit the step3 on Rmd = "
```

```
## Time difference of 3.516413 secs
```