RENDU-PROJET CLASSIFICATION

Bernice AGOSSOUVO

17/06/2021

Chargement des bibliothèques utiles dans le cadre de ce projet

```
## Chargement des biblioteques
library(readr)

## Warning: package 'readr' was built under R version 4.0.5

library(ggplot2)
library(readxl)

## Warning: package 'readxl' was built under R version 4.0.5

library(prettyR)

## Warning: package 'prettyR' was built under R version 4.0.3

library(questionr)

## Warning: package 'questionr' was built under R version 4.0.5

##
## Attaching package: 'questionr'

## The following objects are masked from 'package:prettyR':
##
##     describe, freq

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

1) Chargement du jeu de données d'apprentissage

```
algue = read.table(file = "Ex4.dataTrain.txt",header = TRUE)
```

Description statistique des variables

```
summary(algue)
```

```
##     season              size              speed                mxPH      
##  Length:134         Length:134         Length:134         Min.   :7.090  
##  Class :character   Class :character   Class :character   1st Qu.:7.800  
##  Mode  :character   Mode  :character   Mode  :character   Median :8.100  
##                                                           Mean   :8.087  
##                                                           3rd Qu.:8.400  
##                                                           Max.   :9.101  
##       mnO2              Cl               NO3              NH4          
##  Min.   : 1.500   Min.   :  0.80   Min.   : 0.050   Min.   :    5.80  
##  1st Qu.: 7.900   1st Qu.: 11.94   1st Qu.: 1.324   1st Qu.:   51.19  
##  Median : 9.700   Median : 35.95   Median : 2.921   Median :  119.29  
##  Mean   : 9.005   Mean   : 47.03   Mean   : 3.392   Mean   :  567.26  
##  3rd Qu.:10.660   3rd Qu.: 58.88   3rd Qu.: 4.550   3rd Qu.:  238.25  
##  Max.   :13.400   Max.   :391.50   Max.   :45.650   Max.   :24064.00  
##       oPO4              Chla              freq       
##  Min.   :  1.333   Min.   :  0.200   Min.   :0.0000  
##  1st Qu.: 18.778   1st Qu.:  2.000   1st Qu.:0.0000  
##  Median : 48.312   Median :  5.216   Median :0.0000  
##  Mean   : 75.653   Mean   : 14.108   Mean   :0.4925  
##  3rd Qu.:102.356   3rd Qu.: 18.120   3rd Qu.:1.0000  
##  Max.   :564.600   Max.   :110.456   Max.   :1.0000
```

```
describe(algue)
```

```
## [134 obs. x 11 variables] tbl_df tbl data.frame
##
## $season:
## character: "spring" "summer" "autumn" "winter" "winter" "spring" "winter"
"spring" "winter" "spring" ...
## NAs: 0 (0%) - 4 unique values
##
## $size:
## character: "medium" "small" "small" "small" "large" "small" "small"
"small" "medium" "medium" ...
## NAs: 0 (0%) - 3 unique values
##
## $speed:
## character: "high" "high" "high" "high" "medium" "high" "medium" "medium"
"medium" "medium" ...
## NAs: 0 (0%) - 3 unique values
##
## $mxPH:
## numeric: 8.30021704253787 7.72027480969345 7.400455494822 7.89931910517858
8.49985914533632 8.30051972512947 8.00080382501101 8.34990389201557
8.50097497477103 8.00028752578609 ...
## min: 7.08971779259574 - max: 9.10098548735119 - NAs: 0 (0%) - 134 unique
values
```

```
##
## $mnO2:
## numeric: 7.7 11.8 12.5 11 10.5 12.5 9.8 8 8.6 3.3 ...
## min: 1.5 - max: 13.4 - NAs: 0 (0%) - 73 unique values
##
## $Cl:
## numeric: 10.078 6.3 13 6.167 2.75 87 60.8 57.75 125.6 26.76 ...
## min: 0.8 - max: 391.5 - NAs: 0 (0%) - 130 unique values
##
## $NO3:
## numeric: 1.212 1.47 3.33 1.172 0.758 4.87 6.238 1.288 3.778 0.658 ...
## min: 0.05 - max: 45.65 - NAs: 0 (0%) - 132 unique values
##
## $NH4:
## numeric: 103.333 8 60 18.333 10.5 22.5 578 370 124.167 165 ...
## min: 5.8 - max: 24064 - NAs: 0 (0%) - 131 unique values
##
## $oPO4:
## numeric: 48.667 16 72 7.75 4 27 105 428.75 197.83299 37.375 ...
## min: 1.333 - max: 564.59998 - NAs: 0 (0%) - 124 unique values
##
## $Chla:
## numeric: 2 0.5 4.9 0.5 4 3.3 50 1.3 40 3 ...
## min: 0.2 - max: 110.456 - NAs: 0 (0%) - 100 unique values
##
## $freq:
## integer: 0 0 1 0 0 0 1 1 1 0 ...
## min: 0 - max: 1 - NAs: 0 (0%) - 2 unique values
```

```
attach(algue)
```

Nous avons 11 variables dont 03 variables qualitatives (season, size et speed) à respectivement 04, 03 et 03 modalités et 08 variables quantitatives dont la variable freq

Transformation de la variable freq en facteur

```
algue$freq <- as.factor(algue$freq)
summary(algue)
```

```
##     season              size              speed                mxPH
##  Length:134         Length:134         Length:134         Min.   :7.090
##  Class :character   Class :character   Class :character   1st Qu.:7.800
##  Mode  :character   Mode  :character   Mode  :character   Median :8.100
##                                                           Mean   :8.087
##                                                           3rd Qu.:8.400
##                                                           Max.   :9.101
##       mnO2               Cl               NO3              NH4
##  Min.   : 1.500   Min.   :  0.80   Min.   : 0.050   Min.   :    5.80
##  1st Qu.: 7.900   1st Qu.: 11.94   1st Qu.: 1.324   1st Qu.:   51.19
##  Median : 9.700   Median : 35.95   Median : 2.921   Median :  119.29
##  Mean   : 9.005   Mean   : 47.03   Mean   : 3.392   Mean   :  567.26
```

```
##  3rd Qu.:10.660   3rd Qu.: 58.88   3rd Qu.: 4.550   3rd Qu.:  238.25
##  Max.   :13.400   Max.   :391.50   Max.   :45.650   Max.   :24064.00
##       oPO4              Chla         freq
##  Min.   :  1.333   Min.   :  0.200   0:68
##  1st Qu.: 18.778   1st Qu.:  2.000   1:66
##  Median : 48.312   Median :  5.216
##  Mean   : 75.653   Mean   : 14.108
##  3rd Qu.:102.356   3rd Qu.: 18.120
##  Max.   :564.600   Max.   :110.456
```

2) Chargement du package RandomForest

```
library(randomForest)
help("randomForest")

## starting httpd help server ... done
```

randomForest met en œuvre l'algorithme de forêt aléatoire de Breiman (basé sur le code Fortran original de Breiman et Cutler) pour la classification et la régression. Il peut également être utilisé en mode non supervisé pour évaluer les proximités entre les points de données. les codes suivants montrent l'utilisation de ce package

3) Construire l'objet fit à l'aide de la commande randomForest en spécifiant importance = T

```
fit.rf <- randomForest(freq ~ ., data = algue, importance =TRUE)
print(fit.rf) # affiche l'objet crée

##
## Call:
##  randomForest(formula = freq ~ ., data = algue, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.43%
## Confusion matrix:
##    0  1 class.error
## 0 56 12  0.17647059
## 1  6 60  0.09090909
```

Explication de la sortie

On voit d'abord un rappel de la formule utilisée

Après s'ensuit le type of random forest : Classification (puisque notre variable à expliquer est de type qualitative)

On apprend après que la forêt est composée de 500 arbres, qu'à chaque noeud l'algorithme fait un essai sur 3 variables, le taux d'erreur (out of bag error OOB qui est une mesure de l'erreur de prédiction) est donné

en effet “Chaque arbre de la forêt est construit sur une fraction (“in bag”) des données (c’est la fraction qui sert à l’entraînement de l’algorithme. Alors pour chacun des individus de la fraction restante (“out of bag”) l’arbre peut prédire une classe.” Le but du jeu est d’obtenir l’OOB le plus petit possible

et pour finir, nous avons **la matrice de confusion**

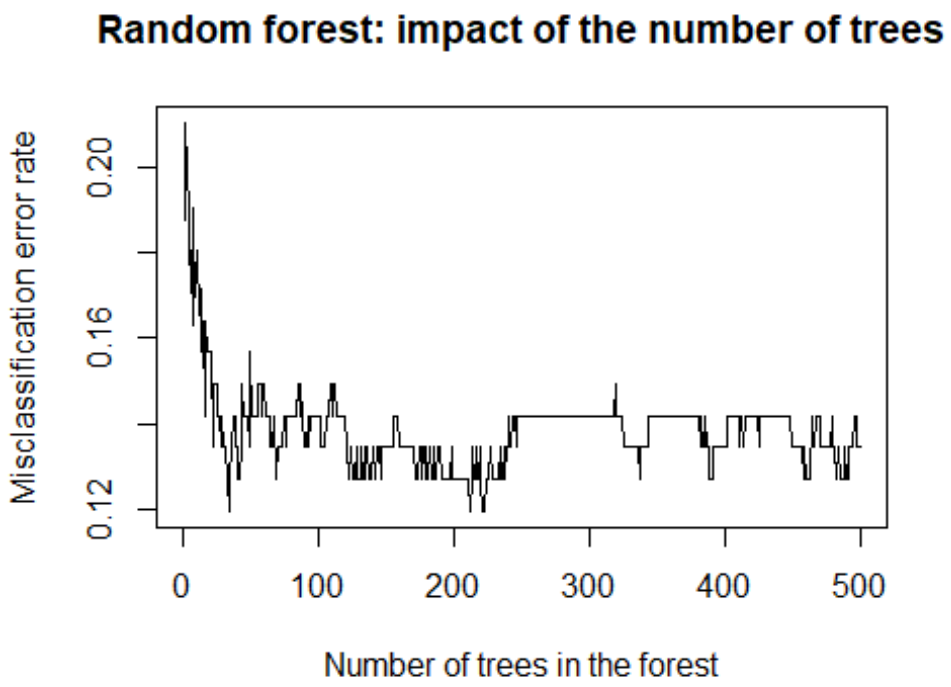
On trouve sur la diagonale (58 et 60) le nombre d’individus bien classé c’est-à-dire les individus dont la prédiction de l’algorithme correspond aux données observées. Les autres valeurs (10 et 6) correspondent aux individus mal classés par l’algorithme. Ainsi il y a 10 cours d’eau dont le seuil de prolifération ne nécéssite pas une action qui sont classés comme nécessitant une action par notre modèle et 06 dont le seuil nécéssite une action mais classés comme n’en nécéssitant pas.

De ces valeurs sont calculé les class.error ou pourcentages de mal classé que l’on calcul simplement (par exemple 10/(5+10) pour le 0.1470).

Aussi **l’err OOB** est égale à **(la somme des valeurs sur la diagonale) / (le total des enregistrements)**

4) Graphique montrant comment réduit l’OOB en fonction du nombre d’arbres générés.

```
plot(fit.rf$err.rate[,1],
     type = "l",
     main = "Random forest: impact of the number of trees",
     xlab = "Number of trees in the forest",
     ylab = "Misclassification error rate")
```

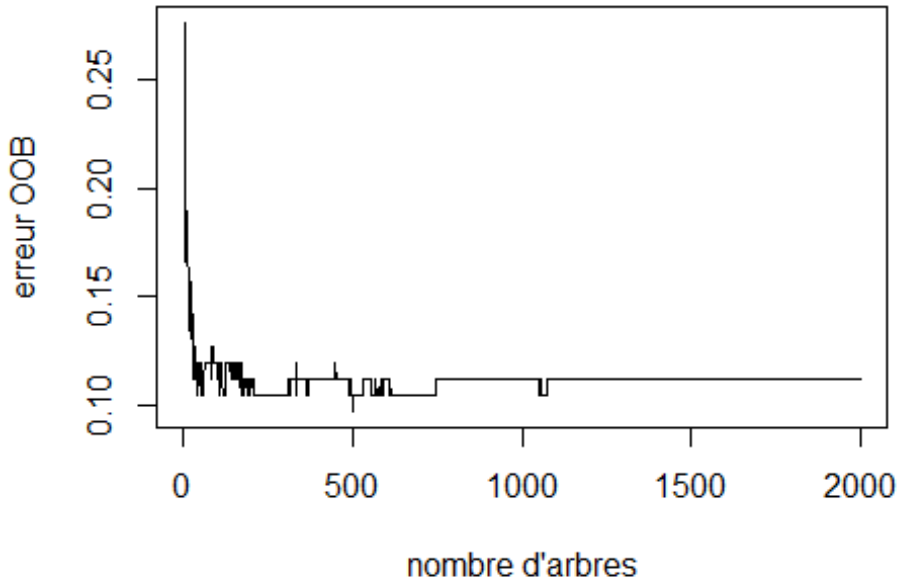


On cherche le nombre d'arbres qui stabilise l'OOB au minimum. On va voir ce que cela donnerait avec 2000 arbres

```
set.seed(123)
fit <- randomForest(freq ~., data = algue, ntree = 2000)
print(fit)

##
## Call:
##  randomForest(formula = freq ~ ., data = algue, ntree = 2000)
##                Type of random forest: classification
##                      Number of trees: 2000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 11.19%
## Confusion matrix:
##    0  1 class.error
## 0 59  9  0.13235294
## 1  6 60  0.09090909

plot(fit$err.rate[, 1], type = "l", xlab = "nombre d'arbres", ylab = "erreur
OOB")
```



5)

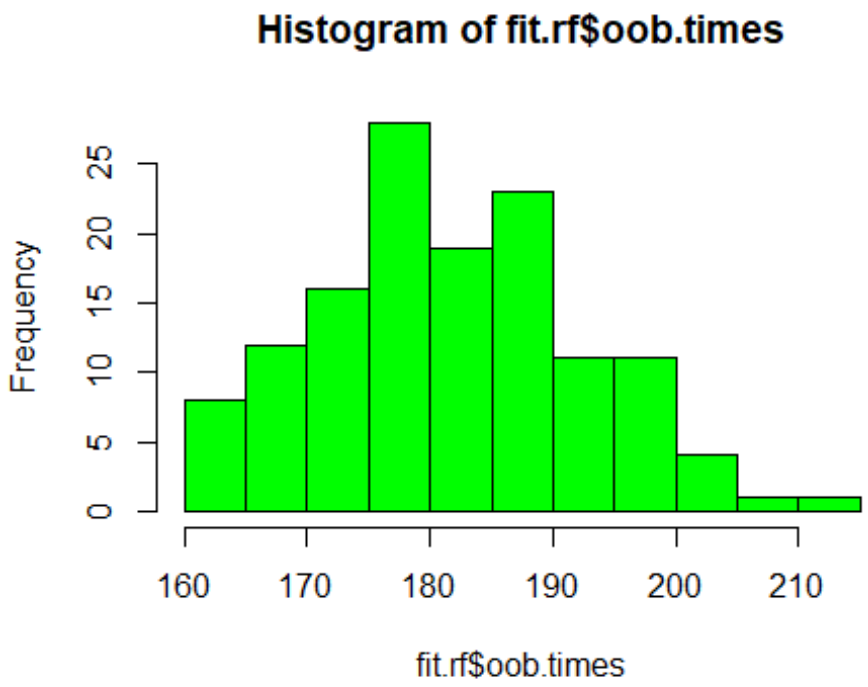
```
fit.rf$oob.times

##   [1] 182 199 183 189 176 161 173 184 182 173 191 164 166 181 174 203 186
178
##  [19] 199 190 182 169 161 191 165 177 188 188 168 191 204 200 181 191 174
176
##  [37] 197 179 176 177 177 168 194 184 174 191 166 160 198 180 166 186 173
182
##  [55] 182 180 185 179 198 177 199 175 178 193 192 178 206 165 177 177 187
186
##  [73] 178 198 186 187 173 175 176 179 189 170 170 188 183 178 180 176 180
185
##  [91] 183 186 183 168 170 186 183 172 169 205 180 201 188 177 188 183 191
188
## [109] 180 192 197 179 172 197 165 190 167 187 173 175 182 179 192 187 212
172
## [127] 187 183 172 162 190 199 173 186
```

la fonction **fit.rf$oob.times* renvoie le nombre de fois où chaque individu est “out of bag”

Histogramme des OOB

```
hist(fit.rf$oob.times,col = "green")
```



et on peut aussi s'intéresser à la proportion de votes que chaque classe a recueilli avec **fit.rf$votes**

```
fit.rf$votes[1:10,]
```

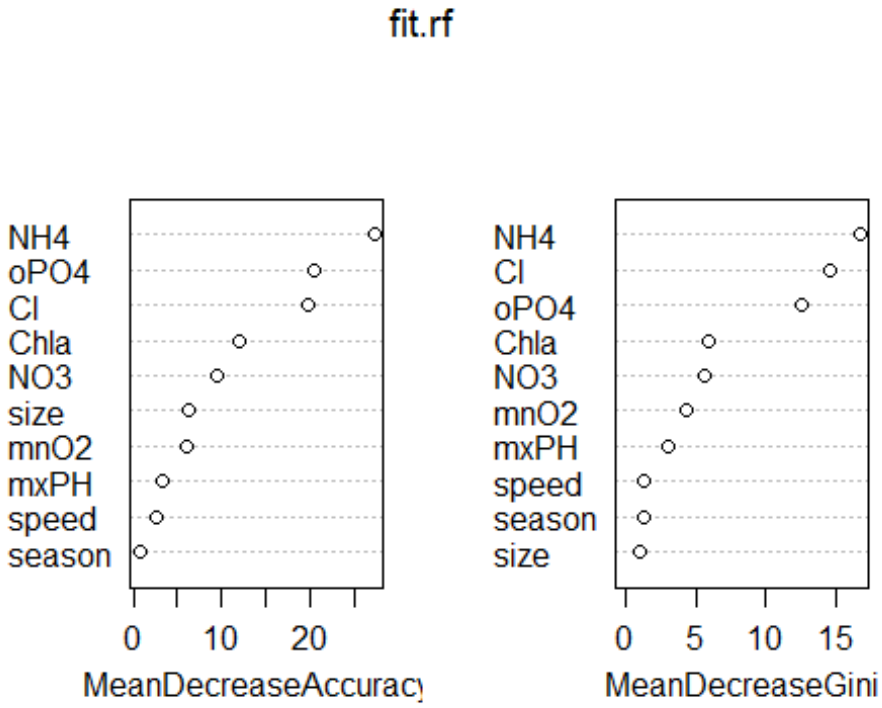
```
##              0           1
## 131 0.91208791 0.087912088
## 14  1.00000000 0.000000000
## 31  0.77595628 0.224043716
## 53  1.00000000 0.000000000
## 182 0.99431818 0.005681818
## 46  0.81366460 0.186335404
## 1   0.06358382 0.936416185
## 2   0.15760870 0.842391304
## 90  0.03846154 0.961538462
## 79  0.47398844 0.526011561
```

Ici, on voit que le premier individu a été classé en 0 dans 88% des cas où il était « OOB », et 11,93% des fois en 1.

6) Analyser l'importance des variables explicatives

Quelles sont les variables qui figurent dans notre modèle et discriminant le mieux la variable à expliquer ? On peut faire un premier graphique qui va nous présenter l'importance des variables explicatives pour distinguer les cours d'eaux dont le seuil de prolifération d'algues toxiques nécéssitant une cation est atteint.

```
varImpPlot(fit.rf)
```



On trouve les mêmes informations dans un tableau avec les commandes suivantes :

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

```
##                    0            1 MeanDecreaseAccuracy MeanDecreaseGini
## NH4      0.097367217 0.1044630947         0.0992262958        16.685788
## oPO4     0.076199256 0.0545417123         0.0648923544        12.582335
## Cl       0.031393789 0.0993771118         0.0641786580        14.559817
## Chla     0.025449982 0.0300325987         0.0270009616         5.938858
## NO3      0.019112713 0.0176700759         0.0177911690         5.617673
## mnO2     0.009894171 0.0086862892         0.0091779590         4.361455
## size     0.002577844 0.0097803710         0.0059859144         1.038031
## mxPH     0.001599500 0.0077507358         0.0042782547         3.058221
## season   0.001234404 0.0003735087         0.0007681698         1.324534
## speed   -0.000234459 0.0057904924         0.0024859716         1.338139
```

Dans le modèle calculé, les trois variables ayant une forte importance sont NH4,oPO4 et Cl

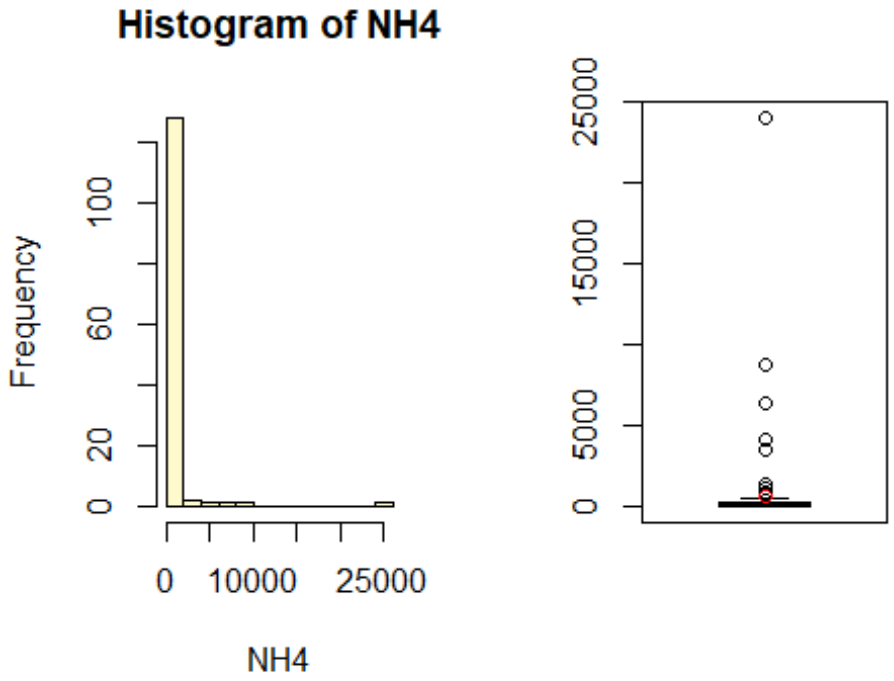
On peut visualiser graphiquement le comportement de chaque variable avec la variable freq

Variable NH4

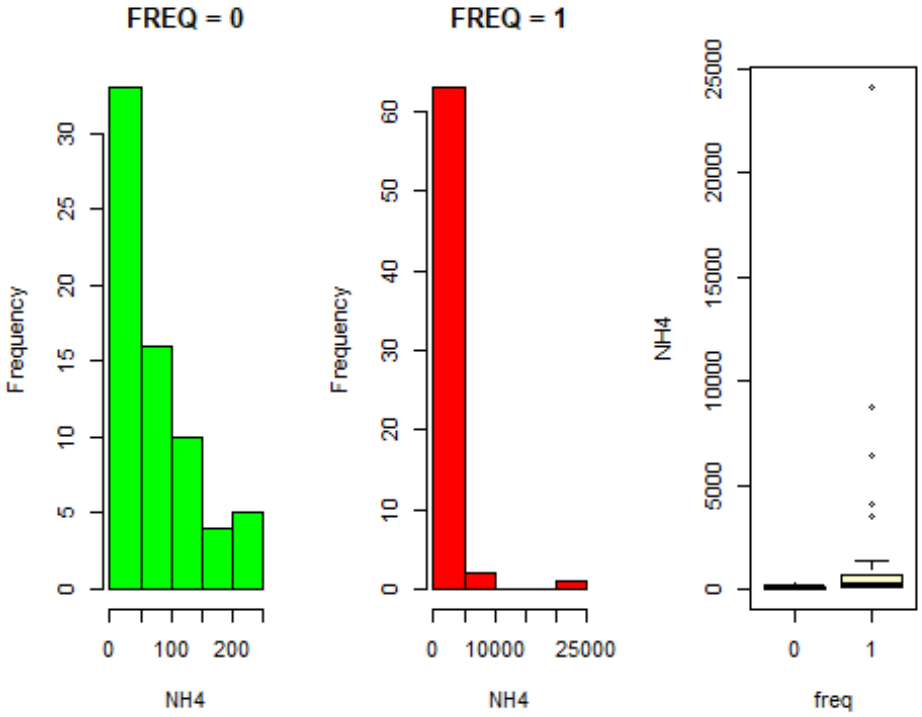
```
split.screen(1:2)
```

```
## [1] 1 2
```

```r
screen(1) ; hist(NH4, col="#FFFACD")
screen(2) ; boxplot(NH4)
points(mean(NH4),col="red")
```




```r
close.screen(all = TRUE)
par(mfrow = c(1, 3))
hist(algue$NH4[algue$freq == 0], main = "FREQ = 0", xlab ="NH4", col =
"green")
hist(algue$NH4[algue$freq == 1], main = "FREQ = 1",xlab ="NH4", col = "red")
boxplot(NH4~ freq,varwidth = FALSE, col="#FFFACD")
```



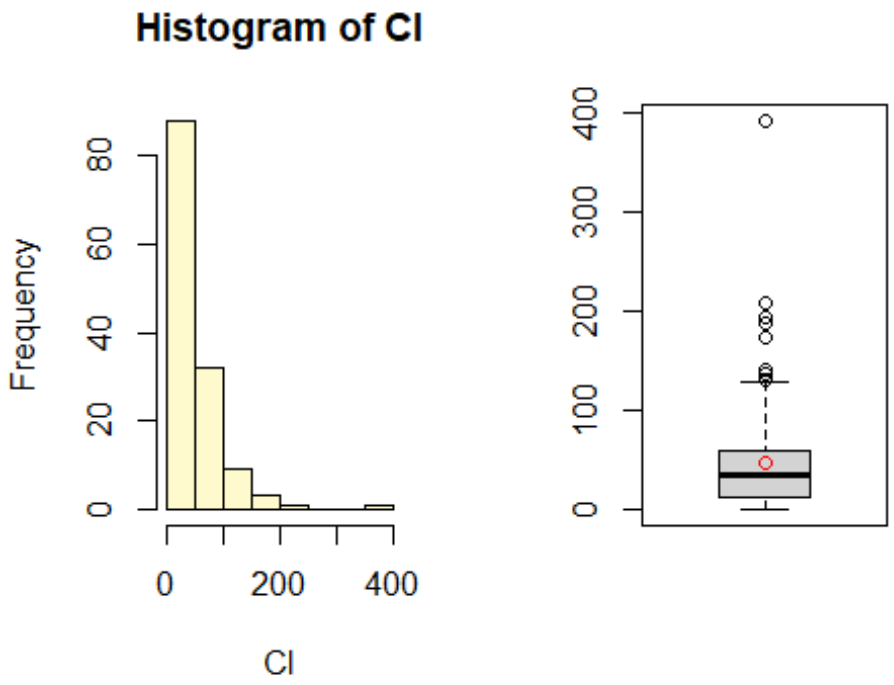
On remarque ainsi que le **NH4** est **très present** dans les cours d'eau à forte prolifération des algues toxiques avec un maximum de 25000 alors que pour les cours d'eau dont la freq est 0, le maximum est à 250. la même remarque est faite sur les variables Cl et opo4. ci dessous la representation graphique

Variable Cl

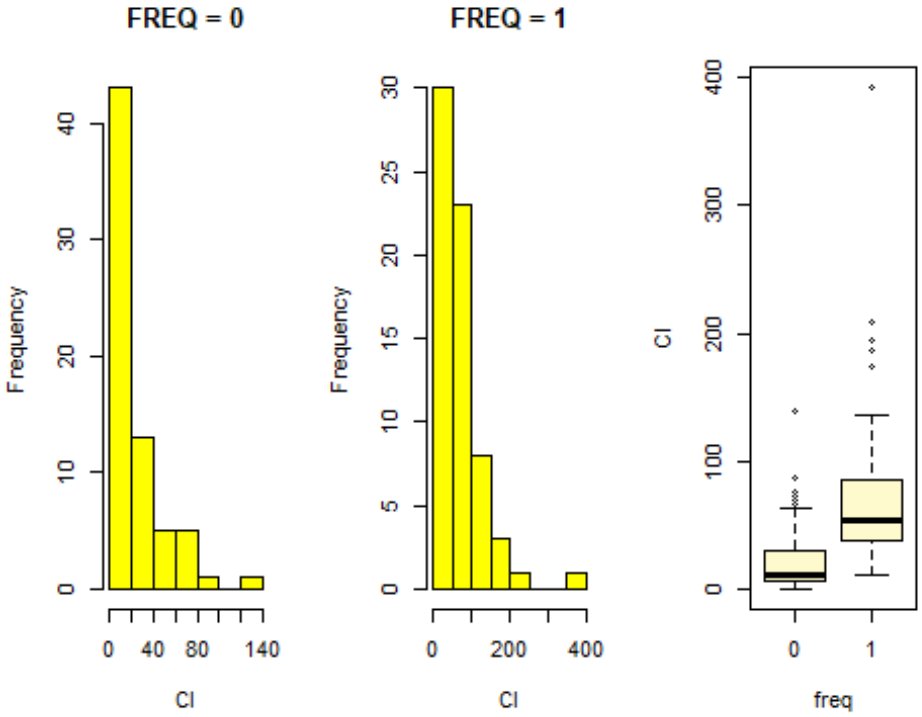
```
split.screen(1:2)

## [1] 1 2

screen(1) ; hist(Cl, col="#FFFACD")
screen(2) ; boxplot(Cl)
points(mean(Cl),col="red")
```




```
close.screen(all = TRUE)
par(mfrow = c(1, 3))
hist(algue$Cl[algue$freq == 0], main = "FREQ = 0", xlab ="Cl", col =
"yellow")
hist(algue$Cl[algue$freq == 1], main = "FREQ = 1",xlab ="Cl", col = "yellow")
boxplot(Cl~ freq,varwidth = FALSE, col="#FFFACD")
```



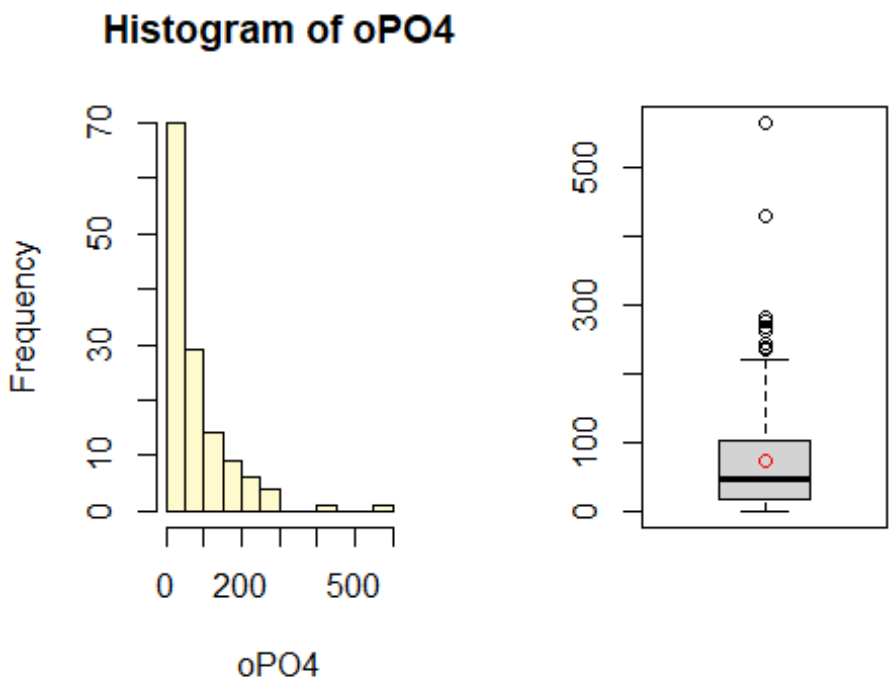
Il n'y a pas de chevauchement au niveau des boites à moustaches. la moyenne de cl dans les cours d'eau quand freq est 0 ou freq est égale à 1 sont très differentes. Cela s'explique par le fait qu'il est aussi importante dans la modélisation

Variable oPO4

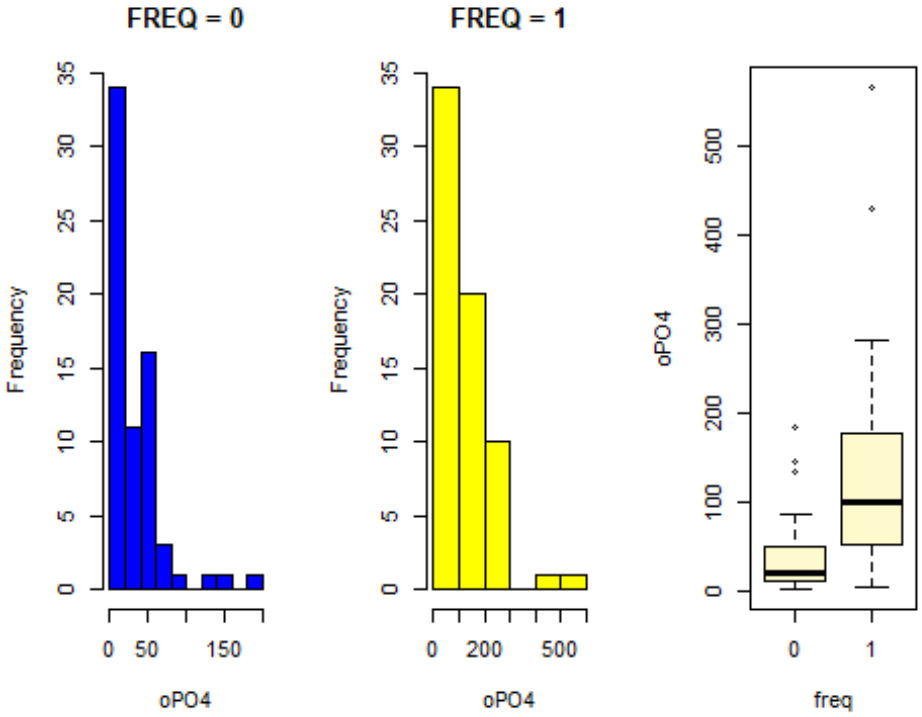
```
split.screen(1:2)

## [1] 1 2

screen(1) ; hist(oPO4, col="#FFFACD")
screen(2) ; boxplot(oPO4)
points(mean(oPO4),col="red")
```




```
close.screen(all = TRUE)
par(mfrow = c(1, 3))
hist(algue$oPO4[algue$freq == 0], main = "FREQ = 0", xlab ="oPO4", col =
"blue")
hist(algue$oPO4[algue$freq == 1], main = "FREQ = 1",xlab ="oPO4", col =
"yellow")
boxplot(oPO4~ freq,varwidth = FALSE, col="#FFFACD")
```



Même remarque que précedemment

7) Calculez le taux d'erreur d'apprentissage

8) Prédire pour le jeu de donnée test

```
test = read.table(file = "Ex4.dataTest.txt",header = TRUE)

test$predicted <- predict(fit.rf, test)
table(test$predicted)

##
##  0  1
## 26 24
```

L'agorithme prédit que 26 Cours d'eau nécessitent de traitement et que 26 n'en nécessitent pas

9)Changer la valeur de mtry pour construire un modèle Bagging et faites une prédiction pour le jeu de donnée test.

```
library(ipred)

## Warning: package 'ipred' was built under R version 4.0.5

bag <- bagging(freq ~., data=algue, coob=TRUE)
print(bag)
```

```
##
## Bagging classification trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = freq ~ ., data = algue, coob = TRUE)
##
## Out-of-bag estimate of misclassification error:  0.1716
```

```
test$predicted2 <- predict(bag, test)
table(test$predicted2)
```

```
##
##  0  1
## 28 22
```

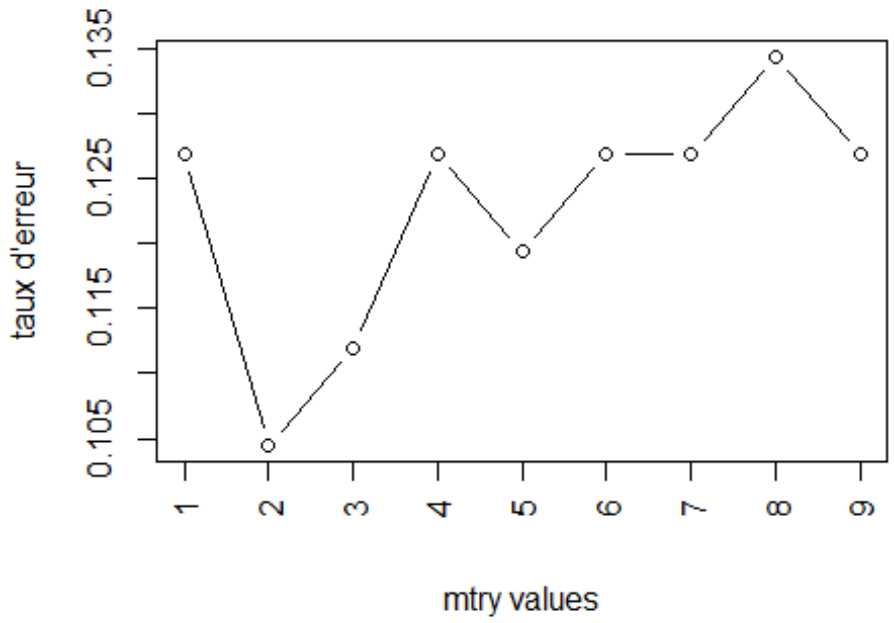
10) Proposer une procédure afin de sélectionner le meilleur mtry au sens de la minimisation du taux d'erreur OOB.

```
v.err <-  NULL
v.mtry <- c(1, 2,3, 4, 5, 6, 7, 8,9)
for(i in v.mtry){
    set.seed(123)
    fit.rf <- randomForest(freq~., data = algue, ntree= 2000, mtry=i)
    v.err <- c(v.err, fit.rf$err.rate[2000,1])
}
```

```
v.err
```

```
##       OOB       OOB       OOB       OOB       OOB       OOB       OOB
OOB
## 0.1268657 0.1044776 0.1119403 0.1268657 0.1194030 0.1268657 0.1268657
0.1343284
##       OOB
## 0.1268657
```

```
plot(v.mtry, v.err, type ="b", xlab="mtry values", ylab="taux d'erreur",
xaxt="n")
axis(1, at=v.mtry, labels=v.mtry, las=2)
```




```
fit.rf <- randomForest(freq ~ ., data = algue,mtry = 4, ntree =2000,
importance =TRUE)
print(fit.rf)

##
## Call:
##  randomForest(formula = freq ~ ., data = algue, mtry = 4, ntree = 2000,
importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 2000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 12.69%
## Confusion matrix:
##    0  1 class.error
## 0 57 11  0.16176471
## 1  6 60  0.09090909
```