Agrégation de modèles - Partie 1

Mathieu Pigeon

UQAM

Avantages des arbres de décision

- Simplicité d'interprétation $\rightarrow$ représentation graphique.
- Le fonctionnement est simple à comprendre et semble $\ll$ naturel $\gg$.
- Permet d'inclure des variables explicatives catégorielles sans créer un grand nombre de variables indicatrices.

Inconvénients des arbres de décision

- Si la relation que l'on tente de modéliser est linéaire (ou presque), alors un modèle de régression linéaire classique sera nettement meilleur ($\rightarrow$ biais et variance faibles) qu'un arbre de décision.
- Le pouvoir prédictif est généralement plus faible que pour d'autres catégories de modèles.
- Les arbres de décision sont généralement peu robustes : un petit changement dans la base de données peut avoir un énorme impact sur l'arbre final obtenu.

Exemple 1 : Préparation des données

```
library(CASdatasets)

data(freaggnumber)
dataAGG <- freaggnumber
head(dataAGG)
```

DriverAge	LicenceAge	VehAge	Exposure	ClaimNumber
39	18	3	1356.402	192
35	18	3	1243.948	172
37	18	1	1263.064	172
38	18	3	1328.589	171
39	18	2	1346.795	170
40	18	3	1263.537	165

```
dataAGG$Y <- dataAGG$ClaimNumber/dataAGG$Exposure
```

Exemple 1A : Trois arbres

```
arbre1 <- rpart(Y ~ DriverAge + LicenceAge, data = dataAGG)
arbre2 <- rpart(Y ~ DriverAge + VehAge, data = dataAGG)
arbre3 <- rpart(Y ~ VehAge + LicenceAge, data = dataAGG)
arbre4 <- rpart(Y ~ VehAge + LicenceAge + DriverAge,
                data = dataAGG)

dataAGG$pred1 <- predict(arbre1)*dataAGG$Exposure
dataAGG$pred2 <- predict(arbre2)*dataAGG$Exposure
dataAGG$pred3 <- predict(arbre3)*dataAGG$Exposure
dataAGG$pred4 <- predict(arbre4)*dataAGG$Exposure
```

Exemple 1A : Trois arbres

```
round(head(dataAGG[, c(5, 7, 8, 9, 10)]), 0)
```

	ClaimNumber	pred1	pred2	pred3	pred4
1	192	238	212	162	141
2	172	219	195	149	130
3	172	222	198	151	132
4	171	234	208	159	138
5	170	237	211	161	140
6	165	140	198	151	132

Exemple 1B : Division de la base

```
n <- length(dataAGG$Y)
indice <- sample(1:n, round(0.5*n,0), replace = FALSE)
dataA <- dataAGG[indice,]
dataB <- dataAGG[-indice,]

arbreA <- rpart(Y ~ VehAge + LicenceAge + DriverAge,
                data = dataA)
arbreB <- rpart(Y ~ VehAge + LicenceAge + DriverAge,
                data = dataB)
dataAGG$predA <- predict(arbreA, newdata = dataAGG)
                 *dataAGG$Exposure
dataAGG$predB <- predict(arbreB, newdata = dataAGG)
                 *dataAGG$Exposure
```

Exemple 1B : Division de la base

```
round(head(dataAGG[c(5, 11, 12)]), 0)
```

	ClaimNumber	predA	predB
1	192	145	160
2	172	199	147
3	172	202	149
4	171	212	157
5	170	144	159
6	165	135	149

Principe

- L'approche présentée au dernier cours pour construire des arbres de décision conduit à des prédictions ayant une **grande variance**.
- Le *bagging*, ou *Bootstrap AGGregatING* est un ensemble de (méta-)algorithmes conçus pour améliorer la stabilité et la précision d'un modèle (ou d'un algorithme) utilisé. Il permet également de réduire la variance et d'éviter le surajustement.
- Particulièrement utile pour les arbres de décision.

Algorithme Bagging

1. Générer B bases de données d'entrainement différentes.
2. À partir de chacune des bases de données d'entrainement, construire un modèle $\widehat{f}_b$, $b = 1, \ldots, B$.
3. Pour une nouvelle observation $\mathbf{X}$, la prédiction agrégée (*bagging*) sera

$$\widehat{f}_{ag}(\mathbf{X}) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}_b(\mathbf{X}).$$

Étape 1

(1) Générer les B bases de données.

- En pratique, on n'a pas accès à B bases de données différentes.
- Si on divise la base de données initiale en B bases indépendantes (comme pour de la validation croisée), on perd de l'information et les bases ainsi créées sont trop petites (il faut que B soit grand pour que le *Bagging* soit intéressant).
- On va plutôt utiliser du *bootstrap* pour générer ces B bases de données.

Étape 1

- La base de données b, $b = 1, \ldots, B$, de taille n est obtenue en pigeant au hasard **avec remise** n observations de la base de données initiale.
- Les B bases de données sont créées à partir de la même base initiale $\rightarrow$ les prédictions obtenues $\widehat{f}_b$ **ne sont pas** indépendantes.

Étape 2

(2) Construction du modèle b, $b = 1, \ldots, B$.

- En pratique, quand la technique du *bagging* est utilisée pour des arbres de décision, ces derniers **ne sont pas** élagués.
- Chacun des arbres b, $b = 1, \ldots, B$, a donc un faible biais mais une grande variance.
- La réduction de la variance se fait à l'étape 3 en agrégeant les différents arbres obtenus.

Étape 3

(3) Agrégation des modèles

$$\begin{aligned}
\text{Var}\left[\widehat{f}_{ag}(\mathbf{X})\right] &= \text{Var}\left[\frac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(\mathbf{X})\right] \\
&= \frac{1}{B^2}\text{Var}\left[\sum_{b=1}^{B}\widehat{f}_b(\mathbf{X})\right] \\
&= \frac{1}{B^2}\sum_{b_1=1}^{B}\sum_{b_2=1}^{B}\text{Cov}\left[\widehat{f}_{b_1}(\mathbf{X}),\widehat{f}_{b_2}(\mathbf{X})\right] \\
&< \frac{1}{B^2}B^2\text{Var}\left[\widehat{f}_b(\mathbf{X})\right] = \text{Var}\left[\widehat{f}_b(\mathbf{X})\right]
\end{aligned}$$

puisque lorsque $b_1 \neq b_2$, on a $\text{Corr}\left[\widehat{f}_{b_1}(\mathbf{X}),\widehat{f}_{b_2}(\mathbf{X})\right] < 1$.

Algorithme Bagging

- Empiriquement, on observe qu'agréger des centaines, voire des milliers d'arbres ($B = 100, 1\,000, 10\,000, \ldots$) augmente beaucoup le pouvoir prédictif des arbres de décision.
- Par contre, les résultats sont plus difficiles à interpréter : pas de graphique simple, etc.
- Les logiciels qui offrent ce type de méta-algorithme proposent généralement des méthodes permettant de mesurer l'importance des variables explicatives dans le modèle.

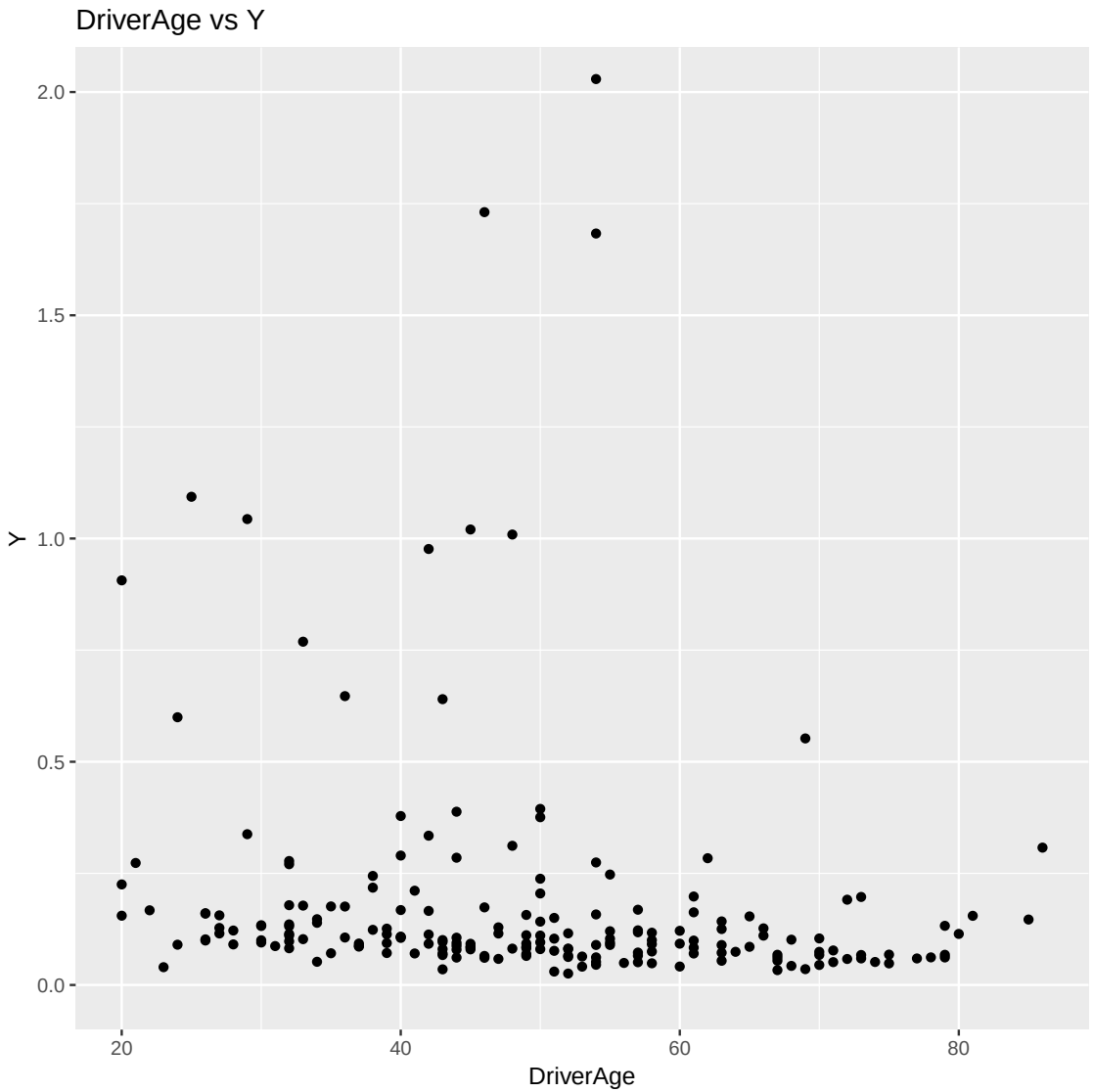
Exemple 2 : Préparation des données

```
library(data.table)
library(rpart)
library(ggplot2)

indice <- sample(1:length(dataAGG$Y), 200, replace = FALSE)
EX2 <- dataAGG[indice, c(1, 6)]
head(EX2)
```

```
      DriverAge          Y
1027         43  0.10064863
503          52  0.11572296
8590         61  0.19823609
12485        54  5.53409091
8966         68  0.10157119
3376         35  0.07114336
```

Exemple 2 : Préparation des données



Exemple 2 : Modèle sans Bagging

```
data_test <- data.table(EX2)

### Ajustement du modèle sans Bagging
modele0 <- rpart(Y ~ DriverAge, data_test, control =
                   rpart.control(minsplit = 6))

### Prédictions pour le modèle sans Bagging
pred0 <- predict(modele0, EX2)
```

Exemple 2 : Modèle avec Bagging

```
n_model <- 100

bagged_models <- list()
for (i in 1:n_model)
{
new_sample <- sample(1:length(data_test$DriverAge),
                     size = length(data_test$Y),
                     replace = TRUE)
  bagged_models <- c(bagged_models, list(rpart(Y~DriverAge,
                     data = data_test[new_sample,],
                     control = rpart.control(minsplit = 6))))
}
```

Exemple 2 : Modèle avec Bagging

```
bagged_result <- NULL

i <- 0

for (j in bagged_models)
{
  if (is.null(bagged_result))
    bagged_result <- predict(j, EX2)
  else
    bagged_result <- (i*bagged_result + predict(j, EX2))
                     /(i + 1)
  i <- i + 1
}
```

Exemple 2 : Graphique (avec ggplot)

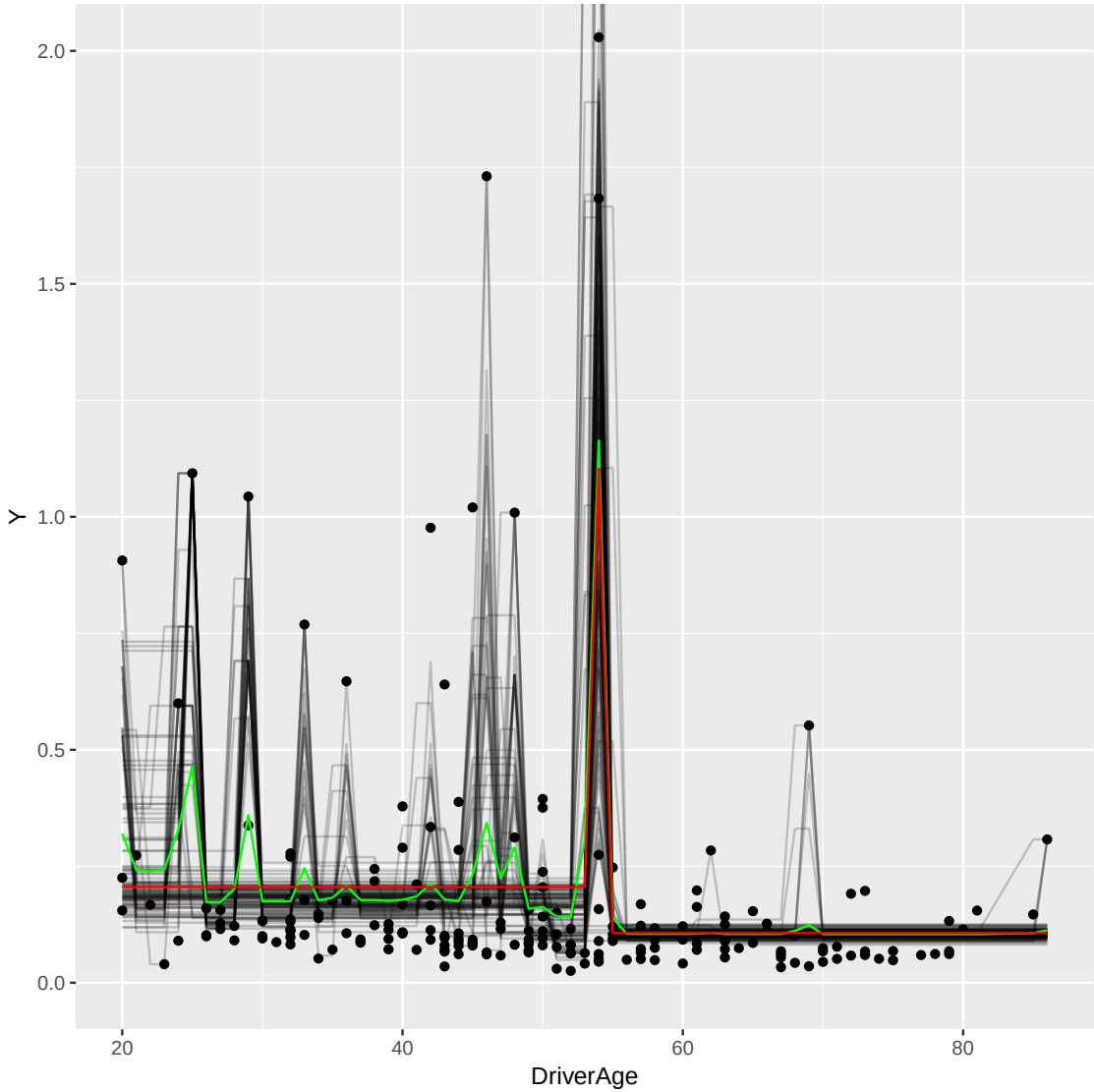
```
gg <- ggplot(data_test, aes(DriverAge,Y), ylim = c(0,2)) +
              geom_point(aes()) +
              coord_cartesian(ylim = c(0, 2))
for (k in bagged_models[1:n_model])
{
  prediction <- predict(k, EX2)
  data_plot <- data.table(DriverAge=EX2$DriverAge,
                          Y = prediction)
  gg = gg + geom_line(data = data_plot[order(DriverAge)],
       aes(x = DriverAge, y = Y), alpha = 0.2)
}
```

Exemple 2 : Graphique (avec ggplot)

```
data_bagged <- data.table(DriverAge = EX2$DriverAge,
                          Y = bagged_result)
gg <- gg + geom_line(data  =data_bagged[order(DriverAge)],
      aes(x = DriverAge, y = Y), color = 'green')

data_no_bag <- data.table(DriverAge = EX2$DriverAge,
                          Y = pred0)
gg <- gg + geom_line(data = data_no_bag[order(DriverAge)],
      aes(x = DriverAge, y = Y), color = 'red')
gg
```

Exemple 2 : Résultats



Exemple 1C : Bagging

```
install.packages("ipred")
library(ipred)

arbre4bagging <- bagging(Y ~ VehAge + LicenceAge + DriverAge,
                         data = dataAGG, nbagg = 1000)
dataAGG$predBAG <- predict(arbre4bagging)*dataAGG$Exposure
```

Exemple 1C : Bagging

```
round(head(dataAGG[, c(5, 7:13)]), 0)
```

	ClaimNumber	pred1	pred2	pred3	pred4	predA	predB	predBAG
1	192	238	212	162	141	145	160	155
2	172	219	195	149	130	199	147	153
3	172	222	198	151	132	202	149	150
4	171	234	208	159	138	212	157	154
5	170	237	211	161	140	144	159	154
6	165	140	198	151	132	135	149	144

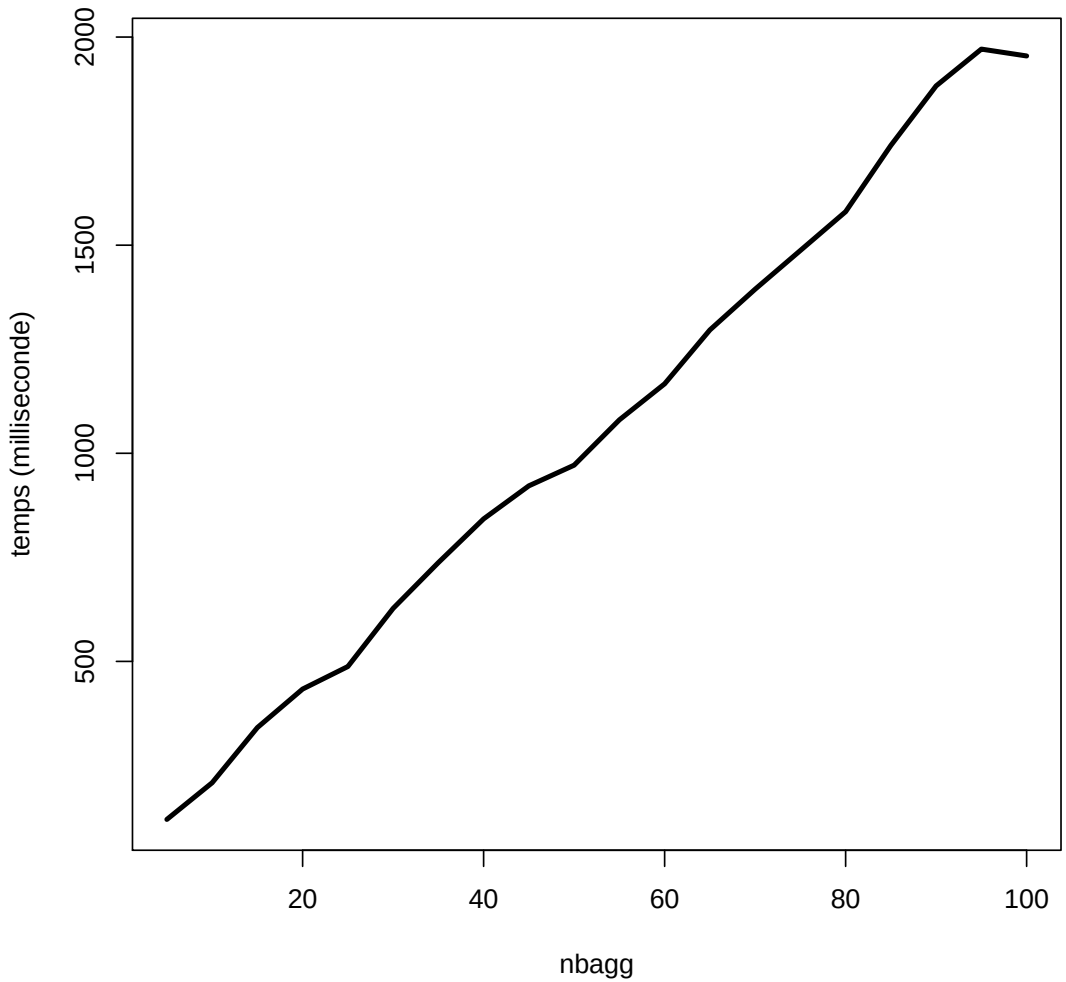
Exemple 1C : Bagging

```
library(microbenchmark)

FUN <- function(x)
  {
  mean(microbenchmark(arbre4bagging <- bagging(Y ~ VehAge
       + LicenceAge + DriverAge, data = dataAGG, nbagg = x),
       unit = "ms", times = 20)$time)/1000000
}

Temps <- sapply((1:20)*5, function(y) FUN(y))
```

Exemple 1C : Temps de calcul



Principe

- Il s'agit d'un autre (méta-)algorithme qui permet d'améliorer les performances des arbres de décision.
- Dans une situation où une des variables explicatives du modèle a un impact très fort sur la variable réponse, pratiquement tous les arbres impliqués dans une procédure de *Bagging* se verront divisés au même endroit ($\pm$) lors du premier stage de la construction $\rightarrow$ les prédictions obtenues seront très semblables.
- Les arbres obtenus lors d'une procédure de *Bagging* sont très fortement corrélés, ce qui limite la diminution de la variabilité des prédictions.

Algorithme

1. On génère B bases de données par un double processus d'échantillonnage :
   1. la base de données b, $b = 1, \ldots, B$, de taille n est obtenue en pigeant au hasard **avec remise**, n observations de la base de données initiale (*bootstrap*) ;
   2. pour la base de données b, $b = 1, \ldots, B$, on conserve uniquement $m \leq p$ variables explicatives (on prend souvent $m \approx \sqrt{p}$). Les variables conservées ne sont pas nécessairement les mêmes d'un échantillon à l'autre.
2. À partir de chacune des bases de données, construire un modèle $\widehat{f}_b(\mathbf{X}^*)$, $b = 1, \ldots, B$.
3. Pour une nouvelle observation $\mathbf{X}$, la prédiction sera

$$\widehat{f}_{rf}(\mathbf{X}) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}_b(\mathbf{X}^*).$$

Forêts aléatoires

- La procédure permet de « décorréler » les arbres obtenus $\rightarrow$ plus grande réduction de la variance possible.
- On a

$$\text{Corr}\left[\widehat{f}_{b_1}(\mathbf{X}), \widehat{f}_{b_2}(\mathbf{X})\right] <<< 1 \approx 0.$$

- Comme pour le *Bagging*, les résultats sont plus difficiles à interpréter : pas de graphique simple, etc.

Exemple 3 : Boston

```
### medv: valeur médiane des maisons en 1000$
library(MASS)
data(Boston)

head(Boston)
...     dis  rad  tax  ptratio  black  lstat  medv
    4.0900    1  296     15.3 396.90   4.98  24.0
... 4.9671    2  242     17.8 396.90   9.14  21.6
... 4.9671    2  242     17.8 392.83   4.03  34.7
... 6.0622    3  222     18.7 394.63   2.94  33.4
```

Exemple 3 : Boston

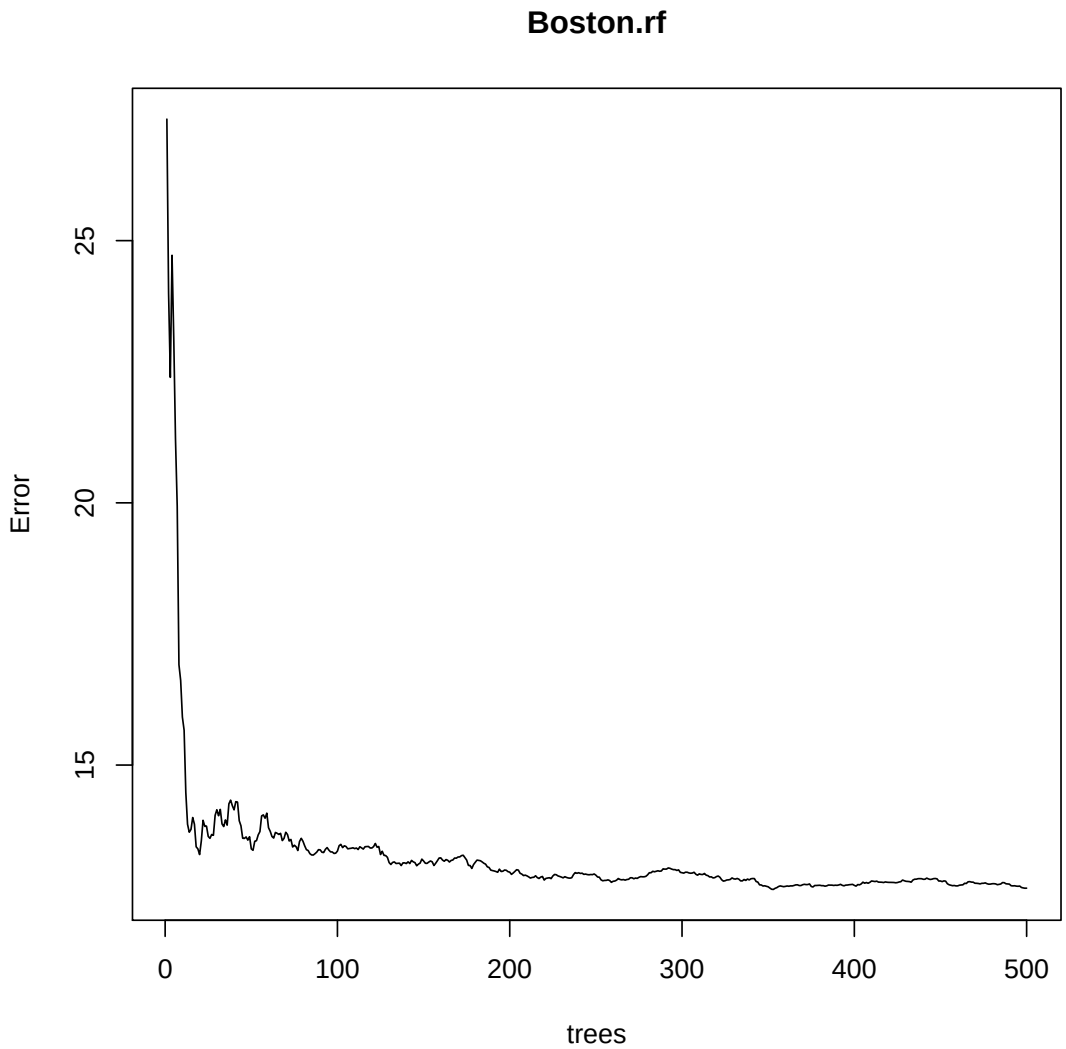
```
train <- sample(1:nrow(Boston), 300)
Boston.rf <- randomForest(medv ~ ., data = Boston,
                          subset = train)
Boston.rf

Type of random forest: regression
Number of trees: 500
No. of variables tried at each split: 4

Mean of squared residuals: 12.6529
% Var explained: 84.2

plot(Boston.rf)
```

Exemple 3 : Boston



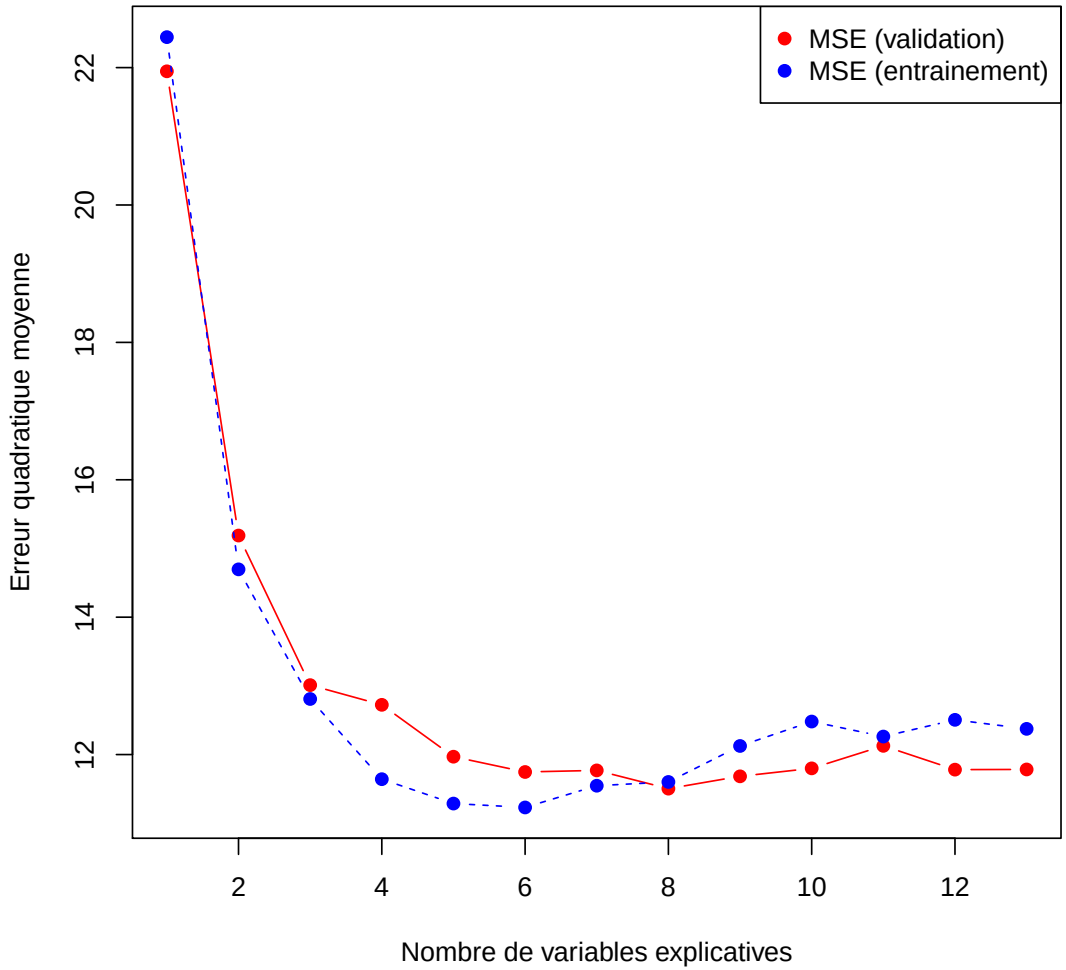
Exemple 3 : Boston

```
oob.err <- double(13)
test.err <- double(13)

for(mtry in 1:13)
{
  rf <- randomForest(medv ~ ., data = Boston, subset = train,
                     mtry = mtry, ntree = 400)
  oob.err[mtry] <- rf$mse[400]

  pred <- predict(rf, Boston[-train,])
  test.err[mtry] <- with(Boston[-train,],
                    mean((medv - pred)^2))
}
```

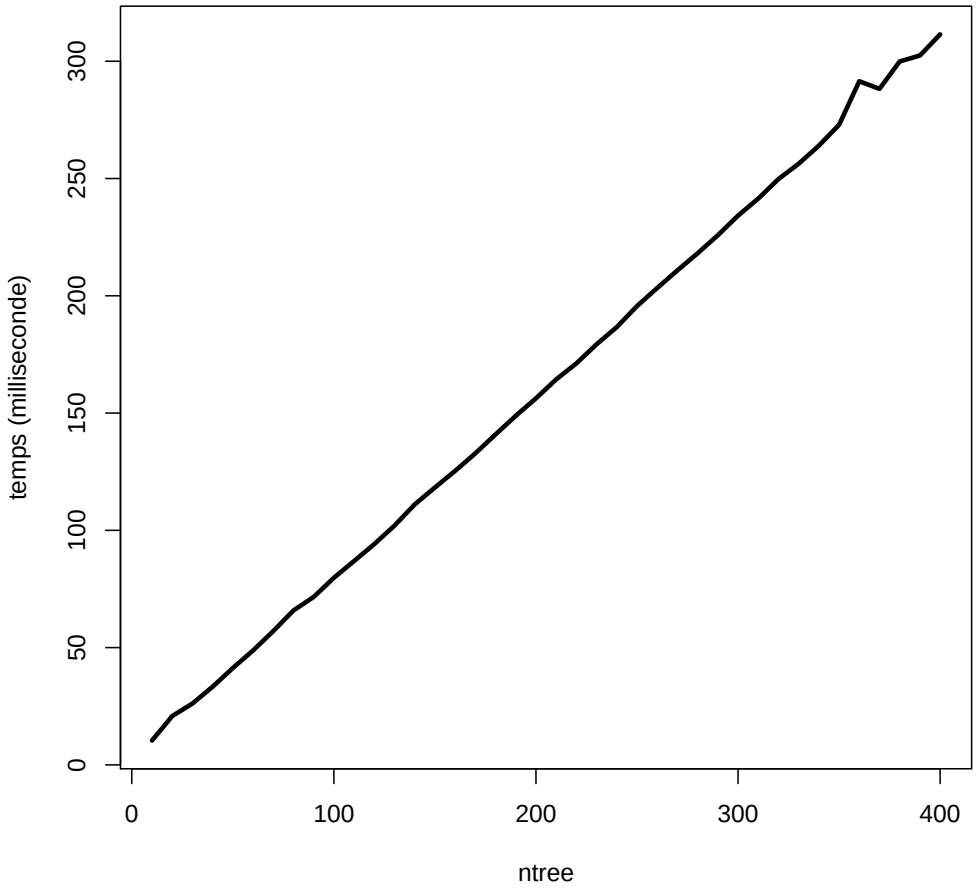
Exemple 3 : Boston



Exemple 3 : Temps de calcul (nombre d'arbres)

```
FUN <- function(x)
{
  mean(microbenchmark(rf <- randomForest(medv ~ .,
       data = Boston,
       subset = train, mtry = 6, ntree = x) ,
       unit = "ms", times = 20)$time)/1000000
}
TempsRF <- sapply((1:40)*10, function(y) FUN(y))
```

Exemple 3 : Temps de calcul (nombre d'arbres)



Exemple 3 : Temps de calcul (nombre de variables explicatives)

```
FUN <- function(x)
{
  mean(microbenchmark(rf <- randomForest(medv ~ .,
       data = Boston, subset = train, mtry = x, ntree = 400),
       unit = "ms", times = 20)$time)/1000000
}
TempsRF1 <- sapply((1:13), function(y) FUN(y))
```

Exemple 3 : Temps de calcul (nombre de variables explicatives)