Introduction aux arbres de décision (de type CART)

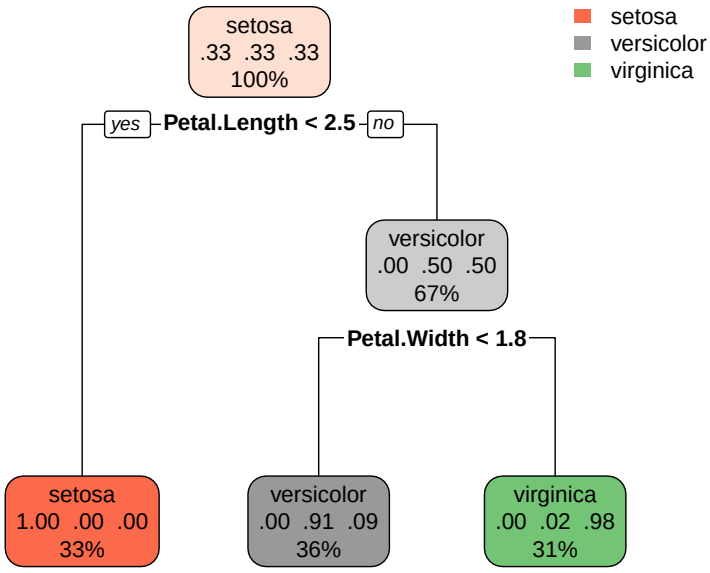
Christophe Chesneau

Université de Caen

Introduction aux arbres de décision (de type CART)

Christophe Chesneau

https://chesneau.users.lmno.cnrs.fr/



Caen, le 12 Février 2022

Table des matières

~ Note ~

Ce document propose une introduction aux arbres de décision de type CART.

On y aborde principalement les arbres de régression, les arbres de classification et les forêts aléatoires d'arbres de décision.

Le logiciel utilisé est R.

N'hésitez pas à me contacter pour tout commentaire :

`christophe.chesneau@gmail.com`

Bonne lecture !

1 Arbres de régression (CART)

1.1 Introduction

Contexte : Pour n individus $\omega_1, \dots, \omega_n$ d'une population, on dispose des valeurs de $p+1$ caractères $X_1, \dots, X_p, Y$. Pour tout $i \in \{1, \dots, n\}$, les valeurs associées à ω_i sont notées $x_{1,i}, \dots, x_{p,i}, y_i$. Elles sont généralement présentées sous la forme suivante :

	X_1	$\dots$	X_p	Y
ω_1	$x_{1,1}$	$\dots$	$x_{p,1}$	y_1
$\vdots$	$\vdots$	$\dots$	$\vdots$	$\vdots$
ω_n	$x_{1,n}$	$\dots$	$x_{p,n}$	y_n

Ces valeurs constituent les données. Dans ce chapitre, on suppose que Y est quantitative, on ne fait pas d'hypothèse particulière sur les autres caractères.

Objectif : Partant des données, l'objectif est de donner une valeur plausible de Y pour un individu dont on connaît les valeurs de $X_1, \dots, X_p$.

Aide à la décision : Pour décider d'une valeur plausible de Y, on peut s'aider d'un arbre de régression de type CART (plus précisément, reposant sur l'algorithme CART, acronyme pour Classification And Regression Trees).

Intérêts : Les intérêts d'un arbre de régression sont les suivants :

- Il est simple à comprendre, à interpréter et à communiquer.
- La nature des caractères $X_1, \dots, X_p$ n'a pas d'importance.
- Il est performant pour de grands jeux de données (big data).

Remarque : Pour atteindre le même objectif, on pourrait aussi utiliser un modèle de régression linéaire (ou non-linéaire). Les avantages de l'arbre de régression par rapport au modèle de régression linéaire sont les suivants :

- Il est plus simple et plus direct dans son approche.

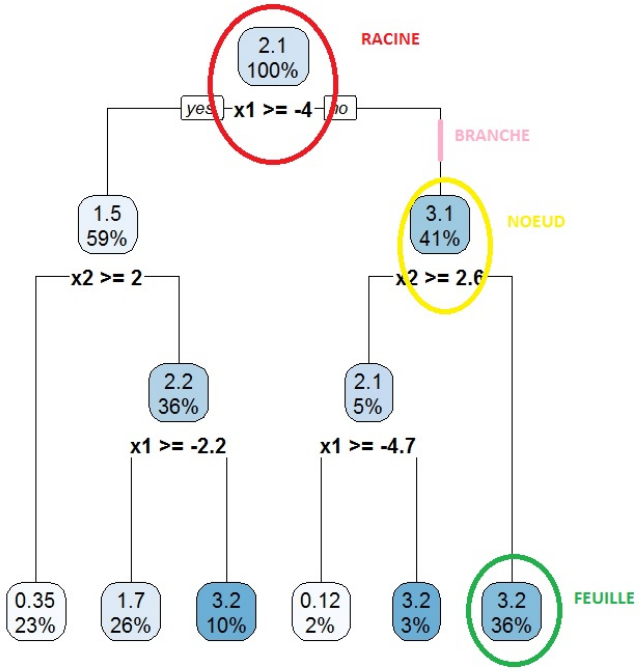
- La structure liant Y à $X_1, \ldots, X_p$ n'importe pas ; celle-ci peut être linéaire ou autre.
- Il n'y a pas d'hypothèse mathématique sous-jacente (pas d'hypothèse de normalité ou autre).
- Les dépendances éventuelles entre $X_1, \ldots, X_p$ ne posent pas de problème.

1.2 Première approche

Arbre binaire, racine, branche, nœud et feuille : Un arbre binaire est une construction hiérarchique de forme "triangulaire en escaliers" constitué de plusieurs éléments : l'élément fondateur est au sommet de la construction ; il est appelé racine, les traits qui partent en descendant de cette racine sont appelés branches, elles joignent des éléments appelés nœuds.

De chaque nœud, partent 0 ou 2 branches joignant alors d'autres nœuds, et ainsi de suite. Un nœud dont part 2 branches est dit coupé. Un nœud dont ne part aucune branche est appelé feuille. Ainsi, un arbre se parcours de la racine aux feuilles (donc de haut en bas).

Exemple : Un exemple d'arbre de régression est présenté ci-dessous.

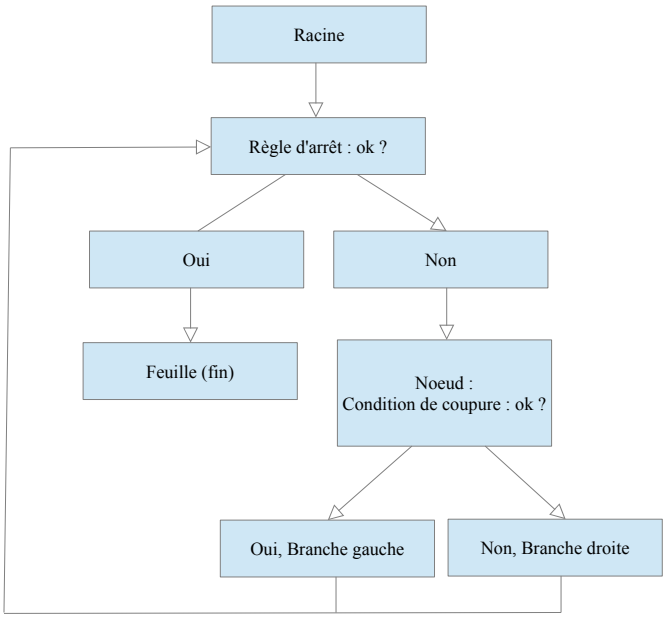


Arbre de régression : principe : Avant toute chose, un arbre de régression de type CART est un arbre binaire aidant à la décision d'une valeur plausible de Y pour un individu dont on connaît les valeurs $X_1, \ldots, X_p$. Sa construction repose sur un partitionnement récursif des individus à l'aide des données. Ce partitionnement se fait par une succession de nœuds coupés. La coupure d'un nœud, et ce qui la caractérise, se fait avec

- des règles d'arrêt,
- des conditions de coupure.

Les règles d'arrêts reposent souvent sur des principes simples (il faut un minimum d'individus au nœud pour envisager une coupure ...). Les conditions de coupures reposent sur des critères mathématiques. Une particularité est que chaque condition de coupure met en jeu un, et un seul, caractère parmi $X_1, \ldots, X_p$ (un même caractère peut être utilisé pour définir plusieurs conditions de coupure). À la fin du partitionnement, il ne reste plus que des feuilles. Les données des individus associés aux feuilles sont alors utilisées à des fins prédictives sur le caractère Y.

Le schéma général est résumé dans l'image suivante :



Commandes R : On utilise `rpart` de la librarie `rpart` (`rpart` est l'acronyme de `r`ecursive `part`itioning). Exemple de syntaxe avec $p = 2$: `arbre = rpart(Y ~ X1 + X2)`. Énventuellement, on peut activer l'option `method = "anova"` s'il peu y avoir confusion sur la nature quantitative de Y. Pour afficher l'arbre, on utilise `rpart.plot` de la librairie `rpart.plot`.

Utilisation : Pour déterminer une valeur plausible de Y pour un individu dont on connaît les valeurs $X_1, \ldots, X_p$, on procède étape par étape de la manière suivante. En partant de la racine, à chaque nœud, on vérifie si la condition de coupure est vérifiée ou pas : si la condition est vérifiée, on se dirige vers la branche associée à la réponse "Oui" (répondant à la question implicite "Est-ce la condition est vérifiée ?"), sinon, on se dirige vers la branche associée à la réponse "Non". À la dernière étape, on aboutit alors à une seule feuille de l'arbre. Dès lors, une valeur plausible de Y pour l'individu est la moyenne des valeurs de Y associées aux individus de cette feuille. En général, un arbre de régression construit avec un logiciel statistique affiche la moyenne des valeurs de Y et ce, pour chaque nœud et feuille.

Commandes R : On utilise `predict`.

1.3 Considérations mathématiques

Questions centrales : Ainsi, pour construire un arbre de régression, deux questions se posent :

- Comment définir la condition de coupure d'un nœud ?
- Comment définir la règle d'arrêt ?

Des réponses à ces questions sont apportées par des critères mathématiques. Pour les définir clairement, quelques outils sont à présenter, ce qui est fait ci-dessous.

Dans un premier temps, on suppose que les caractères $X_1, \ldots, X_p$ sont quantitatifs.

Outils : Soient $j \in \{1, \ldots, p\}$ et $c \in \mathbb{R}$. Pour un nœud donné, on pose :

- $\overline{y}_{j,c,gauche}$ la moyenne des valeurs de Y pour les individus vérifiant $X_j < c$,
- $\overline{y}_{j,c,droit}$ la moyenne des valeurs de Y pour les individus vérifiant $X_j \geq c$,
- $SS_{gauche}(j, c)$ la somme des carrés des écarts entre les valeurs de Y et $\overline{y}_{j,c,gauche}$, pour les individus vérifiant $X_j < c$,

- $SS_{droit}(j,c)$ la somme des carrés des écarts entre les valeurs de Y et $\overline{y}_{j,c,droit}$, pour les individus vérifiant $X_j \geq c$.

Idée : Pour un nœud donné, l'erreur globale que l'on commet en séparant les individus selon que $X_j < c$ ou $X_j \geq c$ est donnée par

$$E(j,c) = SS_{gauche}(j,c) + SS_{droit}(j,c).$$

L'idée est donc de minimiser cette erreur.

Condition de coupure : Pour un nœud donné, la condition de coupure adoptée est

$$X_{j_*} \geq c_*,$$

où j_* et c_* rendent minimale l'erreur $E(j,c)$, i.e, pour tout $j \in \{1,\ldots,p\}$ et $c \in \mathbb{R}$, $E(j_*,c_*) \leq E(j,c)$. Le caractère X_{j_*} est alors appelé caractère de coupure et c_* est appelée valeur seuil. Ainsi, on sépare les individus en 2 groupes selon qu'ils vérifient $X_{j_*} < c_*$ ou $X_{j_*} \geq c_*$; X_{j_*} est le plus influant des caractère quant à la séparation des individus du nœud en 2 groupes. En outre, le plus influant des caractères sur Y quant à la séparation des individus du nœud en 2 groupes est le caractère de coupure de la racine.

Remarque :

- En général, la maximisation sur c ne se fait pas sur $\mathbb{R}$ tout entier ; elle se fait sur une grille de valeurs données par les moyennes de deux observations consécutives du caractère X_j.
  Commandes R : `rpart` fonctionne ainsi pour déterminer les valeurs seuils candidates.
- On peut également écrire la condition de coupure comme $X_{j_*} < c_*$, puisque seule la réponse "Oui" ou "Non" importe dans une prédiction. Si $X_{j_*} \geq c_*$ est "Oui", $X_{j_*} < c_*$ sera "Non", et vice-versa.

Indice d'amélioration : Pour un nœud donné, on appelle indice d'amélioration le réel I défini par

$$I = 1 - \frac{E(j_*,c_*)}{E},$$

où E désigne la somme des carrés des écarts entre les valeurs de Y et la moyenne de celles-ci

(pour les individus du nœud considéré). En outre, plus I est proche de 1, plus l'amélioration générée par la condition de coupure du nœud est forte.

Commandes R : Les indices d'amélioration sont précisés dans la sortie de `summary(arbre)` par les valeurs de `improve`.

Règle d'arrêt : Il existe plusieurs règles d'arrêt, lesquelles peuvent se combiner. Entre autre, on peut se fixer

- la profondeur de l'arbre (quantifiée par les premiers niveaux/étages de l'arbre à partir de la racine initialisée à 0),
- le nombre minimum d'individus présents à l'étape d'un nœud pour envisager une coupure (en dessous de ce nombre, le nœud devient alors une feuille),
- nombre minimum d'individus présents à l'étape d'un nœud engendré par la coupure du "nœud parent" (en dessous de ce nombre, le nœud parent devient alors une feuille),
- la valeur d'un paramètre de complexité noté cp ; en guise de première approche, disons que plus le cp est petit, plus grand est l'arbre de régression.

Commandes R : Avec `rpart`, on peut fixer

- la profondeur de l'arbre avec la commande `maxdepth`,
- le nombre minimum d'individus présents à l'étape d'un nœud pour envisager une coupure avec la commande `minsplit`,
- le nombre minimum d'individus présents à l'étape d'un nœud que engendrerait la coupure du nœud parent avec la commande `minbucket`,
- la valeur de cp avec la commande `cp`.

Par défaut, la commande `rpart` pose : `maxdepth = 30`, `minsplit = 20`, `minbucket = minsplit / 3` et `cp = 0.1`.

Élagage (pruning en anglais) : Une fois l'arbre de régression construit, si le nombre de feuilles est jugé trop grand, on peut le simplifier en élaguant ses branches de bas en haut. Un élagage judicieux s'arrête quand on atteint un bon compromis entre la complexité de l'arbre et la précision de la prédiction.

Ce compromis peut se faire à l'aide d'une méthode de validation croisée testant différentes versions élaguées de l'arbre. Un élagage judicieux correspond à une valeur du paramètre de complexité *cp* rendant petite une certaine erreur appelée erreur de validation croisée ou xerror.

On peut alors utiliser cette valeur de *cp* comme règle d'arrêt dans un nouvel arbre de régression, donnant ainsi une version élaguée de l'arbre initial.

Commandes R : Pour déterminer le *cp* optimal, on utilise `plotcp`. On utilise `prune` pour élaguer.

Cas de caractères qualitatifs : Si un ou plusieurs caractères parmi $X_1, \dots, X_p$ sont qualitatifs, le schéma général dessus s'applique, mais la condition $X_j \geq c$ devient $X_j = m$, où m désigne une des modalités du caractère. Ainsi, on sépare les individus en 2 groupes suivant que $X_j = m$ ou $X_j \neq m$.

1.4 Mise en œuvre avec R

Exemple 1 : On utilise le jeu de données `data-arbre`. D'abord, on charge ce jeu de données et on le décrit brièvement avec les commandes suivantes :

```
w = read.csv("https://chesneau.users.lmno.cnrs.fr/data-arbre.csv", header =
T, sep = ";")
attach(w)
str(w)
```

Cela renvoie :

```
'data.frame':   400 obs. of  3 variables:
$ x1: num  1.5915 2.2424 0.0987 2.147 1.6422 ...
$ x2: num  -0.968 0.549 -0.9 0.682 -0.473 ...
$ Y : num  1.107 1.474 2.15 0.999 1.955 ...
```

Ainsi, ce jeu de données contient 400 observations de 3 caractères quantitatifs notés $x1$, $x2$ et Y.

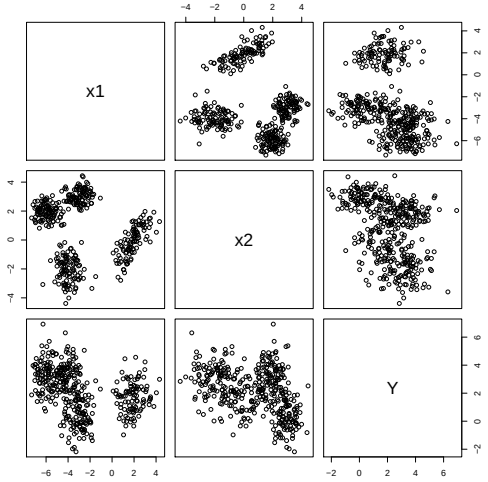
L'objectif est de donner une valeur plausible de Y pour un individu dont on connait les valeurs de $x1$ et $x2$.

On peut alors utiliser un arbre de régression suivant la méthode CART. Mais avant, à titre de remarque, on peut éventuellement tenter d'identifier le caractère de coupure à l'aide de graphiques élémentaires.

On fait :

```
pairs(w)
```

Cela renvoie :



Si l'on analyse les nuages de points associés à $(x1, Y)$ et $(x2, Y)$, il semble que $x1$ ait une influence plus nette que $x2$ sur la grandeur des valeurs de Y ; sur le nuage de points associé à $(x1, Y)$, on distingue deux "boules de points" bien séparées, contrairement au nuage de points associé à $(x2, Y)$, où une telle séparation est moins nette. On peut donc penser que $x1$ est le principal caractère de coupure dans l'explication des valeurs de Y, chose à confirmer dans l'arbre de régression.

Pour déterminer l'arbre, on fait :

```
library(rpart)

arbre = rpart(Y ~ x1 + x2)

arbre
```

Cela renvoie le nécessaire numérique pour construire l'arbre de régression, à savoir :

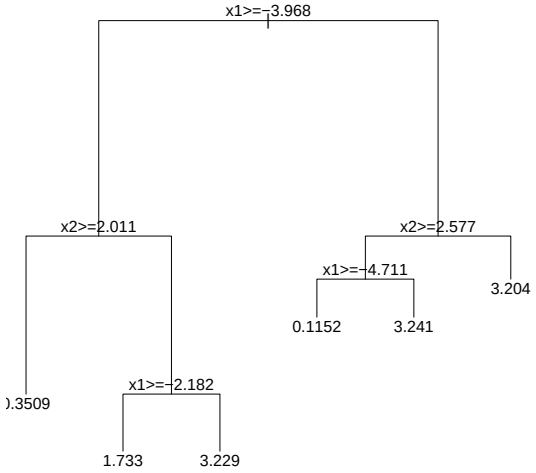
```
 n= 400


node), split, n, deviance, yval
      * denotes terminal node


 1) root 400 1048.064000 2.1294040
   2) x1>=-3.967763 235  526.937500 1.4651350
     4) x2>=2.011389 91  111.131800 0.3508848 *
     5) x2< 2.011389 144  231.426600 2.1692780
      10) x1>=-2.181836 102  120.766200 1.7327210 *
      11) x1< -2.181836 42   44.011010 3.2294890 *
   3) x1< -3.967763 165  269.745900 3.0754850
     6) x2>=2.576998 20   68.207690 2.1467220
      12) x1>=-4.711209 7    2.379668 0.1151748 *
      13) x1< -4.711209 13   21.381430 3.2406320 *
     7) x2< 2.576998 145  181.906600 3.2035900 *
```

Ici, `node)` représente le numéro du nœud, `split` présente la condition de coupure, puis, pour un nœud donné : `n` est le nombre d'individus, `deviance` est la somme des carrés des écarts des valeurs de Y et `yval` est la moyenne des valeurs de Y (pour les individus présents à l'étape du nœud). Le symbole $*$ parfois présent à la fin précise que le nœud est une feuille.

On peut visualiser cet arbre en faisant :

```
plot(arbre)
text(arbre)
```

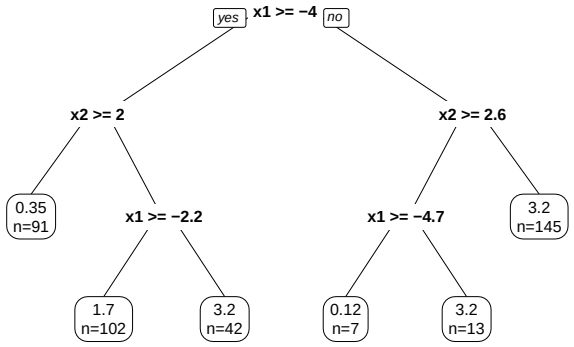
Cela renvoie :



Comme le rendu est peu esthétique, en général, on utilise la commande `rpart.plot` ou `prp` de la librairie `rpart.plot`. Avec `prp`, on fait :

```
library(rpart.plot)

prp(arbre, extra = 1)
```

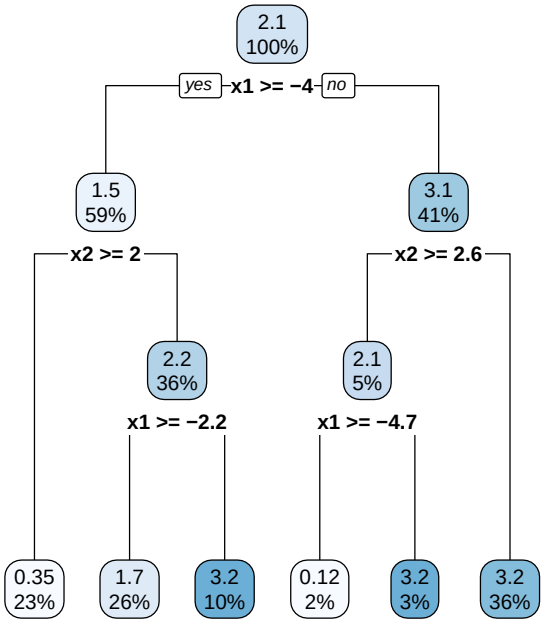
Cela renvoie :



On peut afficher un arbre plus esthétique et informatif avec `rpart.plot` :

```
library(rpart.plot)

rpart.plot(arbre)
```

Cela renvoie :



L'arbre s'utilise comme suit. Par exemple, pour donner une valeur plausible y_* de Y d'un individu vérifiant $x1 = 3$ et $x2 = 1$, on part de la racine et la première question qui se pose est : "Est-ce que $x1 \geq -4$?" La réponse est "Oui", donc on choisit la branche de gauche, qui amène une nouvelle question : "Est-ce que $x2 \geq 2$?" La réponse est "Non", on choisit donc la branche de droite, qui amène une nouvelle question : "Est-ce que $x2 \geq -2.2$?" La réponse est "Oui", on choisit donc la branche de gauche et le nœud associé est un feuille qui affiche le résultat final : une valeur plausible de Y pour notre individu est

$$y_* = 1.7.$$

On rappelle que y_* est la moyenne des valeurs de Y pour les individus satisfaisant les mêmes conditions sur $x1$ et $x2$. La feuille précise que ces individus représente 26% de la totalité des individus du jeu de données.

On peut retrouver ce résultat directement avec la commande `predict` :

```
predict(arbre, newdata = data.frame(x1 = 3, x2 = 1))
```

Cela renvoie :

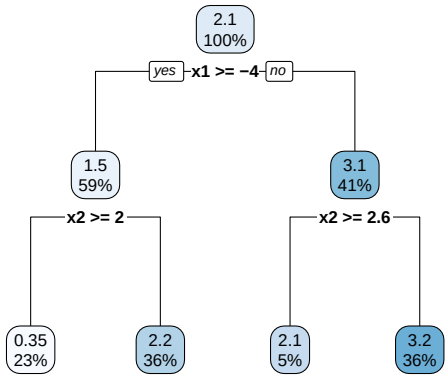
```
       1
1.732721
```

D'où $y_* = 1.7$.

On peut également faire varier les règles d'arrêt à titre informatif (pour le moment). Par exemple, si l'on veut une profondeur d'arbre de 2, on fait :

```
library(rpart)
arbre = rpart(Y ~ x1 + x2, maxdepth = 2)
arbre
```

Cela renvoie :



On a ainsi laisser que les "2 premiers niveaux" de l'arbre, à partir de la racine initialisée à 0.

Exemple 2 : On s'intéresse au jeu de données `Hitters` de la librairie `ISLR`. Celui-ci provient de la ligue majeure de baseball en 1986 et 1987.

```
library(ISLR)
Hitters = na.omit(Hitters)
attach(Hitters)
str(Hitters)
```

Cela renvoie, entre autre :

```
'data.frame':   263 obs. of  20 variables:
 $ AtBat    : int  315 479 496 321 594 185 298 323 401 574 ...
 $ Hits     : int  81 130 141 87 169 37 73 81 92 159 ...
 $ HmRun    : int  7 18 20 10 4 1 0 6 17 21 ...
 $ Runs     : int  24 66 65 39 74 23 24 26 49 107 ...
 $ RBI      : int  38 72 78 42 51 8 24 32 66 75 ...
 $ Walks    : int  39 76 37 30 35 21 7 8 65 59 ...
 $ Years    : int  14 3 11 2 11 2 3 2 13 10 ...
 $ CAtBat   : int  3449 1624 5628 396 4408 214 509 341 5206 4631 ...
 $ CHits    : int  835 457 1575 101 1133 42 108 86 1332 1300 ...
 $ CHmRun   : int  69 63 225 12 19 1 0 6 253 90 ...
 $ CRuns    : int  321 224 828 48 501 30 41 32 784 702 ...
 $ CRBI     : int  414 266 838 46 336 9 37 34 890 504 ...
 $ CWalks   : int  375 263 354 33 194 24 12 8 866 488 ...
 $ League   : Factor w/ 2 levels "A","N": 2 1 2 2 1 2 1 2 1 1 ...
 $ Division : Factor w/ 2 levels "E","W": 2 2 1 1 2 1 2 2 1 1 ...
 $ PutOuts  : int  632 880 200 805 282 76 121 143 0 238 ...
 $ Assists  : int  43 82 11 40 421 127 283 290 0 445 ...
 $ Errors   : int  10 14 3 4 25 7 9 19 0 22 ...
 $ Salary   : num  475 480 500 91.5 750 ...
```

```
 $ NewLeague: Factor w/ 2 levels "A","N": 2 1 2 2 1 1 1 2 1 1 ...
```

Ici, `Salary` est le salaire annuel en milliers de dollars d'un joueur au début de la saison de 1987, `Years` est le nombre d'années où le joueur évolue en ligue majeure et `Hits` est le nombre de frappes effectuées en 1986. On cherche à donner une valeur plausible y_* de `Salary` pour un individu dont on connait les valeurs de `Years` et `Hits`. Pour ce faire, on propose d'utiliser un arbre de régression de type CART. Pour déterminer l'arbre de régression, on utilise `rpart` :

```
library(rpart)
arbre = rpart(Salary ~ Years + Hits)
arbre
```

Cela renvoie le nécessaire numérique pour construire l'arbre de régression, à savoir :

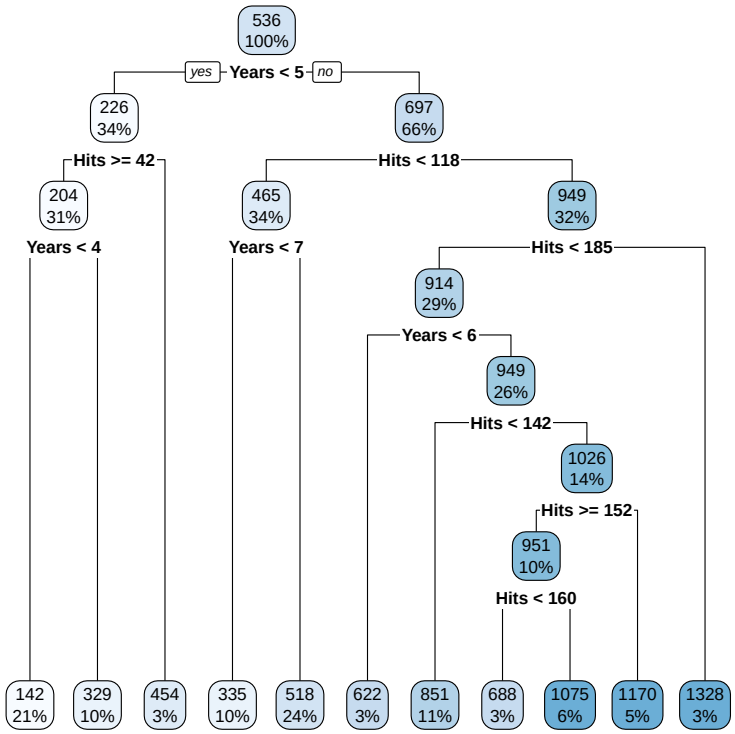
```
n= 263

node), split, n, deviance, yval
      * denotes terminal node

  1) root 263 53319110.0  535.9259
    2) Years< 4.5 90  6769171.0  225.8315
      4) Hits>=42 82  2521881.0  203.5366
        8) Years< 3.5 55   299487.2  141.8182 *
        9) Years>=3.5 27  1586123.0  329.2593 *
      5) Hits< 42 8  3788751.0  454.3541 *
    3) Years>=4.5 173 33393450.0  697.2467
      6) Hits< 117.5 90  5312120.0  464.9167
       12) Years< 6.5 26   644133.6  334.7115 *
       13) Years>=6.5 64  4048129.0  517.8125 *
      7) Hits>=117.5 83 17955720.0  949.1708
       14) Hits< 185 76 13290200.0  914.3246
```

```
   28) Years< 5.5 8    82787.5  622.5000 *
   29) Years>=5.5 68 12445970.0  948.6570
     58) Hits< 141.5 30  3091490.0  850.9634 *
     59) Hits>=141.5 38  8842112.0 1025.7830
      118) Hits>=151.5 25  3785290.0  950.7324
        236) Hits< 159.5 8   359226.1  687.5595 *
        237) Hits>=159.5 17  2611241.0 1074.5780 *
      119) Hits< 151.5 13  4645204.0 1170.1120 *
 15) Hits>=185 7  3571312.0 1327.5000 *
```

On peut visualiser cet arbre avec la commande `rpart.plot` de la librairie `rpart.plot` :

```
library(rpart.plot)

rpart.plot(arbre)
```

Cela renvoie :



Ainsi, par exemple, une valeur plausible de `Salary` pour un individu vérifiant `Years` = 6 et `Hits` = 145, on fait :

```
predict(arbre, newdata = data.frame(Years = 6, Hits = 145))
```

Cela renvoie :

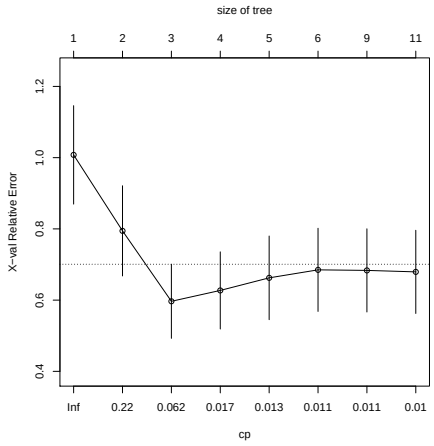
```
       1
1170.112
```

Ainsi, la valeur prédite de `Salary` pour l'individu concerné est $y_* = 1170.112$.

On peut simplifier l'arbre en procédant à un élagage. Pour ce faire, on détermine le coefficient de complexité optimal en faisant :

```
plotcp(arbre)
```

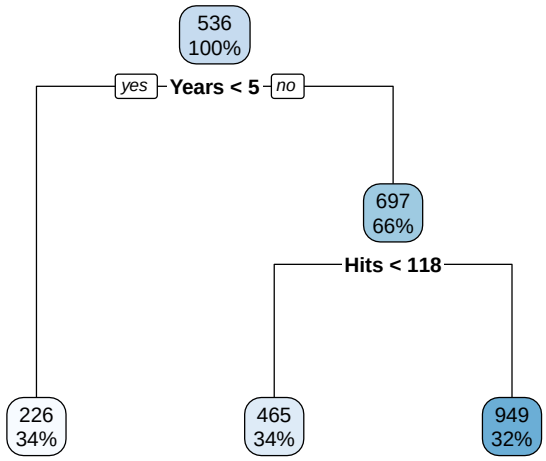
Cela renvoie :



La valeur optimale pour *cp* est celle qui minimise l'erreur relative. Le graphique nous indique que cette valeur est proche de $cp = 0.062$. Nous allons utiliser cette valeur comme règle d'arrêt dans un nouvel arbre de régression, élaguant ainsi l'arbre initial. On peut utiliser la commande `prune` :

```
arbre2 = prune(arbre, cp = 0.062)

rpart.plot(arbre2)
```

Cela renvoie :



On a alors un arbre simplifié qui contient l'essentiel de l'information contenue dans l'arbre initial.

Exemple 3 : Dans cet exemple, on considère le jeu de données `iris` :

```
data("iris")

str(iris)
```

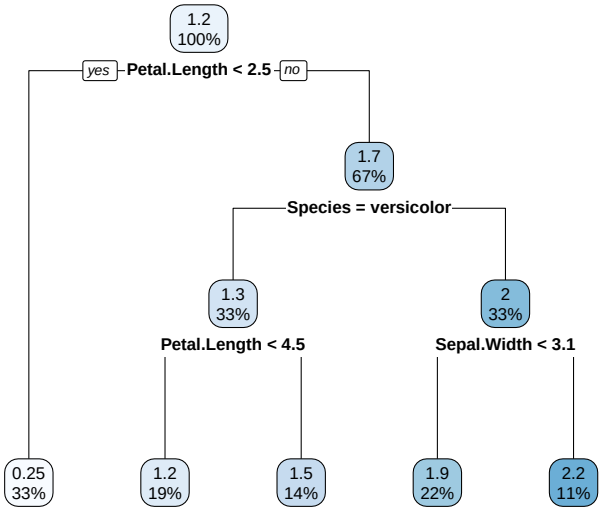
Cela renvoie :

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

On cherche à donner une valeur plausible y_* de `Petal.Width` pour un individu dont on connait les valeurs des autres caractères présents. Parmi ces derniers, il y a un caractère qualitatif qui est `Species`, ce qui est la seule nouveauté de l'exemple par rapport aux précédents, ce qui ne change rien quant à la méthode. On utilise alors un arbre de régression de type CART :

```
library(rpart)
library(rpart.plot)
arbre = rpart(Petal.Width ~., data = iris)
rpart.plot(arbre)
```

Cela renvoie :



On remarque alors comment est traité le caractère qualitatif : à la deuxième étape, c'est le caractère de coupure et la modalité seuil est `versicolor`.

Par exemple, une valeur plausible de `Petal.Width` pour un individu vérifiant `Sepal.Length` $= 5.7$, `Sepal.Width` $= 3.5$, `Petal.Length` $= 1.6$ et `Species` $=$ `Setosa`, on fait :

```
predict(arbre, newdata = data.frame(Sepal.Length = 5.7, Sepal.Width = 3.5,
Petal.Length = 1.6, Species = "setosa"))
```

Cela renvoie :

```
    1
0.246
```

Ainsi, la valeur prédite de `Petal.Width` pour l'individu concerné est $y_* = 0.246$.

À titre de curiosité, on peut s'intéresser à la valeur prédite obtenue à l'aide du modèle de régression logistique, dont la forme générique est :

$$\texttt{Petal.Width} = \beta_0 + \beta_1 \texttt{Sepal.Length} + \beta_2 \texttt{Sepal.Width} + \beta_3 \texttt{Petal.Length} + \beta_4 \texttt{Species} + \epsilon,$$

où β_0, β_1, β_2, β_3 et β_4 sont des coefficients à estimer et ϵ et une variable d'erreur de moyenne nulle. On peut estimer β_0, β_1, β_2 et β_3 avec la méthode des moindres carrés ordinaires, ce qui nous donne des estimateurs ponctuels notés b_0, b_1, b_2, b_3 et b_4, puis on peut déterminer la valeur prédite en faisant :

$$y_* = b_0 + b_1 \texttt{Sepal.Length} + b_2 \texttt{Sepal.Width} + b_3 \texttt{Petal.Length} + b_4 \texttt{Species},$$

avec `Sepal.Length` $= 5.7$, `Sepal.Width` $= 3.5$, `Petal.Length` $= 1.6$ et `Species` $=$ `Setosa`.

On obtient cette valeur en faisant :

```
reg = lm(Petal.Width ~., data = iris)
predict(reg, newdata = data.frame(Sepal.Length = 5.7, Sepal.Width = 3.5,
Petal.Length = 1.6, Species = "setosa"))
```

Cela renvoie :

```
        1
0.2323665
```

Ainsi, on obtient $y_* = 0.2323665$. Cette valeur est très proche de celle obtenue avec l'arbre de régression. L'avantage de l'arbre de régression est que l'on voit en un coup d'œil quels sont les caractères qui discriminent le plus les individus, avec la valeur seuil associée.

(À partir du modèle de régression linéaire multiple, on pourrait étudier la significativité des caractères explicatifs sur Y avec des tests de Student adaptés, mais ceux-ci ne sont valables que si certaines hypothèses sur les résidus sont vérifiées. Et ces conditions ne sont pas vérifiées pour le jeu de données `iris`, par exemple. Essayer : `e = residuals(reg)`, puis `plot(e)`. Cela renvoie un nuage de points dont les points ne sont pas uniformément répartis ; on distingue une structure (en forme de mégaphone) ; les hypothèses standards ne sont pas vérifiées).

2 Arbres de classification (CART)

Dans ce chapitre, la notion d'arbre de régression est supposée connue.

2.1 Présentation

Contexte : Pour n individus $\omega_1, \ldots, \omega_n$ d'une population, on dispose des valeurs de $p+1$ caractères $X_1, \ldots, X_p, Y$. Pour tout $i \in \{1, \ldots, n\}$, les valeurs associées à ω_i sont notées $x_{1,i}, \ldots, x_{p,i}, y_i$. Elles sont généralement présentées sous la forme suivante :

	X_1	$\ldots$	X_p	Y
ω_1	$x_{1,1}$	$\ldots$	$x_{p,1}$	y_1
$\vdots$	$\vdots$	$\ldots$	$\vdots$	$\vdots$
ω_n	$x_{1,n}$	$\ldots$	$x_{p,n}$	y_n

Ces valeurs constituent les données. Dans ce chapitre, on suppose que Y est qualitative, on ne fait pas d'hypothèses particulières sur les autres caractères.

Objectif : Partant des données, l'objectif est de donner une modalité plausible de Y pour un individu dont on connaît les valeurs de $X_1, \ldots, X_p$.

Aide à la décision : Pour décider d'une modalité plausible de Y, on peut s'aider d'un arbre de classification de type CART.

Intérêts : Les intérêts d'un arbre de classification sont les mêmes que pour l'arbre de régression (simple à comprendre...).

Remarque : Pour atteindre le même objectif, on pourrait aussi utiliser un modèle de régression logistique ou multinomial. Les avantages de l'arbre de régression par rapport au modèle de régression sont les suivants :

- Il est plus simple à interpréter ; la communication des résultats est simplifié.
- La structure liant Y à $X_1, \ldots, X_p$ importe peu ; celle-ci peut être linéaire ou pas.
- Il n'y a pas d'hypothèse mathématique sous-jacente (pas d'hypothèse de normalité ou autre).
- Les dépendances éventuelles entre $X_1, \ldots, X_p$ ne posent pas de problème.

Arbre de classification : Le principe et la construction d'un arbre de classification sont les mêmes que pour les arbres de régression, seuls les outils mathématiques définissant la condition de coupure changent pour s'adapter à la nature qualitative du caractère.

Commandes R : On utilise `rpart` de la librarie `rpart`. Énventuellement, on peut activer l'option `method = "class"` s'il peut y avoir une confusion sur la nature qualitative de Y. Pour afficher l'arbre, on utilise `rpart.plot` de la librairie `rpart.plot`.

Utilisation : La détermination d'une modalité plausible de Y pour un individu dont on connait les valeurs de $X_1, \ldots, X_p$ est affichée par la feuille obtenue à la fin du cheminement étape par étape. Cette modalité correspond à celle la plus observée parmi les individus associés à cette feuille.

Commandes R : On utilise `predict`.

2.2 Considérations mathématiques

Dans un premier temps, pour simplifier, on suppose que tous les caractères $X_1, \ldots, X_p$ sont quantitatifs.

Outils : Soit $j \in \{1, \ldots, p\}$, $c \in \mathbb{R}$, M le nombre de modalités de Y et $i \in \{1, \ldots, M\}$. Pour un nœud donné, on pose :

- $n_{j,c,gauche}$ est le nombre d'individus vérifiant $X_j < c$,
- $n_{j,c,droit}$ est le nombre d'individus vérifiant $X_j \geq c$,
- $f_{i,j,c,gauche}$ est la fréquence de la i-ème modalité de Y pour les individus vérifiant $X_j < c$,
- $f_{i,j,c,droit}$ est la fréquence de la i-ème modalité de Y pour les individus vérifiant $X_j \geq c$,
- $G_{gauche}(j,c) = 1 - \sum_{i=1}^{M} f_{i,j,c,gauche}^2$,
- $G_{droit}(j,c) = 1 - \sum_{i=1}^{M} f_{i,j,c,droit}^2$.

Idée : Pour un nœud donné, une perte globale d'information que l'on obtient en séparant les individus selon que $X_j < c$ ou $X_j \geq c$ est donnée par

$$G(j,c) = \frac{n_{j,c,gauche}}{n} G_{gauche}(j,c) + \frac{n_{j,c,droit}}{n} G_{droit}(j,c).$$

L'idée est donc de minimiser cette perte.

Condition de coupure : Pour un nœud donné, la condition de coupure adoptée est

$$X_{j_*} \geq c_*,$$

où j_* et c_* rendent minimal $G(j,c)$, i.e, pour tout $j \in \{1,\dots,p\}$ et $c \in \mathbb{R}$, $G(j_*,c_*) \leq G(j,c)$. Le caractère X_{j_*} est alors appelé caractère de coupure et c_* est appelée valeur seuil. Ainsi, on sépare les individus en 2 groupes selon qu'ils vérifient $X_{j_*} < c_*$ ou $X_{j_*} \geq c_*$.

Remarques :

- Le réel $G(j,c)$ est appelé indice moyen de Gini. Il en existe un autre défini avec ce que l'on appelle entropie.
- En général, la maximisation sur c ne se fait pas sur $\mathbb{R}$ tout entier ; elle se fait sur une grille de valeurs données par les moyennes de deux observations consécutives du caractère X_j.
  Commandes R : `rpart` fonctionne ainsi pour déterminer les valeurs seuils candidates.
- On peut traiter les caractères qualitatifs et faire un élagage de l'arbre de la même manière que pour un arbre de régression.

Indice d'amélioration : Pour un nœud donné, on appelle indice d'amélioration le réel J défini par

$$J = n(G - G(j_*,c_*)),$$

où $G = 1 - \sum_{i=1}^{M} f_i^2$ et f_i désigne la fréquence de la i-ème modalité de Y (pour les individus du nœud considéré). En outre, plus l'indice d'amélioration est grand, plus l'amélioration générée par la condition de coupure du nœud est forte.

Commandes R : Les indices d'amélioration sont précisés dans la sortie de `summary(arbre)` par les valeurs de `improve`.

Taux erreur global : On appelle taux d'erreur global de l'arbre de classification le rapport entre le nombre de fois où l'arbre s'est trompé dans la classification des individus du jeu de données et le nombre total d'individus. Plus ce taux est petit, plus la qualité prédictive de l'arbre est bonne.

Méthode apprentissage-test : L'arbre de classification étant construit avec tous les individus, y compris ceux qui seront finalement mal classés, le taux d'erreur global est légèrement faussé. Une alternative satisfaisante est apportée par la méthode apprentissage-test.

La méthode apprentissage-test consiste à construire un arbre de classification dont on peut évaluer la qualité prédictive en utilisant des "nouvelles données". On procède de la manière suivante. D'abord, on sélectionne aléatoirement quelques d'individus du jeu de données (en général, un cinquième de la totalité des individus). Ensuite, on construit 2 jeux de données : l'un avec les individus non sélectionnés, appelé jeu de données d'apprentissage, et l'autre avec les individus sélectionnés, appelé jeu de données test. On construit alors l'arbre de classification avec le jeu de données d'apprentissage (cet arbre peut donc différer de celui obtenu avec tous les individus). Puis, on utilise le jeu de données test pour étudier la qualité prédictive de l'arbre ainsi construit.

Dès lors, on appelle erreur apprentissage-test le rapport entre nombre de fois où l'arbre s'est trompé dans la classification des individus du jeu de données test et le nombre total d'individus du jeu de données test. Plus ce taux est petit, plus la qualité prédictive de l'arbre est bonne. On peut alors utilisé ce nouvel arbre de classification pour faire de la prédiction.

2.3 Mise en œuvre avec R

Exemple 1 : Dans cet exemple, on considère le jeu de données `iris` :

```
data("iris")
```

On cherche à donner une modalité plausible y_* de `Species`, caractère qualitatif à 3 modalités : `setosa`, `versicolor` et `virginica` pour un individu dont on connait les valeurs des autres caractères présents. On utilise alors un arbre de classification de type CART :

```
library(rpart)
arbre = rpart(Species ~., data = iris)
arbre
```

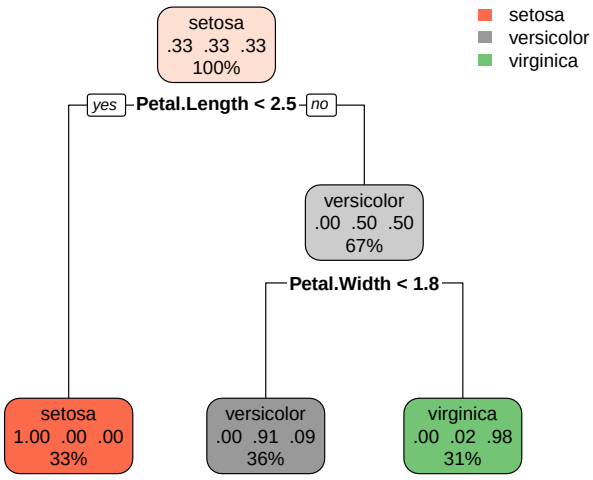
Cela renvoie :

```
n= 150


node), split, n, loss, yval, (yprob)
      * denotes terminal node


1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
  2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
  3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
    6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
    7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

On a alors tous les détails permettant la construction de l'arbre. On peut l'afficher en faisant :

```
library(rpart.plot)
rpart.plot(arbre)
```

Cela renvoie :



Par exemple, une valeur plausible de `Species` pour un individu vérifiant `Sepal.Length` $= 5.1$, `Sepal.Width` $= 3.5$, `Petal.Length` $= 1.6$ et `Petal.Width` $= 0.2$, on fait :

```
predict(arbre, newdata = data.frame(Sepal.Length = 5.1, Sepal.Width = 3.5,
Petal.Length = 1.6, Petal.Width = 0.2), type = "class")
```

Cela renvoie :

```
     1
setosa
Levels: setosa versicolor virginica
```

Ainsi, une modalité plausible de `Species` pour l'individu concerné est $y_* =$ `setosa`.

On peut voir le nombre de fois où l'arbre de classification se trompe avec les données de bases sous la forme d'un tableau (appelé tableau ou matrice de confusion). On fait :

```
tab = table(predict(arbre, type = "class"), iris$Species)
tab
```

Cela renvoie :

```
            setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         49         5
  virginica       0          1        45
```

On constate alors que l'arbre de classification est performant, avec un taux d'erreur global très faible, i.e., $t = (1 + 5)/(50 + 49 + 45 + 5 + 1) = 0.04$.

À titre de curiosité, on peut s'intéresser à la modalité prédite obtenue à l'aide du modèle de régression multinomial. On obtient cette valeur en faisant :

```
library(nnet)
reg = multinom(Species ~., data = iris)
predict(reg, newdata = data.frame(Sepal.Length = 5.1, Sepal.Width = 3.5,
Petal.Length = 1.6, Petal.Width = 0.2), type = "class")
```

Cela renvoie :

```
[1] setosa
Levels: setosa versicolor virginica
```

Ainsi, on obtient $y_* =$ `setosa`. Précisons que le modèle sous-jacent est tout de même plus sophistiqué que l'arbre de classification (ce qui ne veut pas dire qu'il est plus performant).

On peut voir le nombre de fois où le modèle de régression multinomial se trompe avec les données de bases sous la forme d'un tableau. On fait :

```
tab2 = table(predict(reg), iris$Species)
tab2
```

Cela renvoie :

```
            setosa versicolor virginica
setosa          50          0         0
versicolor       0         49         1
virginica        0          1        49
```

On constate alors que l'arbre de classification est performant, avec un taux d'erreur global très faible, i.e., $t = (1+1)/(50+49+49+1+1) = 0.02$.

Ce taux est meilleur que celui obtenu avec l'arbre de classification. En revanche, le modèle de régression multinomial est sophistiqué et difficile à interpréter. Dans ce sens, il est plus opaque que l'arbre de classification et donc plus difficile à communiquer, pour un résultat presque équivalent.

Exemple 2 : On considère le jeu de données `spam7` de la librairie `DAAG`. D'abord, on charge ce jeu de données :

```
library(DAAG)
str(spam7)
```

Cela renvoie :

```
'data.frame':   4601 obs. of  7 variables:
```

```
$ crl.tot: num  278 1028 2259 191 191 ...
$ dollar : num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
$ bang   : num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
$ money  : num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
$ n000   : num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
$ make   : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
$ yesno  : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 2 2 2 2 ...
```

Ainsi, il y a 4601 observations de plusieurs caractères, dont `yesno` qui est un caractère qualitatif à 2 modalités (`y` ou `n`). Ce caractère nous indique si un un email est un spam ou pas, en fonction de plusieurs caractères quantitatifs : `crl.tot`, `dollar`, `bang`, `money`, `n000` et `make`. Ces caractères sont définis comme suit :

- `crl.tot` : longueur totale des mots en majuscules,
- `dollar` : nombre d'occurrences du symbole "$",
- `bang` : nombre d'occurrences du symbole "!",
- `money` : nombre d'occurrences du mot "money",
- `n000` : nombre d'occurrences de la suite de chiffres "000",
- `make` : nombre d'occurrences du mot "make".

Dans un premier temps, on peut voir le nombre de emails considérés comme des spams dans le jeu de données en faisant :

```
table(spam7$yesno)
```

Cela renvoie :

```
   n    y
2788 1813
```

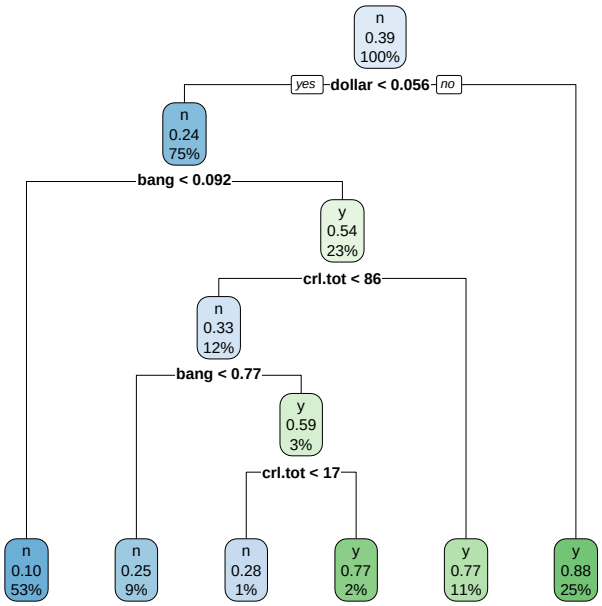
Ainsi, il y a 1813 spams parmi les 4601 emails. L'objectif est de savoir si un email dont on connait les valeurs de `crl.tot`, `dollar`, `bang`, `money`, `n000` et `make` peut être considéré comme

étant un spam ou pas. On peut donc utiliser un arbre de classification de type CART.

On fait :

```
library(rpart)

arbre = rpart(yesno ~ ., data = spam7)

library(rpart.plot)

rpart.plot(arbre)
```

Cela renvoie :



On obtient un arbre de classification peu complexe et communicable.

Ainsi, par exemple, pour savoir si un email vérifiant `crl.tot` $= 565$, `dollar` $= 0.156$, `bang` $= 0.452$, `money` $= 0.51$, `n000` $= 1.02$ et `make` $= 0.32$ est probablement un spam, on fait :

```
predict(arbre, newdata = data.frame(crl.tot = 565, dollar = 0.156, bang =

0.452, money = 0.51, n000 = 1.02, make = 0.32), type = "class")
```

Cela renvoie :

```
1

y

Levels: n y
```

Ainsi, une valeur plausible y_* de `yesno` pour l'email concerné est $y_* =$ `y`. Donc le email reçu est probablement un spam.

Pour aller un peu plus loin, on peut s'intéresser au taux d'erreur global de l'arbre de classification. On fait :

```
pred = predict(arbre, newdata = spam7, type = "class")
mc = table(spam7$yesno, pred)
t = (mc[1, 2] + mc[2, 1]) / sum(mc)
t
```

Cela renvoie :

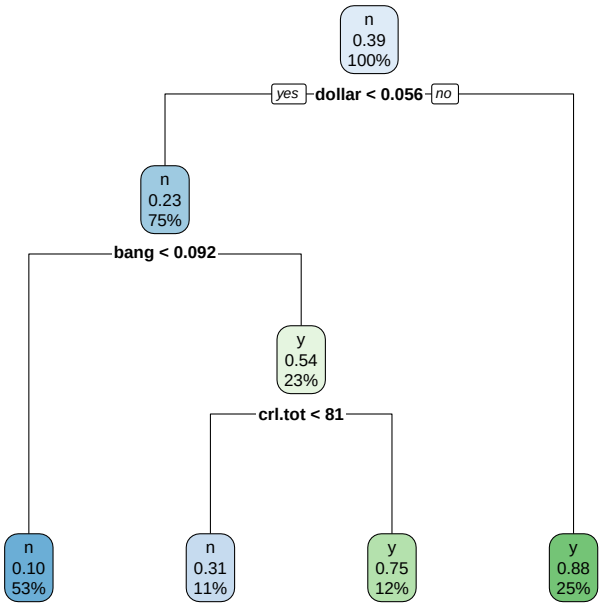
```
[1] 0.1380135
```

Ainsi, le taux d'erreur global est de 0.1380135. Donc, pour les données considérées, l'arbre s'est trompé une fois sur 10 environ, ce qui est très acceptable.

Afin d'avoir un indicateur d'erreur plus solide, on peut appliquer la méthode apprentissage-test. Ainsi, on propose de sélectionner 101 emails au hasard dans le jeu de données (pour qu'il en reste le chiffre rond de 4500), ce qui constituera le jeu de données de test. Puis, on utilise les emails non sélectionnés pour construire le jeu de données d'apprentissage avec lequel on va construire un nouvel arbre de classification, dont on calculera te taux d'erreur apprentissage-classification. On fait :

```
n = nrow(spam7)
m = 4500
s = sample(1:n, m)
apprentissage = spam7[s, ]
test = spam7[-s, ]
arbre_new = rpart(yesno ~ ., data = apprentissage)
rpart.plot(arbre_new)
```

Cela renvoie :



On obtient un arbre de régression légèrement différent du premier.

Ensuite, on calcule le taux d'erreur apprentissage-test en faisant :

```
c = predict(arbre_new, test[1:6], type = "class")

sum(c != test[ ,7]) / (n - m)
```

Cela renvoie :

```
[1] 0.1287129
```

On obtient un taux d'erreur sensiblement inférieur taux d'erreur global.

3 Forêts d'arbres de décision

Dans ce chapitre, la notion d'arbre de décision (régression et classification) est supposée connue.

3.1 Forêts d'arbres de régression

Introduction : Le contexte et l'objectif d'une forêt d'arbres de régression sont les mêmes que pour un arbre de régression. On les rappelle ci-après. Pour n individus $\omega_1, \ldots, \omega_n$ d'une population, on dispose des valeurs de $p+1$ caractères $X_1, \ldots, X_p, Y$, constituant ainsi les données. Ici, on suppose que Y est quantitative. Partant des données, l'objectif est de donner une valeur plausible de Y pour un individu dont on connaît les valeurs de $X_1, \ldots, X_p$.

Aide à la décision : Pour décider d'une valeur plausible de Y, on peut s'aider d'une forêt d'arbres de régression.

Forêt d'arbres de régression : première approche Comme son nom l'indique, une forêt d'arbres de régression est un ensemble d'arbres de régression. Par rapport à un arbre seul de régression, une telle forêt vise à avoir un taux d'erreur global moindre et à traiter plus efficacement de grands jeux de données.

Forêt aléatoire d'arbres de régression : Parmi les différents types de forêts d'arbres de régression, la forêt aléatoire est celle qui est la plus utilisée. On la décrit ci-après. D'abord, précisons que 3 entiers vont être mis en jeu, notés N, m et q, avec $m \in \{1, \ldots, n\}$ et $q \in \{1, \ldots, p\}$. L'entier N désigne le nombre d'arbres de régression dans la forêt. Pour k allant de 1 à N, le k-ème arbre est construit de la manière suivante :

- on sélectionne au hasard et avec remise m individus parmi $\omega_1, \ldots, \omega_n$, l'arbre sera alors construit exclusivement avec les individus sélectionnés et les données associées,
- pour chaque nœud de l'arbre, on sélectionne sans remise q caractères parmi $X_1, \ldots, X_p$, avec q généralement petit pour que l'arbre ne soit pas grand. Dès lors, un caractère de coupure sera déterminé parmi les caractères sélectionnés et une valeur seuil sera calculée en fonction.

On obtient alors N arbres partiellement indépendants. Une fois la forêt construite, on peut alors passer à l'étape de prédiction.

Pour donner un ordre de grandeur, des choix raisonnables pour N, m et q sont : $N = 500$, $m =$ environ 2 tiers du nombre d'individus et $q =$ environ $\sqrt{p}$.

Commandes R : On utilise `randomForest` de la librairie `randomForest`, avec la même syntaxe de base que `rpart`.

Utilisation : Partant d'une forêt d'arbres de régression, une valeur plausible de Y pour un individu dont on connaît les valeurs $X_1, \ldots, X_p$ est donnée par la moyenne des valeurs plausibles de Y pour cet individu provenant des arbres de la forêt. Précisons qu'il n'y a pas moyen de visualiser la forêt comme on le ferait pour un arbre unique.

Commandes R : On utilise `predict`.

Remarque : Bagging : Pour la culture, une méthode appelée Bagging (fusion de bootstrap aggregating) à inspiré les forêts aléatoires ; elle reprend les mêmes étapes de construction, mais considère tous les caractères $X_1, \ldots, X_p$ pour déterminer les conditions de coupure (il n'y a pas de sélection sans remise de ces caractères). En règle générale, elle est moins performante qu'une forêt aléatoire (cela dépend aussi de la configuration adoptée).

Commandes R : On utilise `randomForest` avec l'option `mtry` $= p$ (où p désigne le nombre de caractères $X_1, \ldots, X_p$).

3.2 Forêts d'arbres de classification

Introduction : Le contexte et l'objectif d'une forêt d'arbres de décision sont les mêmes que pour un arbre de classification. On le rappelle ci-après. Pour n individus $\omega_1, \ldots, \omega_n$ d'une population, on dispose des valeurs de $p+1$ caractères $X_1, \ldots, X_p, Y$, constituant ainsi les données.

Ici, on suppose que Y est qualitative. Partant des données, l'objectif est de donner une modalité plausible de Y pour un individu dont on connaît les valeurs de $X_1, \ldots, X_p$.

Aide à la décision : Pour décider d'une modalité plausible de Y, on peut s'aider d'une forêt d'arbres de classification.

Forêt aléatoire d'arbres de classification : Une forêt aléatoire d'arbres de classification est définie exactement comme une forêt aléatoire d'arbres de régression ; seule la nature de Y change, ce qui n'influe aucunement dans le processus de construction de l'arbre.

Commandes R : On utilise `randomForest` de la librairie `randomForest`.

Utilisation : Une modalité plausible de Y pour un individu dont on connaît les valeurs $X_1, \ldots, X_p$ est donnée par celle qui revient le plus parmi les modalités plausibles de Y pour cet individu provenant des arbres de la forêt.

Taux d'erreur Out-Of-Bag : Le taux d'erreur Out-Of-Bag, noté OOB, est une estimation du taux d'erreur théorique. Plus ce taux est petit, plus la qualité prédictive de la forêt est bonne.

Commandes R : On utilise `foret$err.rate[1]`.

Remarques :

- On peut calculer le taux d'erreur global, exactement comme pour un arbre de classification.
- La methode Bagging existe aussi dans ce cadre.

3.3 Mise en œuvre avec R

Exemple 1 : On utilise le jeu de données `data-arbre`, déjà analysé dans un exemple précédent par un arbre de régression. L'objectif est d'expliquer un caractère quantitatif Y en fonction de 2 caractères quantitatifs $x1$ et $x2$. D'abord, on charge les données :

```
w = read.csv("https://chesneau.users.lmno.cnrs.fr/data-arbre.csv", header =
T, sep = ";")
attach(w)
```

Pour atteindre l'objectif, comme Y est quantitative, on peut utiliser une forêt aléatoire d'arbres de régression. La commande principale est `randomForest` de la librairie `randomForest`. On fait :

```
library(randomForest)
foret = randomForest(Y ~ x1 + x2)
```

Par exemple, pour donner une valeur plausible y_* de Y d'un individu vérifiant $x1 = 3$ et $x2 = 1$, on peut utiliser la commande `predict` :

```
predict(foret, newdata = data.frame(x1 = 3, x2 = 1))
```

Cela renvoie :

```
       1
2.601674
```

D'où $y_* = 2.601674$. Avec un arbre de régression, on avait obtenu $y_* = 1.7$, on constate donc une différence. Les forêts aléatoires présentant des garanties de précision, à choisir, on prendra plutôt la première valeur.

Exemple 2 : On considère le jeu de données `spam7` de la librairie `DAAG`. On rappelle qu'il y a 8 caractères, dont `yesno` qui est un caractère qualitatif à 2 modalités (`y` ou `n`). Ce caractère nous indique si un email est un spam ou pas en fonction de plusieurs caractères quantitatifs : `crl.tot`, `dollar`, `bang`, `money`, `n000` et `make`. On souhaite expliquer `yesno` en fonction des autres caractères. Pour ce faire, on utilise une forêt aléatoire d'arbres de classification. On fait :

```
library(DAAG)
str(spam7)
library(randomForest)
foret = randomForest(yesno ~ ., data = spam7)
```

On peut alors faire de la prédiction en utilisant cette forêt. Par exemple, pour savoir si un email vérifiant `crl.tot` $= 565$, `dollar` $= 0.156$, `bang` $= 0.452$, `money` $= 0.51$, `n000` $= 1.02$ et `make` $= 0.32$ est probablement un spam, on fait :

```
predict(foret, newdata = data.frame(crl.tot = 565, dollar = 0.156, bang =
0.452, money = 0.51, n000 = 1.02, make = 0.32), type = "class")
```

Cela renvoie :

```
1
y
Levels: n y
```

Ainsi, une valeur plausible y_* de `yesno` pour l'email concerné est $y_* =$ `y`. Donc le email reçu est probablement un spam.

Pour aller un peu plus loin, on peut s'intéresser au taux d'erreur Out-Of-Bag. On fait :

```
foret$err.rate[1]
```

Cela renvoie :

```
[1] 0.1402688
```

Ainsi, le taux d'erreur Out-Of-Bag est de $oob = 0.1402688$. Comme ce taux est petit, on peut admettre que la qualité prédictive de la forêt est bonne.

Pour confirmer cela, on peut également déterminer le taux d'erreur global de la forêt de classification. On fait :

```
pred = predict(foret, newdata = spam7, type = "class")
mc = table(spam7$yesno, pred)
t = (mc[1, 2] + mc[2, 1]) / sum(mc)
t
```

Cela renvoie :

```
[1] 0.07541839
```

Ainsi, le taux d'erreur global de la foret aléatoire est de 0.07541839. On avait obtenu l'erreur globale pour un arbre unique de classification 0.1380135. On a dont divisé par 2 l'erreur, et par conséquent, multiplié les chances par 2 de faire le bon choix.

À titre de curiosité, on peut voir le taux d'erreur que donnerait la méthode du Bagging. On fait :

```
foret2 = randomForest(yesno ~ ., mtry = 6, data = spam7)
pred = predict(foret2, newdata = spam7, type = "class")
mc = table(spam7$yesno, pred)
t = (mc[1, 2] + mc[2, 1]) / sum(mc)
t
```

Cela renvoie :

```
[1] 0.02890676
```

Ainsi, le taux d'erreur global obtenu est de 0.07541839. Dans cet exemple, avec ce critère, la méthode du Bagging est donc plus performante que la forêt aléatoire.

4 Pour aller plus loin

Dans ce chapitre, on discute succinctement d'autres outils modernes aidant à la décision, tous reposant sur la notion d'arbre de décision.

Autre algorithmes : Dans ce qui précède, on s'est focalisé sur les arbres reposant sur l'algorithme CART. D'autres algorithmes peuvent être utilisés, parmi lesquels :

- ID3 (acronyme pour Iterative Dichotomiser 3),
- CHAID (acronyme pour CHi-squared Automatic Interaction Detector),
- QUEST (acronyme pour Quick, Unbiased, Efficient, claSsification Tree),
- MARS (acronyme pour Multivariate Adaptive Regression Splines),
- Conditional Inference Trees.

Commandes R : On utilise `ctree` de la librairie `party`.

Boosting : La méthode appelée Boosting consiste à créer une forêt composée d'arbres dont les constructions se suivent de manière séquentielle : chaque arbre est construit en utilisant les informations d'arbres précédemment développés. C'est une méthode très performante adapté aux grands jeux de données.

Commandes R : On utilise `gmb` de la librairie `gmb`.

5 Exercices

Exercice 1. On souhaite expliquer un caractère quantitatif Y en fonction de 2 caractères quantitatifs X_1 et X_2. On dispose des observations de ces 3 caractères pour 10 individus $\omega_1, \ldots, \omega_{10}$. En particulier, on a les valeurs de Y suivantes :

ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9	ω_{10}
1.03	12.35	3.76	3.61	2.72	2.79	6.04	4.64	10.37	8.20

Grâce aux données, on construit un arbre de régression de type CART (avec différentes conditions de coupures portant sur X_1 et X_2). On souhaite donner une valeur plausible pour un individu vérifiant $X_1 = 12$ et $X_2 = 3$. Dès lors, le cheminement de l'arbre nous amène à une feuille associée aux individus ω_3, ω_5, ω_9 et ω_{10}. Donner alors une valeur plausible de Y.

Exercice 2. On souhaite expliquer un caractère quantitatif Y en fonction de 2 caractères quantitatifs X_1 et X_2. On dispose des observations de ces 3 caractères pour 7 individus et on précise que celles-ci sont toutes différentes. On utilise alors un arbre de régression. Pour effectuer la coupure de la racine, on dispose des valeurs de toutes les erreurs globales ; elles sont données par :

$$E(1,6) = 1.2, \quad E(1,8) = 3.1, \quad E(1,10) = 1.7, \quad E(1,14) = 2.4, \quad E(1,18) = 2.1, \quad E(1,26) = 1.9$$

et

$$E(2,7) = 2.3, \quad E(2,9) = 3.3, \quad E(2,13) = 1.1, \quad E(2,17) = 2.1, \quad E(2,20) = 1.5, \quad E(2,28) = 3.8.$$

1. Donner les valeurs observées de X_1 et X_2 avec la même méthode que celle utilisée par `rpart`, sachant que X_1 prend la valeur 5 et X_2 prend la valeur 6.
2. Donner le caractère de coupure et la valeur seuil associée.

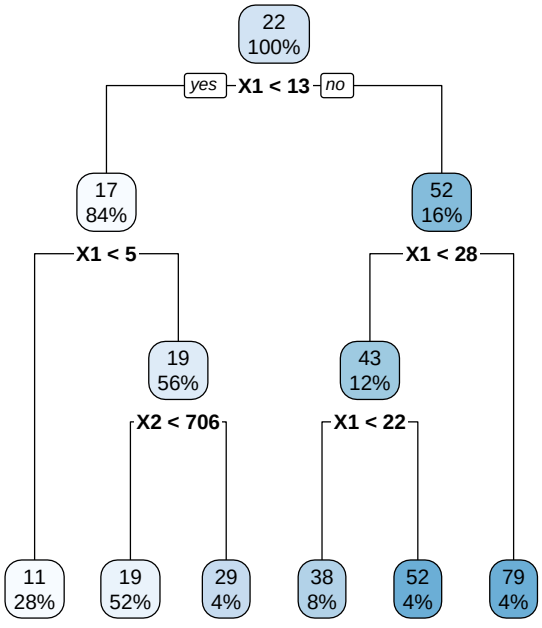
3. Sachant que $SS = 4.2$, donner l'indice d'amélioration associé à cette première coupure.

Exercice 3. Le responsable de la maintenance d'un réseau de distributeurs de boissons aimerait prédire le temps nécessaire pour l'approvisionner (caractère Y, unité en minutes). Le temps d'approvisionnement dépend du nombre de caisses à charger (caractère $X1$) et de la distance parcourue par l'employé pour approvisionner l'ensemble des machines (caractère $X2$, unité en mètres). Les données recueillies sont disponibles ici :

```
w = read.table("https://chesneau.users.lmno.cnrs.fr/boisson.txt", header =
T)
attach(w)
```

Comme Y est quantitative, on propose d'utiliser un arbre de régression.

1. Partant de `w`, écrire les commandes R permettant d'afficher l'arbre de régression avec des nœuds que l'on coupera uniquement si 3 ou plus de 3 individus jours sont présents.
2. On obtient l'arbre de régression de type CART suivant :



(a) Combien de nœuds a cet arbre (hors racine et feuille) ? Combien de feuilles a cet arbre ? Quelle est le premier caractère de coupure ? Quelle est la première valeur seuil associée ? Combien y a t-il eu de conditions de coupure ?

(b) En utilisant leurs définitions, calculer avec R l'erreur globale définie par $E(1, 13)$, ainsi que l'amélioration associée (on pourra utiliser les vecteurs `Y[X1 >= 13]` et `Y[X1 < 13]`). Retrouver l'amélioration avec des commandes R.

(c) En parcourant cet arbre, donc sans utiliser de commandes R, donner une valeur plausible du temps en minutes que met l'employé pour approvisionner le responsable avec 6 caisses à charger et 128 mètres à parcourir.

(d) Écrire des commandes R permettant d'obtenir la valeur plausible de Y déterminée précédemment.

(e) Comparer la valeur précédente avec une valeur prédite que l'on obtiendrait avec le modèle de régression linéaire multiple, i.e.,

$$Y = \beta_0 + \beta_1 X1 + \beta_2 X2 + \epsilon,$$

où β_0, β_1 et β_2 sont des coefficients inconnus et ϵ est une variable d'erreur.

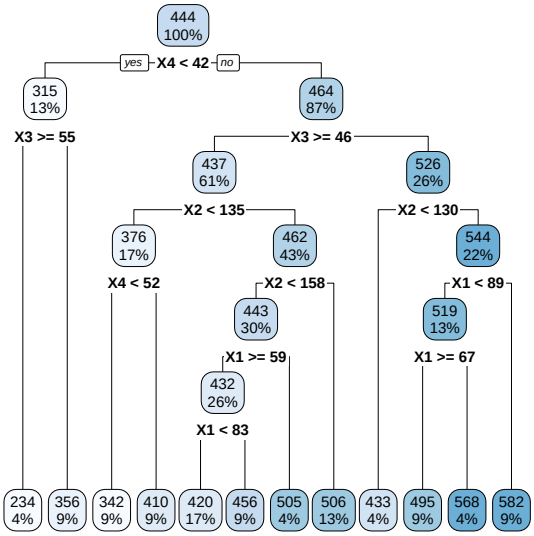
Exercice 4. Dans une étude statistique, 23 professeurs sont évalués quant à la qualité de leur enseignement. Pour chacun d'entre eux, on dispose d'un indice de performance globale donné par les étudiants (caractère Y), des résultats de 4 tests écrits donnés à chaque professeur (caractères $X1$, $X2$, $X3$ et $X4$) et du sexe (caractère $X5$, avec $X5 = 0$ pour femme, $X5 = 1$ pour homme). Les données sont disponibles ici :

```
w = read.table("https://chesneau.users.lmno.cnrs.fr/profs.txt", header = T)
attach(w)
```

On souhaite donner une valeur plausible de Y pour un individu dont on connaît les valeurs de $X1$, $X2$, $X3$, $X4$ et $X5$. Comme Y est quantitative, on propose d'utiliser un arbre de régression.

1. Partant de `w`, écrire les commandes R permettant d'afficher l'arbre de régression, noté `arbre`, avec des nœuds que l'on coupera uniquement si 3 ou plus de 3 individus jours sont présents.

2. On obtient l'arbre de régression de type CART suivant :

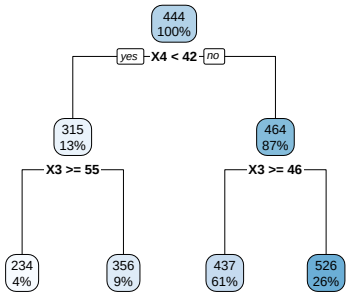


Celui-ci est un peu complexe. Dans la suite, on vise à le simplifier.

(a) Dans un premier temps, on tente une approche brutale. Expliquer l'enjeu et la sortie des commandes suivantes :

```
arbre2 = rpart(Y ~ ., minsplit = 3, maxdepth = 2, data = w)

rpart.plot(arbre2)
```

Cela renvoie :



(b) Dans un deuxième temps, on tente un élagage en utilisant le paramètre de complexité. On fait :

```
printcp(arbre)
```

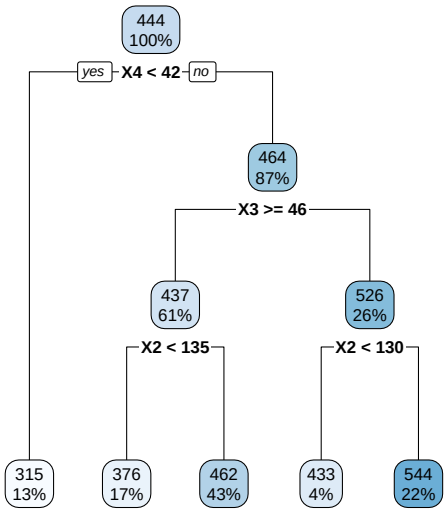
Cela renvoie, entre autre :

```
         CP nsplit rel error  xerror    xstd
1  0.352925      0  1.000000 1.12730 0.32486
2  0.201819      1  0.647075 1.27109 0.34755
3  0.129935      2  0.445256 1.09069 0.25340
4  0.063288      3  0.315321 0.84720 0.16514
5  0.060943      4  0.252033 0.80219 0.16290
6  0.052202      5  0.191090 0.82847 0.16507
7  0.028483      6  0.138888 0.82664 0.14947
8  0.027983      7  0.110405 0.75513 0.11950
9  0.027798      8  0.082421 0.75513 0.11950
10 0.021820      9  0.054623 0.79527 0.11444
11 0.010320     10  0.032803 0.85652 0.14643
12 0.010000     11  0.022483 0.85652 0.14643
```

(Cette sortie diffère d'une exécution à une autre, les xerrors faisant intervenir de l'aléatoire dans leur construction).

Dès lors, proposer un élagage judicieux de l'arbre initial, ainsi que les commandes associées.

Pour information, on aboutit à l'arbre suivant :



(c) En parcourant l'arbre précédent, donc sans utiliser de commandes R, donner une valeur plausible de Y pour $X1 = 445$ et $X2 = 141$, $X3 = 52$, $X4 = 45$ et $X5 = 1$.

Exercice 5. Décrire l'enjeu et expliquer la succession logique des commandes R suivantes :

```
tennis = read.table("https://chesneau.users.lmno.cnrs.fr/tennis.txt", header
= T)
attach(tennis)
library(rpart); library(rpart.plot)
arbre = rpart(Jouer ~ ., data = tennis)
arbre
rpart.plot(arbre)
options = rpart.control(minsplit = 6)
arbre2 = rpart(Jouer ~ ., data = tennis, control = options)
rpart.plot(arbre2)
predict(arbre2, tennis)
pred = predict(arbre2, tennis, type = "class")
sum(pred != tennis$Jouer) / 14
```

Exercice 6. On souhaite expliquer un caractère qualitatif Y à trois modalités : A, B et C, en fonction de 2 caractères quantitatifs X_1 et X_2. On dispose des observations de ces 3 caractères pour 10 individus $\omega_1, \ldots, \omega_{10}$. En particulier, on a les valeurs de Y suivantes :

ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9	ω_{10}
A	B	A	C	A	C	A	B	C	A

Grâce aux données, on construit un arbre de classification de type CART. On souhaite donner une valeur plausible pour un individu vérifiant $X_1 = 2$ et $X_2 = 3.6$. Dès lors, le cheminement de l'arbre nous amène à une feuille associée aux individus ω_3, ω_5, ω_9 et ω_{10}. Donner alors une modalité plausible de Y.

Exercice 7. On souhaite expliquer un caractère qualitatif Y à trois modalités : A, B et C, en fonction de 2 caractères quantitatifs X_1 et X_2. On dispose des observations de ces 3 caractères pour 7 individus $\omega_1, \ldots, \omega_7$ et on précise que celles-ci sont toutes différentes. On utilise alors un arbre de régression. Pour effectuer la coupure de la racine, on dispose des valeurs de toutes les pertes globales d'information ; elles sont données par :

$$G(1,6) = 3.1, \quad G(1,8) = 3.1, \quad G(1,10) = 1.2, \quad G(1,14) = 2.8, \quad G(1,18) = 1.3, \quad G(1,26) = 1.9$$

et

$$G(2,7) = 2.3, \quad G(2,9) = 3.3, \quad G(2,13) = 1.3, \quad G(2,17) = 0.8, \quad G(2,20) = 1.9, \quad G(2,28) = 2.8.$$

1. Donner le caractère de coupure et la valeur seuil associée.
2. Sachant que $G = 4.5$, donner l'indice d'amélioration associé à cette première coupure.

Exercice 8. Lors d'une étude, on souhaite expliquer le fait qu'un individu joue au tennis en fonction des divers conditions climatiques. Ainsi, on considère les caractères qualitatifs suivants : Y : Jouer (Oui ou Non), $X1$: Ensoleillement (Soleil ou Couvert), $X2$: Température (Chaud ou Frais), $X3$: Humidité

(Humide ou Sec) et $X4$: Vent (Faible ou Fort). Les données, collectées pour 14 jours, sont présentées dans le tableau suivant :

Jour	$X1$	$X2$	$X3$	$X4$	Y
1	Soleil	Chaud	Humide	Faible	Non
2	Soleil	Chaud	Humide	Fort	Non
3	Couvert	Chaud	Humide	Faible	Oui
4	Couvert	Chaud	Humide	Faible	Oui
5	Couvert	Frais	Sec	Faible	Oui
6	Couvert	Frais	Sec	Fort	Non
7	Couvert	Frais	Sec	Fort	Oui
8	Soleil	Chaud	Humide	Faible	Non
9	Soleil	Frais	Sec	Faible	Oui
10	Couvert	Chaud	Sec	Faible	Oui
11	Soleil	Chaud	Sec	Fort	Oui
12	Couvert	Chaud	Humide	Fort	Oui
13	Couvert	Chaud	Humide	Fort	Non
14	Couvert	Chaud	Sec	Faible	Oui

1. Coder dans R le tableau de données en adoptant les codes binaires suivants : (Soleil = 1, Couvert = 0), (Chaud = 1, Frais = 0), (Humide = 1, Sec = 0), (Faible = 0, Fort = 1) et (Non = 0, Oui = 1).

2. On propose alors de faire un arbre de classification avec des nœuds que l'on coupera uniquement si 4 ou plus de 4 individus (jours) sont présents. On obtient alors à la sortie numérique suivante :

```
n= 14


node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

```
1) root 14 5 1 (0.3571429 0.6428571)
  2) X3=1 7 3 0 (0.5714286 0.4285714)
    4) X1=1 3 0 0 (1.0000000 0.0000000) *
    5) X1=0 4 1 1 (0.2500000 0.7500000) *
  3) X3=0 7 1 1 (0.1428571 0.8571429) *
```

(a) À partir de cette sortie, représenter graphiquement l'arbre de classification.

(b) Donner les commandes qui ont permis d'obtenir cette sortie.

(c) Donner les commandes qui permettrait de représenter graphiquement l'arbre de classification.

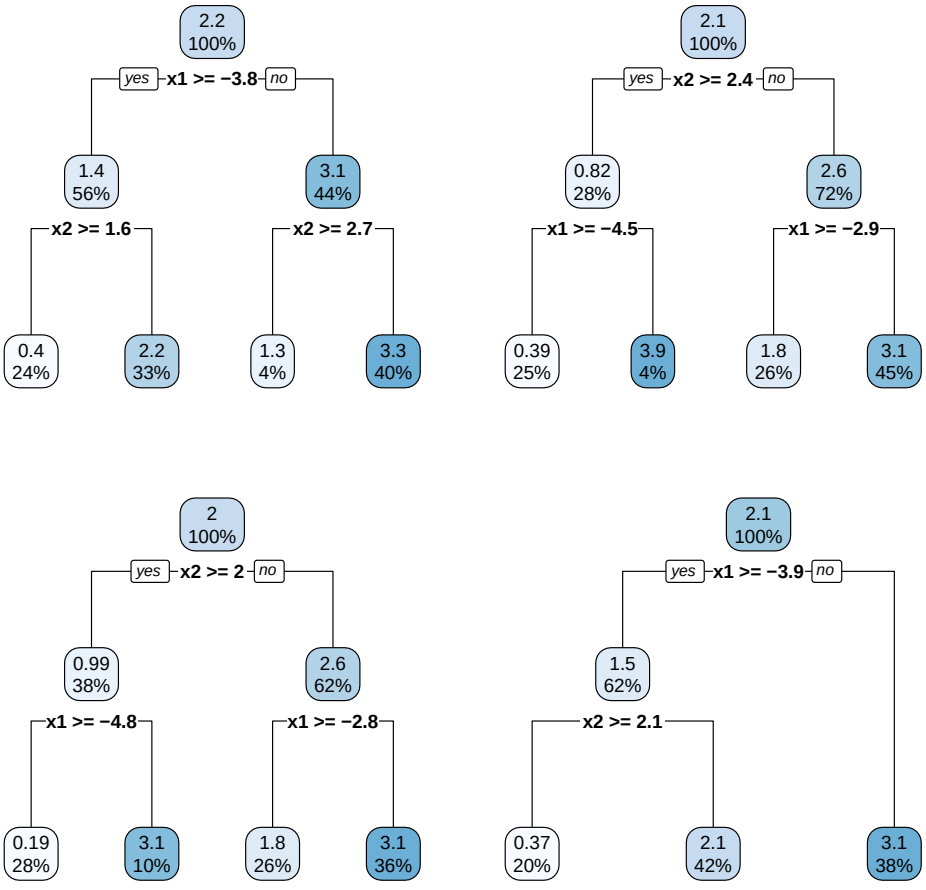
3. Retrouver le fait que le caractère de coupure de la racine est $X3$ à l'aide de l'indice moyen de Gini et la fonction R `GiniMoy` suivante :

```
GiniMoy = function(X,Y)
{
  n = length(X)
  ind0 = which(X == 0)
  n0 = length(ind0)
  nY0 = sum(Y[ind0] == 0)
  nY1 = n0 - nY0
  I0 = 1 - (nY0 / n0)^2 - (nY1 / n0)^2
  ind1 = which(X == 1)
  n1 = length(ind1)
  nY0 = sum(Y[ind1] == 0)
  nY1 = n1 - nY0
  I1 = 1 - (nY0 / n1)^2 - (nY1 / n1)^2
  C = (n0 / n) * I0 + (n1 / n) * I1
  return(C)
}
```

4. Au jour 15, les conditions climatiques observées sont les suivantes : $X1$ = Soleil, $X2$ = Frais, $X3$ = Sec et $X4$ = Fort. Dès lors, donner une valeur plausible de Y à partir de l'arbre de classification.

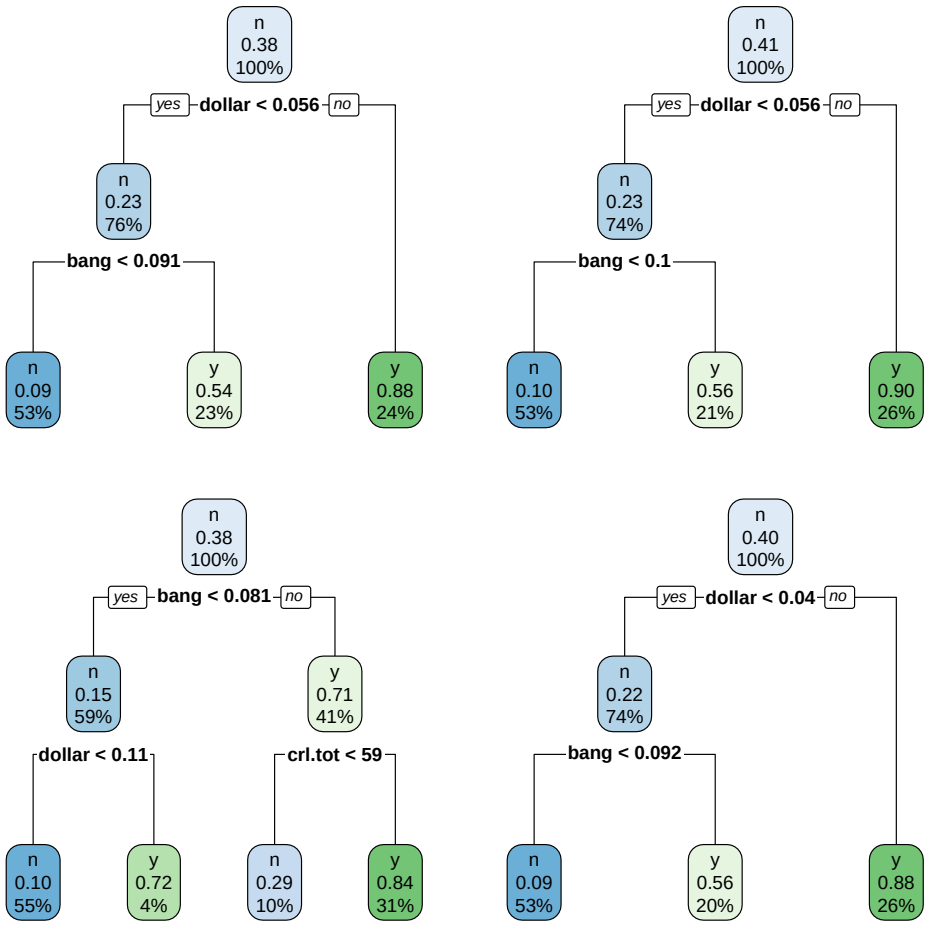
5. Déterminer le taux d'erreur global de l'arbre de classification. Qu'en pensez-vous ?

Exercice 9. On souhaite expliquer un caractère quantitatif Y en fonction de 2 caractères quantitatifs $x1$ et $x2$. On dispose des observations de ces 3 caractères pour 100 individus. Pour ce faire, on utilise une forêt aléatoire d'arbres de régression avec 4 arbres (pour s'amuser, en pratique, on la construit plutôt avec 500 arbres...). On obtient les 4 arbres de régression suivants :



À partir de cette forêt aléatoire, donner une valeur plausible de Y pour un individu vérifiant $x1 = 1.8$ et $x2 = 2.3$.

Exercice 10. On considère le jeu de données `spam7`, avec lequel on connait la nature spam ou non d'un email en fonction de certaines de ses caractéristiques, notamment des caractères `bang`, `dollar` et `crl.tot`. Le caractère que l'on veut expliquer est `yesno`, lequel est qualitatif de modalités `y` et `n`. Pour ce faire, on utilise une forêt aléatoire d'arbres de classification avec 4 arbres (pour s'amuser, en pratique, on la construit plutôt avec 500 arbres...). On obtient les 4 arbres de classification suivants :



À partir de cette forêt aléatoire, donner une modalité plausible de `yesno` pour un individu vérifiant `bang` $= 0.096$, `dollar` $= 0.045$ et `crl.tot` $= 55$.

6 Solutions

Solution 1. Par le principe d'un arbre de régression (de type CART), pour un individu vérifiant $X_1 = 12$ et $X_2 = 3$, une valeur plausible de Y est donnée par la moyenne des valeurs de Y pour les individus ω_3, ω_5, ω_9 et ω_{10}. Cette valeur est donc :

$$\overline{y}_* = \frac{1}{4}(3.76 + 2.72 + 10.37 + 8.20) = 6.2625.$$

Exercice 2.

1. La commande `rpart` utilise une grille de valeurs données par les moyennes de deux observations consécutives d'un caractère pour définir les valeurs seuils (notées c). Dès lors, on raisonne comme suit, dans l'ordre :
   - i d'une part, on sait que X_1 prend la valeur 5, et le fait que l'on considère $E(1, 6)$ signifie que la première valeur seuil est 6, ce qui implique que la deuxième valeur de X_1 est 7 (6 étant la moyenne entre 5 et 7),
   - ii puis, comme l'on considère $E(1, 8)$ signifie que la valeur seuil suivante est 8, et par conséquent, la troisième valeur de X_1 est 9 (8 étant la moyenne entre 7 et 9),
   - iii puis vient $E(1, 10)$ ce qui signifie que la valeur seuil suivante est 10, et par conséquent, la quatrième valeur de X_1 est 11 (10 étant la moyenne entre 9 et 11),
   - iv puis vient $E(1, 14)$ signifie que la valeur seuil suivante est 14, et par conséquent, la cinquième valeur de X_1 est 17 (14 étant la moyenne entre 11 et 17),
   - v puis vient $E(1, 18)$ signifie que la valeur seuil suivante est 18, et par conséquent, la sixième valeur de X_1 est 19 (18 étant la moyenne entre 17 et 19),
   - vi puis vient $E(1, 26)$ signifie que la valeur seuil suivante est 26, et par conséquent, la septième valeur de X_1 est 33 (26 étant la moyenne entre 18 et 26).

   Ainsi, les valeurs de X_1 sont : 5, 7, 9, 11, 17, 19, et 33.

   En procédant de même pour X_2, on obtient les valeurs : 6, 8, 10, 16, 18, 22, et 34.

2. La plus petite erreur correspond à $E(2, 13)$, avec la valeur 1.1. Par conséquent, le caractère de coupure est X_2 et la valeur seuil associée est 13.

3. Comme $SS = 4.2$, l'indice d'amélioration associé à cette première coupure est

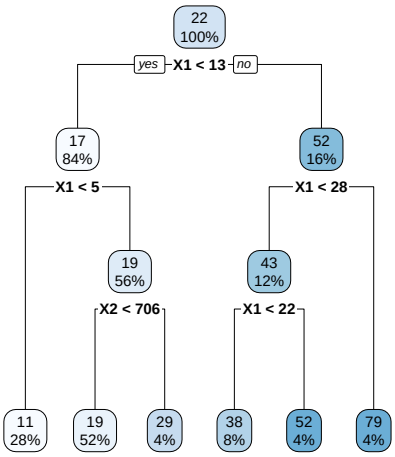
$$I = 1 - \frac{E(2, 13)}{SS} = 1 - \frac{1.1}{4.2} = 0.7380952.$$

Solution 3.

1. Partant de `w`, les commandes R permettant d'afficher l'arbre de régression avec des nœuds que l'on coupera uniquement si 3 ou plus de 3 individus jours sont présents sont les suivantes :

```
w = read.table("https://chesneau.users.lmno.cnrs.fr/boisson.txt", header =
T)
attach(w)
library(rpart)
arbre = rpart(Y ~ X1 + X2, minsplit = 3)
library(rpart.plot)
rpart.plot(arbre)
```

2. On obtient l'arbre de régression de type CART suivant :



(a) Cet arbre a 4 nœuds (hors racine et feuille). Il a 6 feuilles. Le premier caractère de coupure est $X1$. La première valeur seuil associée est 13. Il y a eu 5 conditions de coupure.

(b) Pour obtenir l'erreur globale définie par $E(1, 13)$, ainsi que l'amélioration associée, on peut faire :

```
E = sum((Y[X1>= 13] - mean(Y[X1>= 13]))^2) + sum((Y[X1< 13] - mean(Y[X1<
13]))^2)
E
```

Cela renvoie : `[1] 1678.756`

Ainsi, on a $E = 1678.756$.

Pour calculer l'indice d'amélioration, on peut faire :

```
SS = sum((Y - mean(Y))^2)
1- E / SS
```

Cela renvoie : `[1] 0.7097858`

Ainsi, l'indice d'amélioration est $I = 0.7097858$.

On peut retrouver ces informations avec les commandes suivantes :

```
summary(arbre)$splits[1,3]
```

(c) En parcourant l'arbre obtenu, on peut supposer que l'employé mettra 19 minutes pour approvisionner s'il y a 6 caisses à charger et 128 mètres à parcourir.

(d) Les commandes R permettant d'obtenir la valeur plausible de Y déterminée précédemment sont :

```
predict(arbre, newdata=data.frame(X1 = 6, X2 = 128))
```

Cela renvoie :

```
       1
18.72692
```

On obtient donc la même chose, à l'arrondi près.

(e) Pour avoir une valeur prédite que l'on obtiendrait avec le modèle de régression linéaire multiple, i.e.,

$$Y = \beta_0 + \beta_1 X1 + \beta_2 X2 + \epsilon,$$

où β_0, β_1 et β_2 sont des coefficients inconnus et ϵ est une variable d'erreur, on peut faire :

```
reg = lm(Y ~ X1 + X2)
predict(reg, newdata=data.frame(X1 = 6, X2 = 128))
```

Cela renvoie :

```
       1
13.87793
```

On constate que le résultat obtenu est relativement éloigné de celui obtenu avec l'arbre. Cet éloignement peut être amoindri avec le choix d'autres critère d'arrêts pour l'arbre, entre autre. On pourrait aussi penser q'une prédiction intermédiaire serait judicieuse, à savoir :

$$\overline{y}_* = \frac{1}{2}(18.72692 + 13.87793) = 16.30242.$$

Solution 4.

1. Partant de `w`, les commandes R permettant d'afficher l'arbre de régression, noté `arbre`, avec des nœuds que l'on coupera uniquement si 3 ou plus de 3 individus jours sont présents sont :

```
w = read.table("https://chesneau.users.lmno.cnrs.fr/profs.txt", header=
T)
attach(w)
library(rpart)
arbre = rpart(Y ~ ., minsplit = 3, data = w)
library(rpart.plot)
rpart.plot(arbre)
```

2.

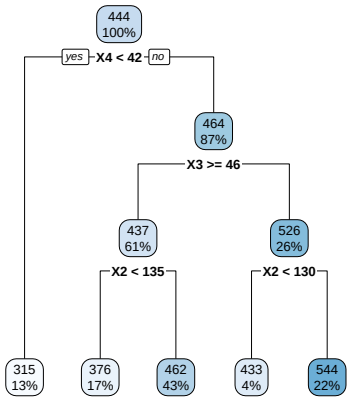
(a) L'enjeu de ces commandes est de modifier l'arbre de régression précédent avec une profondeur

maximale de 2 étages.

(b) Avec la commande `printcp`, on constate sur la cinquième ligne que la `xerror` est la première ayant une valeur plus basse que celles de ces plus proches voisins. On considère alors le paramètre cp associé : $cp = 0.060943$. On peut alors considérer les commandes suivantes :

```
arbre3 = rpart(Y   ., minsplit = 3, data=w, cp = 0.060943)
rpart.plot(arbre3)
```

Cela renvoie :



(c) En parcourant l'arbre précédent, Une valeur plausible de Y pour $X1 = 445$ et $X2 = 141$, $X3 = 52$, $X4 = 45$ et $X5 = 1$, est 462.

Solution 5. Dans un premier temps, on charge le jeu de données `tennis` en précisant que les noms des caractères sont présents en haut de chaque colonne, et on attache les données à ces noms. Ensuite, on charge les librairies `rpart` et `rpart.plot`. Le caractère `Jouer` étant qualitatif, on construit un arbre de classification qui a pour objectif de prédire des valeurs plausibles pour Y à partir des valeurs des autres caractères mis en jeu. On demande les caractéristiques numériques de cet arbre, puis on l'affiche. On remarque alors, que par défaut, l'arbre n'est pas du tout developpé. En effet, les régles d'arrêt sont atteintes dès le debut : principalement, l'option `minsplit` est, par defaut, fixée a 20 : une branche n'est plus developpée dès qu'elle contient 20 individus ou moins. Dès lors, on fait un nouvel essai avec l'option `minsplit = 4`. On obtient un arbre plus développé. On compare alors les prédictions des données de départs à partir de cet arbre afin de déterminer son taux d'erreur.

Solution 6. Par le principe d'un arbre de classification (de type CART), pour un individu vérifiant $X_1 = 2$ et $X_2 = 3.6$, une valeur plausible de Y est donnée par la modalité la plus présente parmi celles des modalités de Y pour les individus ω_3, ω_5, ω_9 et ω_{10}. Comme ces individus ont les modalités : A, A, C et A, la modalité de Y est A.

Solution 7.

1. La plus petite erreur correspond à $G(2, 17)$, avec la valeur 0.8. Par conséquent, le caractère de coupure est X_2 et la valeur seuil associée est 17.
2. Comme $G = 4.5$ et $n = 7$ (correspondant au nombre d'individus), l'indice d'amélioration associé à cette première coupure est

$$J = n(G - G(2, 17)) = 7 \times (4.5 - 0.8) = 25.9.$$

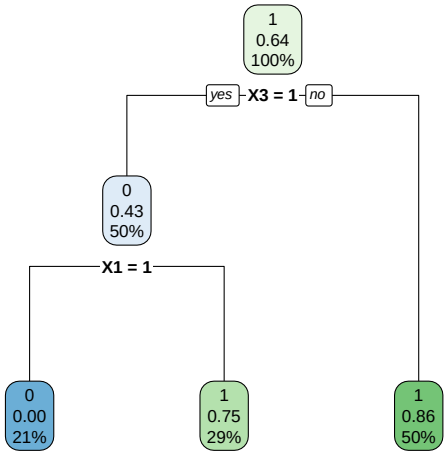
Solution 8.

1. On propose les commandes R suivantes :

```
X1 = c(1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0)
X2 = c(1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1)
X3 = c(1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0)
X4 = c(0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0)
Y = c(0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1)
X1 = as.factor(X1)
X2 = as.factor(X2)
X3 = as.factor(X3)
X4 = as.factor(X4)
Y = as.factor(Y)
w = data.frame(X1, X2, X3, X4)
attach(w)
```

2.

(a) À partir de cette sortie, l'arbre de classification peut se représenter par :



(b) Les commandes qui ont permis d'obtenir cette sortie sont les suivantes :

```
library(rpart)

arbre = rpart(Y ~ X1 + X2 + X3 + X4, minsplit = 4)
```

(c) Donner les commandes qui permettraient de représenter graphiquement l'arbre de classification sont les suivantes :

```
library(rpart.plot)

rpart.plot(arbre)
```

3. On considère l'indice moyen de Gini et la fonction R `GiniMoy` suivante :

```
GiniMoy = function(X,Y)
{
  n = length(X)
  ind0 = which(X == 0)
  n0 = length(ind0)
  nY0 = sum(Y[ind0] == 0)
```

```
  nY1 = n0 - nY0
  I0 = 1 - (nY0 / n0)^2 - (nY1 / n0)^2
  ind1 = which(X == 1)
  n1 = length(ind1)
  nY0 = sum(Y[ind1] == 0)
  nY1 = n1 - nY0
  I1 = 1 - (nY0 / n1)^2 - (nY1 / n1)^2
  C = (n0 / n) * I0 + (n1 / n) * I1
  return(C)
}
```

Dès lors, pour retrouver le fait que le caractère de coupure de la racine est $X3$, on fait :

```
GiniMoy(X1, Y); GiniMoy(X2, Y); GiniMoy(X3, Y); GiniMoy(X4, Y)
```

Cela renvoie :

```
[1] 0.3936508
[1] 0.45
[1] 0.3673469
[1] 0.4285714
```

La plus petite valeur est la troisième, correspondant à `GiniMoy3`. Par conséquent, le caractère de coupure de la racine est $X3$.

4. Pour $X1$ = Soleil, $X2$ = Frais, $X3$ = Sec et $X4$ = Fort, une valeur plausible de Y est $Y = 1$. On peut aussi l'obtenir avec les commandes :

```
predict(arbre, newdata = data.frame(X1 = "1", X2 = "0", X3= "0", X4= "1"), type = "class")
```

5. Le taux d'erreur global de l'arbre de classification est donné par les commandes :

```
pred = predict(arbre, newdata = Y, type = "class")
mc = table(Y, pred)
e = (mc[1, 2] + mc[2, 1]) / sum(mc)
e
```

Cela renvoie :

```
[1] 0.1428571
```

On remarque que le taux d'erreur est faible, ce qui atteste de la bonne qualité prédictive de l'arbre de classification.

Solution 9. Par le principe d'une forêt aléatoire d'arbres de régression, on détermine une valeur plausible de Y pour un individu vérifiant $x1 = 1.8$ et $x2 = 2.3$ pour chaque arbre :

i Pour le premier arbre, une valeur plausible de Y est $y_1 = 0.4$,

ii Pour le deuxième arbre (à droite), une valeur plausible de Y est $y_2 = 1.8$,

iii Pour le troisième arbre, une valeur plausible de Y est $y_3 = 0.19$,

iv Pour le troisième arbre, une valeur plausible de Y est $y_4 = 0.37$.

Dès lors, une valeur plausible de Y à partir de cette forêt aléatoire est

$$\overline{y}_* = \frac{1}{4}\sum_{i=1}^{4} y_i = 0.69.$$

Solution 10. Par le principe d'une forêt aléatoire d'arbres de classification, on détermine une modalité plausible de `yesno` pour un individu vérifiant `bang` $= 0.096$, `dollar` $= 0.045$ et `crl.tot` $= 55$ pour chaque arbre :

i Pour le premier arbre, une modalité plausible de `yesno` est $y_1 =$ `yes`,

ii Pour le deuxième arbre (à droite), une modalité plausible de `yesno` est $y_2 =$ `no`,

iii Pour le troisième arbre, une modalité plausible de `yesno` est $y_3 =$ `no`,

iv Pour le troisième arbre, une modalité plausible de `yesno` est $y_4 =$ `yes`.

Il n'y a pas de modalités plausibles en supériorité numérique, on est dans un cas très particulier. Par conséquent, on ne peut pas vraiment conclure, même si certain logiciels vont sortir la modalité `no` en raisonnant par l'ordre alphabétique (le n de `no` apparaissant avant le y de `yes`).