**Initiation à rpart**

Le package [R](http://www.r-project.org/) [rpart](http://cran.r-project.org/web/packages/rpart/index.html) propose une implémentation des méthodes de construction d'[arbres de décision](https://fr.wikipedia.org/wiki/Arbre_de_d%C3%A9cision) inspirées de l'approche CART décrite dans l'ouvrage éponyme de Breiman, Friedman, Olshen et Stone en 1983. Ce billet propose une prise en main élémentaire de rpart en complément dans la documentation et des vignettes disponibles.

**Construction de l'arbre**

**Préparation des données**

On travaille ici sur les données Titanic disponible dans le package [rpart.plot](http://cran.r-project.org/web/packages/rpart.plot/index.html). On charge les deux packages puis les données comme ceci :

**library**(rpart)

**library**(rpart.plot)

data(ptitanic)

Les données décrivent 1046 passagers selon 6 variables :

* pclass donne la classe tarifaire sur le bateau ;
* survided indique si le passager a survécu ;
* sex donne le genre du passager ;
* age donne l'âge du passager exprimé en années ;
* sibsp donne le nombre de frères, sœurs, mari ou épouse à bord ;
* parch donne le nombre d'enfants ou de parents à bord.

Les trois dernières variables sont numériques alors que les trois premières sont nominales. Il faut bien s'assurer que la représentation en R des variables respecte leur nature. Pour ce faire, on utilise :

**lapply**(ptitanic,**class**)

qui donne la classe de chaque variable, soit ici:

$pclass

[1] "factor"

$survived

[1] "factor"

$sex

[1] "factor"

$age

[1] "labelled"

$sibsp

[1] "labelled"

$parch

[1] "labelled"

Les trois variables nominales sont donc bien des factor. Pour les valeurs numériques, on obtient la classe labelled qui est spécifique au package [Hmisc](http://cran.r-project.org/web/packages/Hmisc/index.html). On peut vérifier en supprimant cette classe que les variables correspondantes sont bien numériques, comme dans le code suivant :

**attr**(ptitanic$age,"class") <- **NULL**

**class**(ptitanic$age)

qui affiche bien

[1] "numeric"

comme attendu.

**Construction de l'arbre complet**

La fonction de construction d'un arbre s'appelle rpart, comme le package. On l'utilise en général avec l'interface des [formules](http://cran.r-project.org/doc/manuals/r-release/R-intro.html#Formulae-for-statistical-models) de R, comme ci-dessous :

ptitanicTree <- rpart(survived~**.**,data=ptitanic)

La formule utilisée survived~. indique qu'on souhaite prédire la variable survived en fonction de toutes les autres. Le principe général est que la (ou les) variable(s) à prédire sont à gauche du symbole ~ alors que les variables prédictives sont à droite du symbole. Ici, le point . permet d'indiquer qu'on souhaite utiliser toutes les variables des données comme prédicteurs sauf les variables à prédire (ce qui évite d'avoir à écrire la liste des prédicteurs).

On a utilisé ici les paramètres par défaut de la fonction rpart, ce qui ne conduit pas toujours à la solution désirée. En effet, rpart ne construit en général pas l'arbre le plus complet possible, pour des raisons d'efficacité. Il est rare en pratique qu'un arbre très profond qui ne réalise aucune erreur de classement sur les données d'apprentissage soit le plus adapté. Il sur-apprend massivement, en général. Il n'est donc pas très utile de construire un tel arbre, puisqu'on devra en pratique l'élaguer.

Cependant, il arrive sur des données de petite taille que les paramètres par défaut de rpart soient trop conservateurs. Par exemple, rpart ne découpe pas une feuille contenant 20 observations. De même rpart demande une amélioration relative d'au moins 1 % de la qualité d'une partition pour effectuer un découpage. Pour changer ces valeurs, il suffit d'utiliser la commande rpart.control en précisant les éléments à modifier. Le code suivant

ptitanicTree <- rpart(survived~**.**,data=ptitanic,control=rpart.control(minsplit=**5**,cp=**0**))

construit un arbre en continuant les découpages dans les feuilles qui contiennent au moins 5 observations (paramètre minsplit) et sans contrainte sur la qualité du découpage (paramètre cp mis à 0). L'arbre construit de cette façon est assez volumineux et contient 65 feuilles.

**Simplification de l'arbre**

**Niveaux de simplification**

Pour choisir le bon niveau de simplification, ou encore le bon nombre de feuilles, on procède par validation croisée. La fonction rpart réalise par défaut une estimation des performances de l'arbre par validation croisée à 10 blocs pour chaque niveau de simplification pertinent. Le nombre de blocs se règle au moment de la construction de l'arbre grâce au paramètre xval de rpart.control.

On peut afficher les résultats de cette opération grâce à la fonction printcp, comme ci-dessous. La courbe indique le taux de mauvaises classifications relativement au score d'origine (dans un arbre réduit à une seule feuille dans laquelle la décision correspond à la classe majoritaire), estimé par la validation croisée. Les barres d'erreur autour de chaque estimation sont aussi obtenues par validation croisée. Ici, comme on a 809 morts et 500 survivants, l'erreur de référence est d'environ 38,2 %. L'axe des abscisses indique la complexité de l'arbre par l'intermédiaire du nombre de feuilles.

plotcp(ptitanicTree)

Une image contenant capture d’écran

Description générée automatiquement

Performances par validation croisée

**Simplification**

Comme attendu, les performances s'améliorent dans un premier temps quand on augmente le nombre de feuilles puis se dégradent en raison du sur-apprentissage. On choisit en général la complexité qui minimise l'erreur estimée, soit ici 11 feuilles. Pour obtenir l'arbre simplifié, on utilise la fonction prune, comme dans le code suivant :

ptitanicSimple <- prune(ptitanicTree,cp=**0.0047**)

On utilise ici le cp affiché sur la courbe, ce qui n'est pas très pratique. Pour automatiser le processus, on peut s'appuyer sur l'attribut cptable des objets arbres. Cette matrice contient toutes les informations utilisées pour produire le graphique ci-dessus. En pratique, on peut retrouver l'arbre optimal en faisant :

ptitanicOptimal <- prune(ptitanicTree,cp=ptitanicTree$cptable[**which.min**(ptitanicTree$cptable[,**4**]),**1**])

Notons que la valeur de cp obtenue de cette façon n'est pas la même que celle affichée sur la figure : les valeurs contenues dans la matrice cptable sont celles qui produisent des changements dans le nombre de feuilles de l'arbre. Pour des raisons de stabilité, on préfère parfois utiliser les moyennes géométriques de deux nombres qui se suivent, ce qui explique la valeur de 0.0047 sur le graphique contre 0.0028 par la table. Dans notre contexte, ceci n'est pas très important.

**Représentation graphique**

Le package rpart propose des fonctions d'affichage relativement limitées. On préfère donc s'appuyer sur rpart.plot, en particulier sur sa fonction prp. La figure ci-dessous est obtenue par le simple appel suivant :

prp(ptitanicOptimal,extra=**1**)

Une image contenant texte, carte

Description générée automatiquement

Arbre simplifié : chaque nœud représente une question, la réponse non étant toujours dans la branche droite de l'arbre. Chaque feuille est étiquetée par la décision associée (ici died ou survived) et par l'effectif classe par classe des individus affectés à la feuille. Par exemple, la feuille la plus à gauche classe les individus en died, avec 660 individus de cette classe et 136 de la classe survived.

**Utilisation de l'arbre**

**Prévision**

Comme tout modèle R, un arbre obtenu par rpart (avec ou sans simplification) peut réaliser des prévisions sur de nouvelles données, en s'appuyant sur la fonction predict. Par défaut, la fonction estime les probabilités d'appartenance aux classes pour chaque observation (simplement par le ratio dans la feuille correspondante). Par exemple le code suivant

predict(ptitanicOptimal,ptitanic[**1**:**10**,])

donne les estimations suivantes sur les dix premiers passagers :

died survived

1 0.0680000 0.9320000

2 0.1111111 0.8888889

3 0.0680000 0.9320000

4 0.8291457 0.1708543

5 0.0680000 0.9320000

6 0.8291457 0.1708543

7 0.0680000 0.9320000

8 0.8291457 0.1708543

9 0.0680000 0.9320000

10 0.8291457 0.1708543

Pour obtenir la classe prédite, il suffit d'ajouter le paramètre type avec la bonne valeur, soit :

predict(ptitanicOptimal, ptitanic[**1**:**10**,], type="class")

ce qui donne :

1 2 3 4 5 6 7 8

survived survived survived died survived died survived died

9 10

survived died

Levels: died survived

**Évaluation des performances**

Pour évaluer correctement les performances de l'arbre simplifié, il faut utiliser une procédure de type validation croisée en plus de celle qui est intégrée dans rpart pour choisir la complexité de l'arbre. Le package [caret](http://caret.r-forge.r-project.org/) facilite cette opération en proposant des fonctions de découpage des données bien conçues.

Dans cette prise en main, nous nous contentons d'évaluer les performances sur les données d'apprentissage ce qui sur-estime la qualité du modèle. Il s'agit simplement d'une illustration ! Le principe est de s'appuyer sur la fonction predict et sur la fonction table pour obtenir une matrice de confusion. L'appel suivant

**table**(ptitanic$survived, predict(ptitanicOptimal, ptitanic, type="class"))

produit ainsi la matrice de confusion suivante :

died survived

died 752 57

survived 173 327

On constate que la qualité de la prédiction dépend beaucoup de la classe. En effet, sur les 809 passagers décédés, le taux de prévisions correctes est de 93 % environ, alors que sur les 500 survivants, il n'est que de 65 % environ.