HEC MONTRÉAL

Comparaison des performances prédictives de deux algorithmes construisant des forêts aléatoires

par
Isabelle Bernard

Sciences de la gestion
Option : Intelligence d'affaires

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences en gestion
(M. Sc.)

Résumé

Les forêts aléatoires sont régulièrement utilisées comme modèle de prédiction en raison notamment de leurs excellentes performances prédictives, et de la facilité de leur utilisation.

L'algorithme classique, le plus fréquemment utilisé pour produire de telles forêts se base sur des arbres de type CART. Or, dans la littérature, ce type d'arbre est dénoncé comme étant biaisé lors de la sélection des variables et des méthodes alternatives lui sont proposées.

Notre recherche vise ici à comparer la performance prédictive des forêts aléatoires classique à l'alternative la plus fréquemment utilisée parce qu'implantée dans une librairie du logiciel R : *cforest*, basée sur des arbres d'inférence conditionnelle.

En utilisant 20 jeux de données réels et 6 jeux de données produits par simulations, nous comparerons ici la performance prédictive des deux algorithmes. Pour évaluer l'effet du bruit, de l'écart-type et des types de variables explicatives, des jeux de données artificiels seront également construits. Plus précisément, 3 scénarios seront étudiés: 1) Toutes les variables explicatives seront continues. 2) Un mélange de variables explicatives continues et catégorielles seront utilisées. 3) Seules des variables explicatives catégorielles seront inclues. Chacun des scénarios sera étudié avec un nombre différent d'observations composant le jeu de données d'entraînement, une option introduira du bruit dans les données, l'écart-type sera de 1 ou de 3 et le nombre de variable choisi aléatoirement pour la construction des arbres sera de 4 ou de 11. Nous mesurerons la performance de ces simulations en utilisant deux mesures : l'erreur quadratique moyenne et l'erreur moyenne absolue et ce pour les arbres construits à partir de CART (*rpart*- RP), les arbres construits à partir d'inférence conditionnelle (*ctree*- CT), les forêts classiques (*RFSRC* - RF) et les forêts alternatives (*cforest*- CF*).*

Mots clés : forêts aléatoires; prédiction; modèle; arbres de décision; CART; cforest

Table des matières

Liste des tableaux et des figures

Liste des abréviations

BH : *Boston Housing Data* (ensemble de données)

BC : Wisconsin *Breast Cancer Database* (ensemble de données)

CART : Classification And Regression Trees

CFOREST : Forêts aléatoires basées sur CTREE

CMC: Contraceptive Method Choice (ensemble de données)

CTREE : Conditional Inference Trees

DNA : Primate splice-junction gene sequences (ensemble de données)

F1 : *Benchmark Problem Friedman 1* (ensemble de données simulé)

F2 : *Benchmark Problem Friedman 2* (ensemble de données simulé)

F3 : *Benchmark Problem Friedman 3* (ensemble de données simulé)

GC : Statlog (German Credit Data) Set (ensemble de données)

HV : HouseVote84 : United States Congressional Voting Records 1984 (ensemble de données)

LD : *Liver Disorders* (ensemble de données)

LR : Letter Image Recognition Data (ensemble de données)

PID : Pima Indians Diabetes Database (ensemble de données)

RFSRC : Random Forest SRC (basées sur arbres CART ‘classiques’)

RN : *Ringnorm Benchmark Problem* (ensemble de données simulé)

SMO : Analysis of Attitudes Towards Workplace Smoking Restrictions (ensemble de données)

T2N : *Twonorm Benchmark Problem* (ensemble de données simulé)

T3N : *Threenorm Benchmark Problem* (ensemble de données simulé)

TAE : Teaching Assistant Evaluation (ensemble de données)

VH : *Vehicle Silhouettes* (ensemble de données)

WF :*Waveform Database Generator* (ensemble de données simulé)

Liste des annexes

Remerciements

Je tiens tout d'abord à remercier Denis Larocque puisqu'il est non seulement un excellent professeur capable de vulgariser une partie de ses connaissances extensives et de les transmettre à un public parfois néophyte, mais également un directeur généreux et serviable. Merci également aux professeurs François Bellavance et Jean-François Plante du HEC pour avoir soutenu et accompagné notre équipe lors du concours de *Data mining* de la Société de Statistique du Canada et pour avoir aiguisé mon intérêt pour le *machine learning*.

Merci au Conseil de recherche du Canada d'avoir financé par une bourse d'excellence ce projet de recherche. Merci à l'Association des diplômés de HEC Montréal pour une seconde bourse d'excellence obtenue en fin de parcours.

Merci à Jorge Ruiz et Fabio Guacaneme pour leur aide précieuse.

Enfin, sur une note plus personnelle, merci à Osma de m'avoir supportée et encouragée pendant le périple inhabituellement long qu'a duré la rédaction de ce court mémoire.

Merci à mon frérot Sébastien d'avoir pensé à me faire découvrir un domaine jusqu'alors pour moi inconnu et qui me passionne aujourd'hui: celui de l'intelligence d'affaires et du *Data Science.*

Merci à mes parents pour le coup de pouce donné pour le financement du programme de cette maîtrise.

Merci à vous tous d'avoir fait naître chez moi une nouvelle passion pour les données et de m'avoir permis de vivre un nouveau début de carrière.

Chapitre 1
Introduction

Mise en contexte

Depuis l'article inaugural de Breiman (2001), les forêts aléatoires comme modèle de prédiction sont fréquemment utilisées dans plusieurs disciplines.

Ce type de modèle offre une grande performance prédictive, est versatile et détecte les interactions sans qu'une forme paramétrique n'aie à être spécifiée (Hamza and Larocque 2005). Une forêt aléatoire est une méthode d'ensemble qui consiste à combiner plusieurs arbres de décision et classification. Chacun des arbres composant la forêt est construit en sélectionnant au hasard, à chaque nœud, un nombre restreint de variables explicatives et d'observations avec remise. Cet ajout d'aléatoire augmente la diversité des arbres.

L'algorithme original, et celui qui est le plus utilisé aujourd'hui encore, utilise des arbres construits à partir du paradigme CART (Breiman 1984). Plusieurs chercheurs dénoncent néanmoins un biais de sélection des variables dans cette méthode. Les arbres CART favoriseraient :

1) les variables continues ou catégorielles avec plusieurs possibilités de point de coupure (Kim and Loh 2001) et 2) les variables ayant plusieurs valeurs manquantes (lorsque le critère de Gini est utilisé) (Strobl, Boulesteix, and Augustin 2007). Les forêts aléatoires construites à partir des arbres CART proposent une mesure d'importance des variables qui est également biaisée dans l'identification des variables les plus importantes. Différents algorithmes permettant de construire des arbres 'non-biaisés' ont été proposés pour éviter ces problèmes. Cependant, les capacités prédictives des forêts aléatoires construites à partir d'arbres biaisés et non-biaisés n'ont pas été comparées.

Objectifs de recherche

Les arbres de décisions construits à partir de l'algorithme CART souffrent d'un biais de sélection de variables. Lorsque ces arbres sont utilisés pour créer des forêts aléatoires, il subsiste un biais dans la mesure de l'importance des variables. La question de recherche qui découle de cette problématique est la suivante : Ce biais affecte-t-il les capacités de prédictions des forêts aléatoires construites à partir de tels arbres ?

Pour répondre à cette question, ce mémoire vise à comparer la performance de prédiction de deux algorithmes permettant de construire des arbres et forêts aléatoires. Le premier algorithme construira des arbres et forêts aléatoires 'classiques' basées sur des arbres de type CART. Le second propose des arbres et forêts aléatoires basées sur des arbres 'non-biaisés' construits par inférence conditionnelle. Les deux algorithmes sont implantés dans le logiciel libre R, respectivement dans les packages '`rpart`' (arbres) (Thernau, Atkinson, and Ripley 2018) et `randomForestSRC`' (forêts) (Ishwaran and Kogalur 2017) et '`partykit`' (Hothorn and Zeileis 2015). Comme mentionné par Rusch et Zeileis (2014), l'implantation des algorithmes proposés par les chercheurs dans des logiciels comme R et Python est cruciale pour l'utilisation de ceux-ci par la communauté, mais aussi pour encourager la comparaison entre les différents algorithmes et pour réduire la fragmentation dans la littérature à propos des arbres.

La comparaison des deux algorithmes se fera d'abord sur 20 ensembles de données réels et 7 ensembles de données produits par simulations ayant préalablement été utilisés dans la littérature. L'évaluation de la performance prédictive se fera par validation croisée de l'erreur quadratique moyenne pour les ensembles de données avec une variable-réponse continue et du taux de bonne classification pour ceux avec une variable-réponse catégorielle.

Ensuite, nous construirons par simulation de nouveaux jeux de données pour évaluer les effets : du bruit présent dans les variables explicatives; de la dispersion entre les valeurs de la variable réponse; du nombre de variables choisi aléatoirement pour construire les arbres; et des types de variables explicatives. Trois scénarios seront étudiés: 1) Toutes les variables explicatives seront continues; 2) Seules des variables explicatives catégorielles seront inclues; 3) Un mélange de variables explicatives continues et catégorielles seront utilisées. Chacun des scénarios sera étudié avec un nombre différent d'observations composant le jeu de données d'entraînement, une option introduira du bruit dans les données, l'écart-type sera de 1 ou de 3 et le nombre de variables choisi aléatoirement pour la construction des arbres sera de 4 ou de 11. Nous mesurerons la performance de ces simulations en utilisant deux mesures : l'erreur quadratique moyenne et l'erreur moyenne absolue et ce pour les arbres construits à partir de CART (rpart- RP), les arbres construits à partir d'inférence conditionnelle (ctree- CT), les forêts classiques (RFSRC - RF) et les forêts alternatives (cforest- CF).

Selon notre revue de la littérature, les algorithmes ont été comparés au niveau des arbres de décision dans le but de montrer le biais de sélection des variables de CART. Souvent, cette comparaison se faisait dans le but de proposer un nouvel algorithme non-biaisé pour la construction des arbres. Les forêts aléatoires sont fréquemment utilisées pour faire des prédictions. Cependant, il ne semble y avoir aucune publication cherchant à comparer les performances prédictives non pas des arbres, mais de différents modèles de forêts aléatoires. Ce mémoire répond donc à ce manque dans la littérature.

Structure du mémoire

Ce travail débute par une revue de la littérature relative aux forêts aléatoires et au problème du biais dans la sélection des variables des arbres construits à partir de l'algorithme CART. Ensuite, nous exposons la méthodologie adoptée dans cette

recherche. Dans un premier temps, chaque ensemble de données réel et chaque simulation effectuée seront ici présentés en détails. Nous expliquerons également comment les prédictions ont été effectuées et comparées.

Une deuxième étape consistera à expliquer les trois scénarios qui ont été construits dans le but d'étudier des caractéristiques spécifiques des jeux de données dans les résultats de prédiction des deux modèles. Nous présenterons dans une série de douze graphes les résultats de simulations pour chacun des scénario (3) et des écart-types (2 variantes) en considérant deux mesures de l'erreur de prédiction (2). Dans chacun des graphes, huit colonnes représentant les différentes combinaisons de nombres d'observation dans le jeu de données d'entraînement, le nombre de variable utilisées aléatoirement pour construire les arbres, et la présence ou non de bruit.

Finalement, une discussion des résultats tentera de souligner l'effet des différentes caractéristiques propres à chaque scénario pour chacun des modèles testés. Bien que le but premier de l'étude soit de comparer les forêts, une comparaison sommaire de la différence des résultats entre les arbres des deux méthodes sera également proposée avant de conclure.

Chapitre 2
Revue de la littérature

Les arbres de décision

Un arbre se construit à partir d'une succession de nœuds. À chaque étape de cette succession, les données du nœud parent (en haut de l'arbre) sont séparées en deux groupes (constituant deux nouveaux nœuds enfants) en fonction d'un point de coupure (*split*) sur une variable. Le but lors de la séparation est de maximiser la différence entre l'impureté du nœud parent et la moyenne des deux nœuds enfants. C'est ainsi que la variable et le point de coupure à utiliser sont déterminés. Pour la majorité des algorithmes, on quantifie l'impureté par une mesure d'entropie (généralement le coefficient de Gini). Cette étape est reproduite plusieurs fois.

Dépendamment de la méthode utilisée, l'arbre est soit : 1) arrêté lorsque le nombre d'observations dans l'un des nœuds enfant n'est pas suffisant; 2) arrêté lorsque la diminution de l'impureté n'est pas suffisante en choisissant la meilleure variable et le meilleur point de coupure; ou 3) développé beaucoup plus loin, puis élagué (*pruning*) pour obtenir la meilleure prévision possible. Cette étape permet de réduire le sur-apprentissage en coupant les branches n'augmentant pas la valeur prédictive. Il s'agit de la méthode que nous avons utilisée pour construire les arbres qui seront utilisés pour comparer les prédictions que nous verrons plus loin. Les forêts n'utilisent cependant généralement pas l'élagage puisque le sur-apprentissage sera différent pour chaque arbre qui est construit à partir d'un échantillon de variables et d'observations différent et aléatoire.

Amaratunga, Cabrera et Lee (2008) proposent cette description de ce qu'est un arbre de classification (avec variable réponse catégorielle):

> Given a training set X comprised of N cases, which belong to two classes, and G features, a classification tree can be constructed as follows. First, a feature x and a threshold t that splits X into two subsets that are maximally distinct according to a specified criterion are selected from all features of x and all possible values of t. The training set is then split into the two buckets XL and XR depending on whether or not x<t. This procedure is repeated with each of XL and XR using another (x, t) combination. This process is repeated until no further splitting is possible (2008, p. 2010).

Certains éléments diffèrent parmi les algorithmes de construction d'un arbre. D'abord, la façon de mesurer la distinction maximale entre les deux sous-groupes créés (aussi souvent appelée l'impureté) après chaque point de séparation. Ensuite, les critères pour choisir les variables explicatives les plus discriminantes qui se trouveront en haut de l'arbre. Aussi, le critère permettant de décider si le groupe qui vient d'être créé est final ou s'il doit être divisé à nouveau. Finalement, l'arrêt du développement de l'arbre à un point donné ou l'élagage (*pruning*).

La figure 1 illustre ce qu'est un arbre. Pour construire l'arbre, nous avons utilisé, dans R, les packages `rpart` (Thernau, Atkinson, and Ripley 2018) et `rpart.plot` (Milborrow 2017) et l'ensemble de données Iris qui contient 150 observations de fleurs iris. La formule consiste à prédire l'espèce (*Species*), une variable-réponse à trois catégories (*Iris setosa*, *versicolor* et *virginica*), à partir des 4 autres variables indiquant la longueur et la largeur du sépale et du pétale. On voit ici que le modèle a choisi de n'utiliser que deux des quatre variables explicatives : la longueur du pétale et sa largeur.

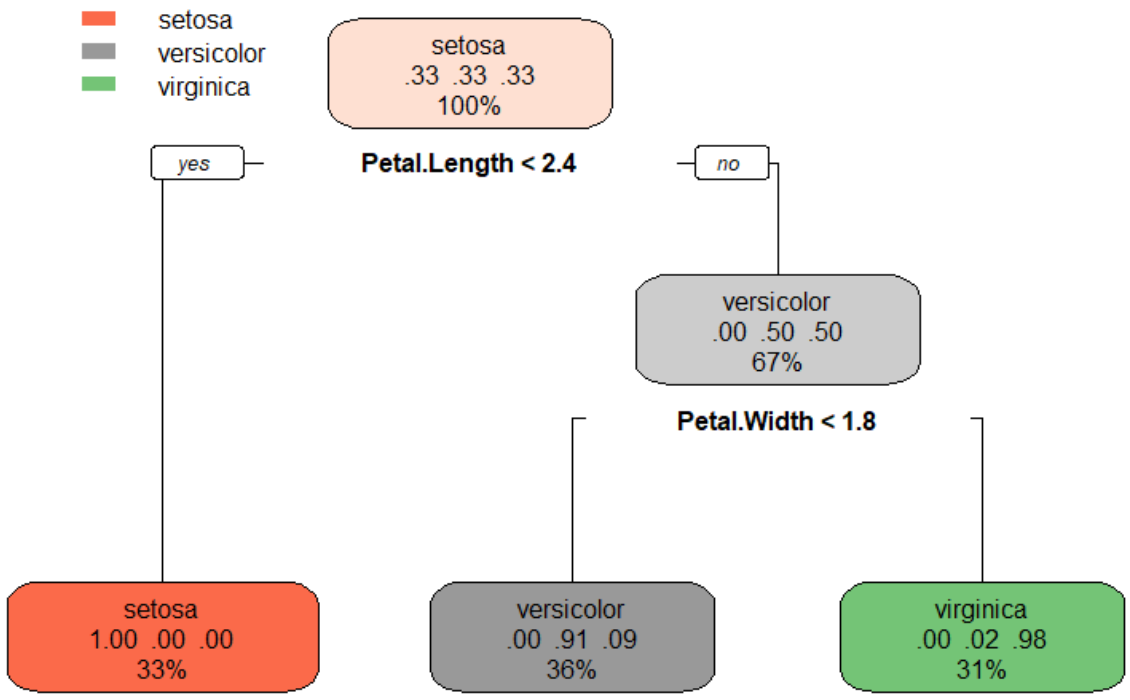


Figure 1. : Arbre construit à partir de l'ensemble de données Iris

Dans la littérature, c'est Morgan et Sonquist (1963) qui proposèrent Automatic Interaction Detection – AID, le premier algorithme permettant de construire un arbre de régression. L'algorithme AID utilise la variable et le point de coupure qui maximiseront l'impureté entre les deux nouveaux groupes créés. Ici, c'est l'écart-type qui est utilisé pour la mesurer. Le processus s'arrête lorsque le nœud final atteint une proportion prédéfinie du nœud original. Un problème aujourd'hui connu de ce premier algorithme est le sur-apprentissage important sur les données de l'échantillon d'entrainement.

D'autres algorithmes ont également été proposés dans les années qui suivirent dont THAID (Messenger and Mandell 1972), l'ancêtre de CART (Breiman and Meisel 1976) et CHAID (Kass 1980). Bien qu'il s'agisse d'un nouveau type de modèle novateur, les

arbres de décision et classification ont peu retenu l'attention des chercheurs de l'époque et sont demeurés peu utilisés jusqu'en 1984.

Le livre de Breiman et al. (1984) changea cette réalité et l'approche fut popularisée. Le livre est le résultat de deux années d'expérimentations cherchant à déterminer les meilleures règles pour choisir un point de coupure optimal (Cutler 2010). L'algorithme de Classification and Regression Trees (CART) suggère d'élaguer les arbres plutôt que d'arrêter leur progression à un moment prédéfini. L'élagage des arbres après leur construction est la principale innovation de CART et réglait notamment le problème de sur-apprentissage qu'avait AID. L'impureté des nœuds est ici mesurée différemment selon le coefficient de Gini (une seconde variante consiste à utiliser le critère de Twoing) (Cutler 2010). Aujourd'hui encore, il s'agit de l'algorithme le plus utilisé pour construire des arbres et il a été implanté dans tous les logiciels de statistiques majeurs (package `rpart` dans R).

Biais des arbres CART et propositions d'alternatives

Depuis le début de l'utilisation de cet algorithme cependant, des critiques dénoncent un biais dans la sélection des variables les plus importantes et qui seront utilisées pour les points de coupures en haut de l'arbre. Même Breiman et al. (1984) admettaient ce biais pour les variables contenant plus de valeurs quand le critère de séparation du plus grand gain d'information est mesuré par coefficient de Gini : "[…] variable selection is biased in favor of those variables having more values and thus offering more splits " (42).

White et Liu (1994) étudient le biais en appliquant différents algorithmes de construction des arbres sur des données simulées. Ils démontrent que les mesures habituelles de gain utilisées pour déterminer les points de coupures des arbres sont biaisées en faveur des variables contenant une grande quantité de valeurs. Ils concluent

que les approches utilisant le khi-deux sont préférables puisqu'elles compensent automatiquement la quantité de valeurs possibles de chaque variable.

En 1997, Loh et Shih proposèrent l'algorithme QUEST qui arrive à réduire considérablement le biais de sélection et qui est beaucoup plus rapide à exécuter que les algorithmes précédents. Ils comparent ensuite les résultats des prédictions de ce nouvel algorithme à ceux reposant sur une recherche exhaustive du meilleur point de coupure et concluent que la différence dans l'estimation de l'erreur n'est pas statistiquement significative. La différence en rapidité cependant est telle qu'une validation-croisée qui prenait dix jours à effectuer avec une recherche exhaustive, prend maintenant dix-sept secondes avec QUEST (Loh and Shih 1997).

En 2000, Lim, Loh et Shih comparent vingt-deux types d'arbres de décision différents (en plus de neuf méthodes statistiques et de deux réseaux de neurones) en les appliquant sur seize ensembles de données (nous reprendrons ici plusieurs de ces ensembles de données dans notre propre comparaison). La comparaison porte sur trois éléments : la justesse des classifications (capacité prédictive), le temps d'apprentissage et le nombre de feuilles pour les arbres. Parmi les arbres de décision, QUEST se classe au premier rang (4^{e} de tous les algorithmes) pour ce qui est de la justesse de classification (Lim, Loh, and Shih 2000).

Dans un article publié en 2001, Shih compare les différentes méthodes permettant de déterminer le meilleur point de coupure à chaque nœud lors de la construction d'un arbre. Alors que l'approche naïve consisterait à chercher parmi tous les points de coupure possibles, il s'agit d'un processus qui prendrait un temps énorme. Des méthodes alternatives permettant de gagner du temps ont donc été proposées et l'auteur les compare ici. Lorsque la variable-réponse est binaire, Shih suggère l'utilisation de l'algorithme présenté dans Breiman et al. (1984) (CART). Cependant, pour une

variable-réponse catégorielle la suggestion se déplace vers un algorithme beaucoup plus rapide de Mola et Siciliano (1999). De plus, lors de la construction de l'arbre, celui-ci est généralement construit au maximum de son développement avant d'être élagué (raccourci vers le haut) en utilisant la validation-croisée. Ce processus d'élagage développé par Breiman (1984) et Ripley (1996) offre un gain d'efficacité qui est amplifié (Shih 2001).

L'article de Kim et Loh (2001) est fondamental pour la démonstration du biais de sélection des variables de CART. Les chercheurs y démontrent que CART tend à privilégier les variables qui ont plus de valeurs manquantes que les autres en plus d'être biaisé en fonction du plus grand nombre de catégories. Ils testent trente-quatre algorithmes : les mêmes que (Lim, Loh, and Shih 2000) en plus de CRUISE et CART sur plusieurs ensembles de données de classification avec variable-réponse binaire. Le nouvel algorithme qu'ils proposent, CRUISE, utilise des points de séparation multiples (*univariate split method with multiway splits*) permettant notamment de créer plus de deux nœuds-enfants. Ils présentent la force de CRUISE comme ayant un biais de sélection négligeable (à la fois pour les variables ayant un grand nombre de catégories et pour les variables ayant plusieurs valeurs manquantes).

Bien que cet article (Kim et Loh 2001) soit paru avant Loh (2002), ce dernier texte est constamment cité (comme étant à paraître) et CRUISE se veut un complément de l'algorithme 'Generalized, Unbiased, Interaction, Detection and Estimation' (GUIDE) (Loh 2002) : "It borrows and improves upon ideas from many sources, but especially from FACT, QUEST, and GUIDE (Loh 2001) [2002] for split selection and CART for pruning (Kim and Loh 2001, 590)". GUIDE vise également à construire des arbres non-biaisés. Cependant, dans l'article paru en 2002, l'algorithme est proposé pour une variable-réponse continue.

> Here, the association between the sign of model residuals and each covariate is measured by a *P* value derived from a χ^2 test. Continuous covariates are categorized to four levels prior to variable selection; however, models are fitted to untransformed covariates in the nodes (Hothorn, Hornik, and Zeileis 2006, 653).

Dans Loh 2002, une comparaison empirique est proposée comparant différentes variantes de GUIDE et de CART sur six ensembles de données (dont trois que nous reprendrons dans la présente étude). L'erreur quadratique moyenne est calculée par validation-croisée à dix groupes. Les conclusions de l'article sont que l'erreur quadratique moyenne de GUIDE version '*piecewise constant model*' est à ± 20% de celle de CART. Pour ce qui est de l'autre version, '*piecewise linear regression model'*, surpasse les prédictions de l'algorithme CART (Loh 2002).

Dans un court article paru en 2004, Shih explique la raison du biais de sélection des variables dans le cas d'une méthode essayant l'ensemble des points de coupures possible, puis sélectionnant parmi toutes ces possibilités le point proposant le plus de différenciations dans les nœuds enfants. Suite à cette explication statistique, il propose d'utiliser la p-value pour mesurer la différence entre les nœuds enfants tout en utilisant le khi-deux de Pearson comme critère de séparation (Shih 2004).

Wei-Yin Loh a conçu plusieurs versions de GUIDE visant toujours à construire des arbres réduisant les biais de sélection de variables. La plus récente de celles-ci (Loh 2009) vise à reprendre les forces des algorithmes QUEST (Loh and Shih 1997) et CRUISE (Kim and Loh 2001) tout en corrigeant leurs faiblesses (Loh 2014, 333). Un logiciel de mise en application de l'algorithme est disponible pour téléchargement au :

http://www.stat.wisc.edu/~loh/guide.html [1].

En 2006, dans un article co-écrit par Hothorn, Hornik et Zeilis, l'algorithme 'Conditional Inference Trees' (CTREE) est proposé :

[1] Page consultée le 5 août 2017.

> It uses p-values from permutation distributions of influence function-based statistics to select split variables. Monte Carlo or asymptotic approximations to the p-values are employed if they cannot be computed exactly. CTREE does not use pruning; it uses stopping rules based on Bonferroni-adjusted p-values to determine tree size. The algorithm is implemented in the R package PARTY. (Loh 2014)

Les auteurs démontrent que leur algorithme construit des arbres structurellement différents (par le choix des variables) que ceux utilisant des recherches exhaustives pour déterminer un point de coupure (CART). Leur algorithme est ensuite empiriquement comparé avec les arbres produits par le package dans R 'rpart' (CART) et avec QUEST/GUIDE. Sur la majorité des 12 ensembles de données réels testés, la performance de CTREE est statistiquement équivalente (ils fixent le seuil à 10%) à ceux des compétiteurs :

> Equivalent performance between conditional inference trees and rpart cannot be postulated for the Glass data. The performance of the conditional inference trees is roughly 10% worse compared with rpart. In all other cases, the performance of conditional inference trees is better than or equivalent to the performance of exhaustive search (rpart) and unbiased procedures (QUEST or GUIDE) with pruning. The conditional inference trees perform better compared to rpart trees by a magnitude of 25% (Boston housing), 10% (ionosphere), and 15% (ozone). The improvement upon unbiased QUEST and piecewise constant GUIDE models is 10% for the Boston housing data and 50% for the ionosphere and soybean data. For all other problems, the performance of conditional inference trees fitted within a permutation testing framework can be assumed to be equivalent to the performance of all three competitors (Hothorn, Hornik, and Zeileis 2006, 668).

Deux modèles d'arbres basés sur une structure différente peuvent donc obtenir des résultats de prédiction similaires :

> This result is interesting from a practical point of view. It implies that two recursive partitioning algorithms can achieve the same prediction accuracy but, at the same time, represent structurally different regression relationships, that is, different models and thus may lead to different conclusions about the influence of certain covariates on the response (Hothorn, Hornik, and Zeileis 2006, 669).

On utilise souvent les arbres de décisions parce qu'ils s'interprètent facilement au niveau du choix des variables. C'est la raison pour laquelle, malgré la performance prédictive similaire des algorithmes CTREE et CART (rpart), les auteurs soutiennent l'utilisation de leur méthode parce qu'elle est non-biaisée :

> […] our findings contradict the common opinion that pruning procedures outperform algorithms with internal stopping with respect to prediction accuracy. […]
>
> However, while the predictions obtained from conditional inference trees are as good as the predictions of pruned exhaustive search trees, the partitions induced by both algorithms differ structurally. Therefore, the interpretations obtained from conditional inference trees and trees fitted by an exhaustive search without bias correction cannot be assumed to be equivalent. Thus, two rather different partitions, and therefore models, may have equal prediction accuracy. Since a key reason for the popularity of tree based methods stems from their ability to represent the estimated regression relationship in an intuitive way, interpretations drawn from regression trees must be taken with a grain of salt (Hothorn, Hornik, and Zeileis 2006, 671).

L'implantation de l'algorithme dans le logiciel R a pour conséquence une utilisation de celui-ci beaucoup plus importante que n'importe lequel des autres algorithmes 'non-biaisés'. Dans la présente étude, nous reprendrons la majorité de ces ensembles de données réels pour comparer les forêts aléatoires issues de CTREE et celles créées par CART.

En 2007, Strobl, Boulesteix et Augustin poursuivent la critique du biais de sélection des algorithmes, comme CART, qui utilisent les 'mesures standard d'impureté' pour déterminer le point de coupure. Ils démontrent ici que le biais que l'on connaissait dans le cas de la construction des arbres se poursuit lorsqu'une forêt aléatoire est construite à partir de tels arbres (CART, notamment). C'est alors la mesure d'importance des variables (VIM) qui est faussée et accorde une importance trop grande à certaines variables, notamment celles contenant beaucoup de valeurs manquantes. Une méthode alternative pour déterminer le meilleur point de coupure est ici aussi proposée (Strobl, Boulesteix, and Augustin 2007).

En résumé, pour ce qui est de la construction des arbres de décision, l'algorithme CART tend à favoriser deux types de variables:

1) les variables continues ou catégorielles avec un très grand nombre de catégories (puisqu'elles offrent plus de possibilité de points de coupures);

2) les variables contenant plusieurs valeurs manquantes.

Plusieurs algorithmes de construction d'arbres de décision ne reproduisant pas le biais de sélection ont donc été proposés. Chacun propose des particularités différentes faisant qu'il est parfois mieux adapté à un type particulier de données.

Comme nous avons vu, plusieurs articles comparent les capacités de prédiction et les biais de sélection de variables des arbres construits à partir de différents algorithmes. Cependant, nous n'avons trouvé aucun article étudiant les différences entre les capacités de prédiction des forêts aléatoires biaisées et non-biaisées.

Les forêts aléatoires

Une forêt aléatoire est une méthode d'ensemble combinant un grand nombre d'arbres différents. Pour chaque arbre, on choisira aléatoirement une petite portion des observations pour l'entraînement du modèle, en prélevant un échantillon bootstrap. Breiman (1996b) nomme cette étape '*bagging*' : les observations choisies sont les '*in-bag cases'* alors que celles qui ne sont pas utilisées sont *'out-of-bag cases'*. Cette étape sera répétée à chaque itération d'un nouvel arbre avec remplacement. Une seconde étape consistera à considérer seulement une partie des variables indépendantes (par défaut la racine carrée ou encore le tiers) choisie aléatoirement, et ce, à chaque nœud de l'arbre. Ces deux étapes de sélection aléatoire des observations et des variables augmenteront grandement la diversité des arbres ainsi construits. Le résultat final consiste à faire la moyenne dans le cas d'une régression ou encore à choisir la catégorie obtenue le plus souvent en compilant le mode à partir de tous les arbres dans le cas d'une classification.

La différence entre les forêts aléatoires 'classiques' et les alternatives 'non-biaisées' se trouve dans les critères de séparation contenus dans les arbres à la base de la forêt :

> The splitting rule is a central component to CART methodology and crucial to the performance of a tree. However, it is widely believed that ensembles such as RF which aggregate trees are far more robust to the splitting rule used (Ishwaran 2015, 76).

C'est en 2001 que Breiman publie l'article séminal qui introduit la méthode des forêts aléatoires. Comme l'explique Cutler, il s'agit de l'aboutissement de plusieurs années de recherches sur l'amélioration des modèles basés sur les arbres :

> In the light of boosting, Breiman spent a lot of time trying to improve individual trees [Shang and Breiman (1996), Breiman (1998b)] and bagged trees [Breiman (2000a, 2001b)]. He also worked very hard to understand what was going on with boosting [Breiman (1997a, 1998a, 1998c, 1999a, 2000b, 2004b)]. However, he never seriously produced a boosting algorithm for practical use, and I believe the reason was that he wanted a method that could give meaningful results for data analysis, not just prediction, and he didn't think he could get this by combining dependent predictors. The culmination of his work on bagging and how to improve it, and his work trying to understand boosting, was a method Breiman called "random forests" (RF) [Breiman (2001a)]. Random forests fit trees to independent bootstrap samples from the data. The trees were grown large (for classification) and at each node independently, m predictors were chosen out of the p available, and the best possible split on these m predictors was used. As a default for classification, Breiman settled on choosing $m = \sqrt{p}$. In RF we see a synthesis of the bagging ideas (bootstrapping), along with ideas that came from boosting (growing large trees), and Breiman's understanding of how to increase instability (randomly choosing predictors at each node) to get more accurate aggregate predictions. Once he came up with RF, Breiman stopped working on new algorithms and started work on how to get the most out of the RF results. He developed measures of variable importance and proximities between observations (Cutler 2010, 1629).

La méthode est ainsi présentée dans l'article de Breiman :

> Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (Breiman 2001, 5).

> The simplest random forest with random features is formed by selecting at random, at each node, a small group of input variables to split on. Grow the tree using CART methodology to maximum size and do not prune (Breiman 2001, 11).

L'article présente des résultats prédictifs : la nouvelle méthode est testée sur 16 ensembles de données réels (dont plusieurs avaient été utilisés dans les recherches précédentes de Breiman) et 4 artificiels. Nous reprendrons plusieurs de ces ensembles de données dans la présente recherche. La force des forêts aléatoire est ici empiriquement

illustrée par comparaison avec la méthode *Adaboost*[2]: la nouvelle méthode est comparable pour la capacité de prédiction, est beaucoup plus rapide et plus efficace lorsque les données contiennent beaucoup de bruit. Selon Google Scholar, l'article a aujourd'hui été cité dans 36 628 publications[3].

Breiman avait développé en fortran le code permettant d'appliquer son algorithme. En 2003, lui et Adele Cutler publient le code en fortran permettant d'utiliser l'algorithme, accompagné d'une documentation :

http://www.math.usu.edu/~adele/forests/cc_home.htm[4].

En 2002, Andy Liaw et Matthew Wiener implantent l'algorithme dans R (2002). Depuis, les packages `randomForests` (Liaw and Wiener 2015) et `randomForestSRC` (Ishwaran and Kogalur 2017) permettent d'utiliser les arbres et forêts aléatoires conçues par Breiman.

Les forêts aléatoires sont rapidement devenues des méthodes de prédictions grandement utilisées dans plusieurs disciplines. La popularité des forêts aléatoires est due à la versatilité et à la puissance de prédiction de ce type de modèle. Un avantage considérable des forêts aléatoires est qu'elles peuvent automatiquement détecter les interactions sans devoir spécifier une formule paramétrique. De plus, elles s'adaptent aux calculs parallèles ce qui les rend pertinentes à l'ère des métadonnées. De plus, l'implantation de l'algorithme dans les principaux logiciels de statistiques et d'apprentissage machine (R, Python, SAS E-Miner, …) permettent un accès facile à la méthode.

[2] "Breiman referred to the AdaBoost algorithm as 'the most accurate general purpose classification algorithm available'" (2004). 'Consistency for a simple model of random forests'. *Technical Report* 670. Dept. Statistics, Univ. California, Berkeley, CA. cité par : (Cutler 2010, 1628).
[3] Page consultée le 5 juin 2018.
[4] Page consultée le 20 juin 2017.

En 2005, Hamza et Larocque prouvent empiriquement la force de prédiction de l'algorithme des forêts aléatoires 'classiques' (Breiman). Les chercheurs comparent 25 méthodes de classification sur 14 ensembles de données réels qui avaient également été utilisé par (Lim, Loh, and Shih 2000) et doublent ces ensembles de données en ajoutant du bruit pour un total de 28 ensembles de données. Les forêts aléatoires (classiques, basées sur des arbres de type CART) surpassent significativement les autres méthodes pour leurs capacités de prédiction et pour leur robustesse lorsque les données contiennent beaucoup de bruit (Hamza and Larocque 2005).

Un autre avantage important des forêts aléatoires (lié à la résistance au bruit) est que, contrairement à plusieurs autres modèles, elles ne sont pas trop sujettes au sur-apprentissage *(overfitting),* et ce, même avec les ensembles de données contenant un nombre important de variables. Cela découle du fait que seul un échantillon des variables et des données sont utilisés pour construire chacun des arbres composant la forêt :

> Most classification procedures when faced with a situation in which there are a large number of features exhibit a tendency to overfit. However, a major advantage of random forests is that they are able to keep the likelihood of overfitting low by using different subsets of the training data and different subsets of features for training the different base classifiers. Thus only patterns truly present in the data would be detected consistently by a majority of the base classifiers and the majority votes turn out to be good indicators of class (Amaratunga, Cabrera, and Lee 2008, 2010–11).

Les forêts aléatoires originales, et qui sont aujourd'hui encore les plus utilisées, sont celles construites à partir de l'algorithme de Breiman. Le biais de sélection des variables décrit plus haut lors de la construction des arbres CART est transféré aux forêts aléatoires pour ce qui est de la mesure d'importance des variables. En effet, Carolin Strobl a montré dans son mémoire de maîtrise (Strobl 2004), sa thèse de doctorat (Strobl 2008) et dans plusieurs articles écrits en collaboration (Strobl, Boulesteix, and Augustin 2007) (Strobl et al. 2007) (Strobl et al. 2008), que cette mesure est également biaisée lorsque CART est utilisé comme algorithme de base pour construire les arbres.

Plusieurs algorithmes alternatifs de construction d'arbres « non-biaisés » ont été proposées comme nous l'avons vu dans la section précédente. Dans ce travail, nous étudierons l'alternative la plus utilisée (puisqu'implantée dans le logiciel R) : les forêts basées sur des arbres d'inférences conditionnelles (CTREE) (Hothorn, Hornik, and Zeileis 2006).

La règle qui détermine le point de coupure pour lequel les données sont séparées lors d'un nœud intermédiaire de chaque arbre est cruciale pour la performance de prédiction d'un arbre. Cependant, les résultats des forêts aléatoires sont beaucoup moins dépendants de la règle utilisée. Dans une forêt, les arbres ne sont pas élagués, il existe alors beaucoup plus de nœuds terminaux, mais ne souffrent pas de surapprentissage puisqu'ils sont construits à partir d'un nombre restreint de variables.

De plus, les arbres sont souvent utilisés pour décrire la pertinence de chaque variable explicative, dans un contexte d'inférence. Un arbre est facile à visualiser et à comprendre. On les utilise donc souvent pour illustrer le lien entre les variables explicatives et la variable à prédire. Les forêts sont plutôt utilisées pour prédire sans chercher à expliquer, puisqu'on ne peut pas les visualiser.

Chapitre 3
Présentation des ensembles de données et méthodologie

Ce chapitre vise à présenter : 1) de façon détaillée les caractéristiques a) des ensembles de données qui ont été utilisés; b) des simulations créées à partir de techniques utilisées précédemment dans la littérature; c) des jeux de données artificiels que nous avons créées afin d'évaluer plus précisément l'effet du bruit, du nombre de données comprises dans le jeu de données d'entraînement, de l'écart-type de la variable réponse, du nombre de variables sélectionnées aléatoirement pour construire les forêts 2) la méthode utilisée pour imputer les valeurs manquantes dans certains des ensembles de données; 3) les '*packages*' et les paramètres qui ont été utilisés.

Objectifs de recherche

Le but de cette recherche est d'évaluer la capacité de prédiction de deux algorithmes construisant des forêts aléatoires. Nous comparerons les forêts aléatoires 'classiques' tels que conçues par Breiman à une alternative : des forêts construites à partir d'arbres 'non-biaisés' (arbres d'inférence conditionnelles). Nous testerons également les arbres de décisions des méthodes CART et ctree pour chacun des jeux de données réels et que nous simulerons. Nous visons à évaluer quelle méthode obtient la meilleure performance prédictive pour ce qui est des forêts. Par les simulations, nous pourrons aussi vérifier si des éléments particuliers du jeu de donnée à l'étude améliorent la performance prédictive pour l'une des deux méthodes plus particulièrement.

Ensembles de données utilisés

La majorité des ensembles de données réels utilisés dans cette étude proviennent du package `mlbench` (Leisch and Dimitriadou 2015), d'autres ont été récupéré de UC Irvine Machine Learning Repository[5].

Boston Housing (BH)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). BH contient 506 observations et 14 variable. Aucune valeur n'est manquante. La variable-réponse 'medv' est continue et désigne : « median value of owner-occupied homes in USD 1000's ». Voici la description des 13 variables explicatives :

crim: per capita crime rate by town

z: proportion of residential land zoned for lots over 25,000 sq.ft

indus: proportion of non-retail business acres per town

chas: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

nox: nitric oxides concentration (parts per 10 million)

rm: average number of rooms per dwelling

age: proportion of owner-occupied units built prior to 1940

dis: weighted distances to five Boston employment centres

rad: index of accessibility to radial highways

tax: full-value property-tax rate per USD 10,000

ptratio: pupil-teacher ratio by town

b: $1000(B - 0.63)^2$ where B is the proportion of blacks by town

lstat: percentage of lower status of the population (Leisch and Dimitriadou 2015, 5).

[5] http://archive.ics.uci.edu/ml/ (page consultée le 18 juillet 2017).

Une des variables explicatives est binaire (*chas*) et les autres sont continues. Le même ensemble de données a aussi été utilisé dans : Breiman (1996a), Lim, Loh et Shih (2000) et Loh (2002).

Wisconsin Breast Cancer Database (BC)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). BC contient 699 observations et 10 variables (nous avons supprimé une onzième variable d'identification). Les 16 valeurs manquantes ont été imputées tel que mentionné plus haut. La variable-réponse 'Class' est catégorielle (2 catégories) et désigne le niveau de gravité du cancer: « *benigh* » ou « *malignant* ». Voici la description des neuf variables explicatives :

Cl.thickness: Clump Thickness

Cell.size: Uniformity of Cell Size

Cell.shape: Uniformity of Cell Shape

Marg.adhesion: Marginal Adhesion

Epith.c.size: Single Epithelial Cell Size

Bare.nuclei: Bare Nuclei

Bl.cromatin: Bland Chromatin

Normal.nucleoli: Normal Nucleoli

Mitoses: Mitoses (Leisch and Dimitriadou 2015, 6).

Les neuf variables explicatives sont ordinales (10 catégories : 1-10). Le même ensemble de données a aussi été utilisé dans : Breiman (1996b), Loh et Shih (1997), Lim, Loh et Shih (2000) et Breiman (2001).

Primate splice-junction gene sequences (DNA)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). DNA contient 3 186 observations et 181 variables. Il n'y a aucune valeur manquante. La variable-réponse 'Class' est catégorielle (3 catégories). Voici l'explication des variables:

> The data points are described by 180 indicator binary variables and the problem is to recognize the 3 classes (ei, ie, neither), i.e., the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). (Leisch and Dimitriadou 2015, 8).

Le même ensemble de données a aussi été utilisé dans : Michie et al. (1994), Breiman (1996b) et Lim, Loh et Shih (2000).

Glass Identification Database (Glass)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). *Glass* contient 214 observations et 10 variables. Il n'y a aucune valeur manquante. La variable-réponse 'Type' est catégorielle (7 catégories) et désigne un type de verre. Les neuf variables explicatives continues décrivent les composantes chimiques du verre :

> [,1] **RI** refractive index
>
> [,2] **Na** Sodium
>
> [,3] **Mg** Magnesium
>
> [,4] **Al** Aluminum
>
> [,5] **Si** Silicon
>
> [,6] **K** Potassium
>
> [,7] **Ca** Calcium
>
> [,8] **Ba** Barium
>
> [,9] **Fe** Iron (Leisch and Dimitriadou 2015, 9).

Le même ensemble de données a aussi été utilisé dans : Breiman (1996b) et Breiman (2001).

Johns Hopkins University Ionosphere Database (Ionosphere)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). *Ionosphere* contient 351 observations et 36 variables. Il n'y a aucune valeur manquante. La variable-réponse 'Class' est catégorielle (2 catégories) et désigne si les retours du radar sont «bons » ou « mauvais ». 32 variables explicatives sont continues et 2 sont binaires. Le même ensemble de données a aussi été utilisé dans : Breiman (1996b) et Breiman (2001).

Los Angeles Ozone Pollution Data, 1976 (Ozone)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). *Ozone* contient 366 observations (représentant un jour) et 13 variables. Les 203 valeurs manquantes ont été imputées tel qu'expliqué plus bas (section préparation des données). La valeur-réponse 'V4' est continue et désigne « the daily maximum one-hour-average ozone reading ». Les variables explicatives sont ainsi décrites :

1 Month: 1 = January, ..., 12 = December
2 Day of month
3 Day of week: 1 = Monday, ..., 7 = Sunday
4 Daily maximum one-hour-average ozone reading
5 500 millibar pressure height (m) measured at Vandenberg AFB
6 Wind speed (mph) at Los Angeles International Airport (LAX)
7 Humidity (%) at LAX
8 Temperature (degrees F) measured at Sandburg, CA
9 Temperature (degrees F) measured at El Monte, CA
10 Inversion base height (feet) at LAX
11 Pressure gradient (mm Hg) from LAX to Daggett, CA
12 Inversion base temperature (degrees F) at LAX
13 Visibility (miles) measured at LAX (Leisch and Dimitriadou 2015, 28).

Le même ensemble de données a aussi été utilisé dans Breiman (1996a).

Pima Indians Diabetes Database (PID)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). PID contient 768 observations et 9 variables. Il n'y a aucune valeur manquante. La variable-réponse 'diabetes' est catégorielle (2 catégories : 'neg' ou 'pos'). Les huit variables explicatives continues décrivent les caractéristiques d'une personne :

pregnant: Number of times pregnant

glucose: Plasma glucose concentration (glucose tolerance test)

pressure: Diastolic blood pressure (mm Hg)

triceps: Triceps skin fold thickness (mm)

insulin: 2-Hour serum insulin (mu U/ml)

mass: Body mass index (weight in kg/(height in m)\^2)

pedigree: Diabetes pedigree function

age: Age (years) (Leisch and Dimitriadou 2015, 9).

Le même ensemble de données a aussi été utilisé dans : Breiman (1996b), Lim, Loh et Shih (2000) et Breiman (2001).

Soybean Database (Soybean)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). Soybean contient 683 observations et 36 variables. Il y avait 2337 valeurs manquantes qui ont toutes été imputées tel que décrit plus bas. La variable-réponse 'Class' est catégorielle (19 catégories).

[,1] Class: the 19 classes

Les 35 variables explicatives catégorielles (numériques) sont les suivantes :

[,2] date: apr(0), may(1), june(2), july(3), aug(4), sept(5), oct(6).

[,3] plant.stand: normal(0), lt-normal(1).

[,4] precip: lt-norm(0), norm(1), gt-norm(2).

[,5] temp: lt-norm(0), norm(1), gt-norm(2).

[,6] hail: yes(0), no(1).

[,7] crop.hist: dif-lst-yr(0), s-l-y(1), s-l-2-y(2), s-l-7-y(3).

[,8] area.dam: scatter(0), low-area(1), upper-ar(2), whole-field(3).

[,9] sever: minor(0), pot-severe(1), severe(2).

[,10] seed.tmt: none(0), fungicide(1), other(2).

[,11] germ: 90-100%(0), 80-89%(1), lt-80%(2).

[,12] plant.growth: norm(0), abnorm(1).

[,13] leaves: norm(0), abnorm(1).

[,14] leaf.halo: absent(0), yellow-halos(1), no-yellow-halos(2).

[,15] leaf.marg: w-s-marg(0), no-w-s-marg(1), dna(2).

[,16] leaf.size: lt-1/8(0), gt-1/8(1), dna(2).

[,17] leaf.shread absent(0), present(1).

[,18] leaf.malf: absent(0), present(1).

[,19] leaf.mild: absent(0), upper-surf(1), lower-surf(2).

[,20] stem: norm(0), abnorm(1).

[,21] lodging: yes(0), no(1).

[,22] stem.cankers: absent(0), below-soil(1), above-s(2), ab-sec-nde(3).

[,23] canker.lesion: dna(0), brown(1), dk-brown-blk(2), tan(3).

[,24] fruiting.bodies: absent(0), present(1).

[,25] ext.decay: absent(0), firm-and-dry(1), watery(2).

[,26] mycelium: absent(0), present(1).

[,27] int.discolor: none(0), brown(1), black(2).

[,28] sclerotia: absent(0),present(1).

[,29] fruit.pods: norm(0),diseased(1),few-present(2),dna(3).

[,30] fruit.spots: absent(0),col(1),br-w/blk-speck(2),distort(3),dna(4).

[,31] seed norm:(0),abnorm(1)

[,32] mold.growth: absent(0),present(1).

[,33] seed.discolor: absent(0),present(1).

[,34] seed.size: norm(0),lt-norm(1).

[,35] shriveling: absent(0),present(1).

[,36] roots: norm(0),rotted(1),galls-cysts(2). (Leisch and Dimitriadou 2015, 36).

Le même ensemble de données a aussi été utilisé dans : Breiman (1996b) et Breiman (2001).

Letter Image Recognition Dataset (LR)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). Soybean contient 20000 observations et 17 variables. Il n'y avait pas de valeurs manquantes. La variable-réponse 'lettr' est catégorielle (26 catégories) et correspond aux 26 lettres de l'alphabet en majuscule.

[,1] : lettr : capital letter

Les 16 variables explicatives continues sont les suivantes :

[,2] x.box: horizontal position of box

[,3] y.box: vertical position of box

[,4] width: width of box

[,5] high: height of box

[,6] onpix: total number of on pixels

[,7] x.bar: mean x of on pixels in box

[,8] y.bar: mean y of on pixels in box

[,9] x2bar: mean x variance

[,10] y2bar: mean y variance

[,11] xybar: mean x y correlation

[,12] x2ybr: mean of x^2y

[,13] xy2br: mean of xy^2

[,14] x.ege: mean edge count left to right

[,15] xegvy: correlation of x.ege with y

[,16] y.ege: mean edge count bottom to top

[,17] yegvx: correlation of y.ege with x (Leisch and Dimitriadou 2015, 13).

Le même ensemble de données a aussi été utilisé dans Breiman (2001).

Landsat Multi-Spectral Scanner Image Data (Satellite)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). *Sattelite* contient 6435 observations et 37 variables. Il n'y avait pas de valeurs manquantes. La variable-réponse 'classes' est catégorielle (6 catégories). Voici la description de l'ensemble de données :

> One frame of Landsat MSS imagery consists of four digital images of the same scene in different spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infra-red. Each pixel is a 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution of a pixel is about 80m x 80m. Each image contains 2340 x 3380 such pixels.
>
> The database is a (tiny) sub-area of a scene, consisting of 82 x 100 pixels. Each line of data corresponds to a 3x3 square neighbourhood of pixels completely contained within the 82x100 subarea. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighbourhood and a number indicating the classification label of the central pixel.
>
> The classes are:
>
> red soil
> cotton crop
> grey soil
> damp grey soil
> soil with vegetation stubble
> very damp grey soil (Leisch and Dimitriadou 2015, 31).

Les 36 variables explicatives sont continues. Le même ensemble de données a aussi été utilisé dans Mitchel et al. (1994), Breiman (1996b), Lim, Loh et Shih (2000) et Breiman (2001).

Teaching Assistant Evaluation (TAE)

Cet ensemble de données est tiré de *UC Irive Machine Learning Repository*. TAE contient 151 observations et 6 variables. Il n'y avait pas de valeurs manquantes. La variable-réponse est catégorielle (3 catégories). Voici la description de l'ensemble de données :

> The data consist of evaluations of teaching performance over three regular semesters and two summer semesters of 151 teaching assistant (TA) assignments at the Statistics Department of the University of Wisconsin-Madison. The scores were divided into 3 roughly equal-sized categories ("low", "medium", and "high") to form the class variable.
>
> Attribute Information:
>
> 1. Whether of not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker
>
> 2. Course instructor (categorical, 25 categories)
>
> 3. Course (categorical, 26 categories)
>
> 4. Summer or regular semester (binary) 1=Summer, 2=Regular
>
> 5. Class size (numerical)
>
> 6. Class attribute (categorical) 1=Low, 2=Medium, 3=High (Lichman 2013).

Le même ensemble de données a aussi été utilisé dans Loh et Shih (1997), Lim, Loh et Shih (2000) et dans Loh (Loh 2002).

Contraceptive Method Choice (CMC)

Cet ensemble de données est tiré de *UC Irive Machine Learning Repository*. CMC contient 1473 observations et 10 variables. Il n'y avait pas de valeurs manquantes. La variable-réponse est catégorielle (3 catégories). Voici la description de l'ensemble de données :

> This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-

term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.

Attribute Information:
1. Wife's age (numerical)
2. Wife's education (categorical) 1=low, 2, 3, 4=high
3. Husband's education (categorical) 1=low, 2, 3, 4=high
4. Number of children ever born (numerical)
5. Wife's religion (binary) 0=Non-Islam, 1=Islam
6. Wife's now working? (binary) 0=Yes, 1=No
7. Husband's occupation (categorical) 1, 2, 3, 4
8. Standard-of-living index (categorical) 1=low, 2, 3, 4=high
9. Media exposure (binary) 0=Good, 1=Not good
10. Contraceptive method used (class attribute) 1=No-use, 2=Long-term, 3=Short-term (Lim 1997).

Le même ensemble de données a aussi été utilisé dans Lim, Loh et Shih (2000).

Liver Disorders Data Set (LD)

Cet ensemble de données est tiré de *UC Irive Machine Learning Repository.* LD contient 345 observations et 6 variables (nous avons éliminé la variable '*selector*' qui peut être utilisée pour séparer l'ensemble de données). Il n'y avait pas de valeurs manquantes. La variable-réponse est continue et désigne le nombre de consommation consommé par semaine en moyenne. Voici la description des 5 variables explicatives continues :

The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the bupa.data file constitutes the record of a single male individual.

Attribute information:
1. **mcv:** mean corpuscular volume
2. **alkphos:** alkaline phosphotase
3. **sgpt:** alamine aminotransferase
4. **sgot:** aspartate aminotransferase
5. **gammagt:** gamma-glutamyl transpeptidase
6. **drinks**: number of half-pint equivalents of alcoholic beverages drunk per day (Forsyth and BUPA Medical Research Ltd 1990)

Le même ensemble de données a aussi été utilisé dans Lim, Loh et Shih (2000) et par Breiman (2001).

***Analysis of Attitudes towards Workplace Smoking Restrictions Data Set* (SMO)**

Cet ensemble de données provient du chapitre 13 de *StatLib : Case Studies in Biometry* qui est tiré de (Lange, Ryan, and Billard 1994) :

> Data from the Attitudes Toward Smoking Legislation Survey - Metropolitan Toronto 1988, which was funded by NHRDP (Health and Welfare Canada), were collected by the Institute for Social Research at York University for Dr. Linda Pederson and Dr. Shelley Bull. (Shelley B. Bull 1994)

SMO contient 2854 observations et 13 variables (nous avons éliminé une variable d'identification). Il n'y avait pas de valeurs manquantes. La variable-réponse est catégorielle (trois catégories). Voici la description des variables:

SHORT DESCRIPTION	NAME	DEFINITION AND CODING
Outcome	y	Attitude toward smoking in the workplace. Smoking should be: (1 = prohibited, 2 = restricted, 0 = unrestricted)
Weight	w	Sampling/post-stratification weight (ranges from 0.305 to 4.494)
Time	x1	Time of survey relative to implementation of the by-law on March 1, 1998 (1=post, 0=pre)
Work	x2	Place of work indicator 1 with City of Toronto as baseline (1 = outside City of Toronto, 0 = otherwise)
	x3	Place of work indicator 2 with City of Toronto as baseline (1 = not outside the home, 0 = otherwise)
Residence	z1	Place of residence (1 = City of Toronto, 0 = other Metro Toronto)
Smoking	z2	Smoking status indicator 1 with those who have never smoked as the baseline (1= current smoker, 0=otherwise)
	z3	Smoking status indicator 2 with never as the baseline (1= quit <=6 months ago, 0=otherwise)
	z4	Smoking status indicator 3 with never as the baseline (1= quit > 6 months ago, 0=otherwise)
	z5	Smoking status indicator 4 with quit > 12 months as the baseline (1= quit 6-12 months, 0=otherwise)
Knowledge	z6	Knowledge of health effects of environmental tobacco smoke (score, ranges from 0 to 12)
Sex	z7	Sex of respondent (1 = male, 0 = female)
Age	z8	Age of respondent ((age in years - 50)/10)
Education	z9	Level of education (-2 = elementary, -1 = some high school, 0 = high or trade school, 1 = college or some university, 2 = university degree)

Le même ensemble de données a aussi été utilisé dans Lim, Loh et Shih (2000).

***Vehicle Silhouettes* (VH)**

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). VH contient 846 observations et 19 variables. Il n'y avait aucune valeur manquante. La variable-réponse 'Class' est catégorielle (4 catégories) et désigne un type de véhicule : « a double decker bus, Cheverolet van, Saab 9000 and an Opel Manta 400 ». Les 18 variables explicatives continues sont les suivantes :

[,1] Comp: Compactness

[,2] Circ: Circularity

[,3] D.Circ: Distance Circularity

[,4] Rad.Ra: Radius ratio

[,5] Pr.Axis.Ra: pr.axis aspect ratio

[,6] Max.L.Ra: max.length aspect ratio

[,7] Scat.Ra: scatter ratio

[,8] Elong: elongatedness

[,9] Pr.Axis.Rect: pr.axis rectangularity

[,10] Max.L.Rect: max.length rectangularity

[,11] Sc.Var.Maxis: scaled variance along major axis

[,12] Sc.Var.maxis: scaled variance along minor axis

[,13] Ra.Gyr: scaled radius of gyration

[,14] Skew.Maxis: skewness about major axis

[,15] Skew.maxis: skewness about minor axis

[,16] Kurt.maxis: kurtosis about minor axis

[,17] Kurt.Maxis: kurtosis about major axis

[,18] Holl.Ra: hollows ratio. (Leisch and Dimitriadou 2015, 38).

Le même ensemble de données a aussi été utilisé dans : Mitchel et al. (1994), Lim, Loh et Shih (2000) et Breiman (2001).

HouseVote84 : United States Congressional Voting Records 1984 (HV)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). HV contient 435 observations et 17 variables. Il y avait 392 valeurs manquantes qui ont toutes été imputées tel que décrit plus bas. La variable-réponse 'Class' est catégorielle (2 catégories : '*democrat*'; '*republican*') :

1 Class Name: 2 (democrat, republican)

Les 16 variables explicatives binaires ('y', 'n') sont les suivantes :

2 handicapped-infants: 2 (y,n)

3 water-project-cost-sharing: 2 (y,n)

4 adoption-of-the-budget-resolution: 2 (y,n)

5 physician-fee-freeze: 2 (y,n)

6 el-salvador-aid: 2 (y,n)

7 religious-groups-in-schools: 2 (y,n)

8 anti-satellite-test-ban: 2 (y,n)

9 aid-to-nicaraguan-contras: 2 (y,n)

10 mx-missile: 2 (y,n)

11 immigration: 2 (y,n)

12 synfuels-corporation-cutback: 2 (y,n)

13 education-spending: 2 (y,n)

14 superfund-right-to-sue: 2 (y,n)

15 crime: 2 (y,n)

16 duty-free-exports: 2 (y,n)

17 export-administration-act-south-africa: 2 (y,n)
(Leisch and Dimitriadou 2015, 10).

Le même ensemble de données a aussi été utilisé dans : Lim, Loh et Shih (2000) et Breiman (2001).

Sonar, Mines vs. Rocks (Sonar)

Cet ensemble de données est tiré du package `mlbench` (Leisch and Dimitriadou 2015). Sonar contient 208 observations et 61 variables. Il n'y avait pas de valeur manquante. La variable-réponse 'Class' est catégorielle (2 catégories : '*M*'; '*R*') : "The label associated with each record contains the letter "R" if the object is a rock and "M" if it is a mine (metal cylinder)" (Leisch and Dimitriadou 2015, 35). Les 60 variables explicatives continues sont ainsi décrites:

> Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp. (Leisch and Dimitriadou 2015, 35).

Le même ensemble de données a aussi été utilisé dans Breiman (2001).

***Statlog (German Credit Data) Data Set* (GC)**

Cet ensemble de données est tiré de *UC Irive Machine Learning Repository.* La version originale (et non numérique) de GC contient 1000 observations et 21 variables. Il n'y avait pas de valeurs manquantes. La variable-réponse est catégorielle (2 catégories : Good/Bad) et représente le risque de crédit d'une personne. Voici la description des 20 variables explicatives (7 continues et 13 catégorielles) :

> 1: (qualitative) Status of existing checking account:
>
> A11 : ... < 0 DM
> A12 : 0 <= ... < 200 DM
> A13 : ... >= 200 DM /salary assignments for at least 1 year
> A14 : no checking account
>
> 2: (numerical) Duration in month
>
> 3: (qualitative) Credit history:
>
> A30 : no credits taken/ all credits paid back duly
> A31 : all credits at this bank paid back duly
> A32 : existing credits paid back duly till now
> A33 : delay in paying off in the past
> A34 : critical account/other credits existing (not at this bank)

4: (qualitative) Purpose
A40 : car (new)
A41 : car (used)
A42 : furniture/equipment
A43 : radio/television
A44 : domestic appliances
A45 : repairs
A46 : education
A47 : (vacation - does not exist?)
A48 : retraining
A49 : business
A410 : others

5: (numerical) Credit amount

6: (qualitative) Savings account/bonds

A61 : ... < 100 DM
A62 : 100 <= ... < 500 DM
A63 : 500 <= ... < 1000 DM
A64 : .. >= 1000 DM
A65 : unknown/ no savings account

7: (qualitative) Present employment since:

A71 : unemployed
A72 : ... < 1 year
A73 : 1 <= ... < 4 years
A74 : 4 <= ... < 7 years
A75 : .. >= 7 years

8: (numerical) Installment rate in percentage of disposable income

9: (qualitative) Personal status and sex

A91 : male : divorced/separated
A92 : female : divorced/separated/married
A93 : male : single
A94 : male : married/widowed
A95 : female : single

10: (qualitative) Other debtors / guarantors

A101 : none
A102 : co-applicant
A103 : guarantor

11: (numerical) Present residence since

12: (qualitative) Property

A121 : real estate
A122 : if not A121 : building society savings agreement/life insurance
A123 : if not A121/A122 : car or other, not in attribute 6
A124 : unknown / no property

13: (numerical) Age in years

14: (qualitative) Other installment plans

A141 : bank
A142 : stores
A143 : none

15: (qualitative) Housing

A151 : rent
A152 : own
A153 : for free

16: (numerical) Number of existing credits at this bank

17: (qualitative) Job

A171 : unemployed/ unskilled - non-resident
A172 : unskilled - resident
A173 : skilled employee / official
A174 : management/ self-employed/highly qualified employee/ officer

18: (numerical) Number of people being liable to provide maintenance for

19: (qualitative) Telephone

A191 : none
A192 : yes, registered under the customers name

20: (qualitative) foreign worker

A201 : yes
A202 : no

(Hofman, Pr. Dr. Hans and Institut für Statistik und Ökonometrie, Universität Hamburg 1994)

Une version numérique de ce même ensemble de données, dans laquelle les variables explicatives catégorielles ordinales sont considérées continues et les variables catégorielles nominales transformées en indicateurs binaires a été utilisée par Statlog (Michie, Spiegelhalter, and Taylor 1994). Le même ensemble de données a aussi été utilisé dans Breiman (2001).

***Ecoli Data Set* (aussi nommé *Protein Localization Sites*)**

Cet ensemble de données est tiré de UC Irive Machine Learning Repository. Ecoli contient 336 observations et 8 variables (nous avons enlevé une variable d'identification). Il n'y avait pas de valeurs manquantes. La variable-réponse est catégorielle (8 catégories) et représente des lieux de localisation des protéines. Il y a un débalancement dans la répartition des données, voici le nombre d'observation dans chacune des 8 catégories :

cp (cytoplasm) : 143
im (inner membrane without signal sequence): 77
pp (perisplasm): 52
imU (inner membrane, uncleavable signal sequence): 35
om (outer membrane) : 20

omL (outer membrane lipoprotein) : 5
imL (inner membrane lipoprotein) : 2
imS (inner membrane, cleavable signal sequence): 2

Voici la description des 7 variables explicatives continues :

mcg: McGeoch's method for signal sequence recognition.
gvh: von Heijne's method for signal sequence recognition.
lip: von Heijne's Signal Peptidase II consensus sequence score. Binary attribute.
chg: Presence of charge on N-terminus of predicted lipoproteins. Binary attribute.
aac: score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins.
alm1: score of the ALOM membrane spanning region prediction program.
alm2: score of ALOM program after excluding putative cleavable signal regions from the sequence. (Horton, Paul and Nakai, Kenta 1994)

Le même ensemble de donnée a aussi été utilisé dans Breiman (2001).

Baseball

Cet ensemble de données est tiré des archives du *Journal of Statistics Education* (JSE). Baseball contient 337 observations et 17 variables :

The data set contains salary information for 337 Major League Baseball (MLB) players who are not pitchers and played at least one game during both the 1991 and 1992 seasons. The purpose of the study is to determine whether a baseball player's salary is a reflection of his offensive performance. For each player, the salary from the 1992 season along with 12 offensive statistics from the 1991 season were collected. In addition to these variables, there are 4 indicator variables which identify free agency and eligibility for arbitration (NC State University and Watnik, Mitchell R. 1996).

Il n'y avait pas de valeurs manquantes. La variable-réponse (Y) est continue et représente le salaire en milliers de dollars d'un joueur. Voici la description des 16 variables explicatives (12 continues et 4 binaires) :

X1	Batting average
X2	On-base percentage
X3	Number of runs
X4	Number of hits
X5	Number of doubles
X6	Number of triples
X7	Number of home runs

X8 Number of runs batted in

X9 Number of walks

X10 Number of strike-outs

X11 Number of stolen bases

X12 Number of errors

X13 Indicator of "free agency eligibility"

X14 Indicator of "free agent in 1991/2"

X15 Indicator of "arbitration eligibility"

X16 Indicator of "arbitration in 1991/2" (NC State University and Watnik, Mitchell R. 1996)

Le même ensemble de donnée a aussi été utilisé dans Loh (2002).

Ensembles de données créés par simulations inspirées de la littérature

Six ensembles de données supplémentaires ont été créés à partir de simulations. Ils ont été choisis pour avoir été utilisés préalablement dans la littérature comparant la performance de forêts aléatoires, de différents types d'arbres de décision entre eux ou encore à celle d'autres modèles de prédiction. Les six ensemble de données ont été créé dans R à partir de fonctions incluses dans le package `mlbench` (Leisch and Dimitriadou 2015). Pour chacune des six simulations, nous avons utilisé le paramètre `set.seed(4756)` et le paramètre déterminant le nombre d'observation : `n=1800`.

Waveform Database Generator (wf)

Voici la description de l'ensemble de données simulé par *Waveform Database Generator :* "The generated data set consists of 21 attributes with continuous values and a variable showing the 3 classes (33% for each of 3 classes). Each class is generated from a combination of 2 of 3 "base" waves." (Leisch and Dimitriadou 2015, 26)

Le même type de simulation avait aussi été créée par : Breiman (1996b) (également avec 1800 observations) et (2001), par Loh et Shih (1997), par Lim, Loh et Shih (2000).

Benchmark Problem Friedman 1 (f1)

Voici la description de l'ensemble de données simulé par *Benchmark Problem Friedman 1 :*

> The regression problem Friedman 1 as described in Friedman (1991) and Breiman (1996). Inputs are 10 independent variables uniformly distributed on the interval [0; 1], only 5 out of these 10 are actually used. Outputs are created according to the formula
>
> $$y = 10\sin(\pi x1x2) + 20(x3 - 0.5)^2 + 10x4 + 5x5 + e$$
>
> where e is N(0,sd) (Leisch and Dimitriadou 2015, 17).

Nous avons utilisé 1 pour le paramètre indiquant l'écart-type (`sd`).

Le même type de simulation avait d'abord été utilisé par Friedman (1991), Michie et al. (1994) et Breiman (1996b) avec le même écart-type qui est aujourd'hui le paramètre par défaut du package.

Benchmark Problem Friedman 2 (f2)

Voici la description de l'ensemble de données simulé par *Benchmark Problem Friedman 2* :

> The regression problem Friedman 2 as described in Friedman (1991) and Breiman (1996). Inputs are 4 independent variables uniformly distrtibuted (*sic.*) over the ranges
>
> $$0 \leq x1 \leq 100$$
> $$40\pi \leq x2 \leq 560\pi$$
> $$0 \leq x3 \leq 1$$
> $$1 \leq x4 \leq 11$$
>
> The outputs are created according to the formula
>
> $$y = (x1^2 + (x2x3 - (1/(x2x4)))\text{^}2)^{0.5} + e$$
>
> where e is N(0,sd).

Nous avons utilisé 125 pour le paramètre indiquant l'écart-type (`sd`) : "The default value of 125 gives a signal to noise ratio (i.e., the ratio of the standard deviations) of 3:1.

Thus, the variance of the function itself (without noise) accounts for 90% of the total variance" (Leisch and Dimitriadou 2015, 18).

Le même type de simulation avait d'abord été utilisé par Friedman (1991) et par Breiman (1996b).

Benchmark Problem Friedman 3 (f3)

Voici la description de l'ensemble de données simulé par *Benchmark Problem Friedman 3* :

> The regression problem Friedman 3 as described in Friedman (1991) and Breiman (1996). Inputs are 4 independent variables uniformly distrtibuted over the ranges
>
> $$0 \leq x1 \leq 100$$
> $$40\pi \leq x2 \leq 560\pi$$
> $$0 \leq x3 \leq 1$$
> $$1 \leq x4 \leq 11$$
>
> The outputs are created according to the formula
>
> $$y = atan((x2x3 - (1/(x2x4)))/x1) + e$$
>
> where e is N(0,sd).

Nous avons utilisé 0.1 pour le paramètre indiquant l'écart-type (`sd`) : "The default value of 0.1 gives a signal to noise ratio (i.e., the ratio of the standard deviations) of 3:1. Thus, the variance of the function itself (without noise) accounts for 90% of the total variance" (Leisch and Dimitriadou 2015, 19).

Le même type de simulation avait d'abord été utilisé par Friedman (1991) et par Breiman (1996b).

Twonorm Benchmark Problem (t2n)

Voici la description de l'ensemble de données simulé par t2n : "The inputs of the twonorm problem are points from two Gaussian distributions with unit covariance matrix. Class 1 is multivariate normal with mean $(a, a, \ldots, a)$ and class 2 with mean $(-a, -a, \ldots, -a), a = 2/d^{0.5}$ (Leisch and Dimitriadou 2015, 25).

Nous avons utilisé 20 pour le paramètre indiquant le nombre de dimensions `(d)`.

Le même type de simulation avait d'abord été utilisé par Breiman (2001).

Threenorm Benchmark Problem (t3n)

Voici la description de l'ensemble de données simulé par t3n :

> The inputs of the threenorm problem are points from two Gaussian distributions with unit covariance matrix. Class 1 is drawn with equal probability from a unit multivariate normal with mean $(a, a, \ldots, a)$ and from a unit multivariate normal with mean $(-a, -a, \ldots, -a)$. Class 2 is drawn from a multivariate normal with mean at $(a, -a, a, \ldots, -a), a = 2/d^{0.5}$ (Leisch and Dimitriadou 2015, 24).

Nous n'avons pas modifié le paramètre par défaut indiquant le nombre de dimensions(d) qui est 20.

Le même type de simulation avait d'abord été utilisé par Breiman (2001).

Ringnorm Benchmark Problem (rn)

Voici la description de l'ensemble de données simulé par rn : "The inputs of the ringnorm problem are points from two Gaussian distributions. Class 1 is multivariate normal with mean 0 and covariance 4 times the identity matrix. Class 2 has unit covariance and mean $(a, a, \ldots, a), a = d^{-0.5}$ (Leisch and Dimitriadou 2015, 21)".

Nous avons utilisé 20 pour le paramètre indiquant le nombre de dimensions(d).

Le même type de simulation avait d'abord été utilisé par Breiman (2001).

Jeux de données artificiels construits selon des scénarios avec contrôle de paramètres précis

Les différents jeux de données seront générés en contrôlant quatre paramètres différents :

1) Le nombre d'observations contenues dans le jeu de données utilisé pour l'apprentissage du modèle. Ce nombre variera entre 200 et 5000.

2) Le scénario utilisé. La génération des données sera basée sur trois scénarios distincts variant le type des variables explicatives utilisées. Dans le scénario 1, toutes les variables seront continues. Dans le scénario 2, seules des variables explicatives catégorielles seront inclues. Dans le scénario 3, un mélange de variables explicatives continues et catégorielles seront utilisées.

3) L'écart-type de l'erreur de la variable-réponse. Pour chacune des séries, un jeu de donnée aura une variable-réponse qui variera selon un écart-type de 1, un autre de 3.

4) Le bruit. Une série de jeux de données sera construite avec ajout de 20 variables supplémentaires (10 continues et 10 catégorielles) n'ajoutant que du bruit dans les données (sans lien avec la variable-réponse). Il y aura donc présence ou non de bruit.

Au total, cette génération résultera en 12 jeux de données artificiels (3 scénarios x 2 options de bruits x 2 options d'écart-types). Le nombre d'observations total dans les jeux de données varieront en fonction du nombre de données d'entraînement (entre 200 et 5000). Le nombre de données qui seront utilisées pour le test est toujours de 5000.

scénario	bruit	Nombre total de X continus	Nombre total de X catégoriels	Nombre de X continus liés à Y	Nombre de X catégoriels liés à Y
1	0	6	6	6	0
1	1	16	16	6	0
2	0	6	6	0	6
2	1	16	16	0	6
3	0	6	6	6	6
3	1	16	16	6	6

La formule créant les variables-réponse est la suivante :

$$y = me + \sigma * \varepsilon$$
$$\varepsilon \sim N(0,1)$$

où me est définit selon les 3 scénarios (voir plus bas). Pour construire les variables prédictives X1 à X12, on utilise la loi normale multivariée avec moyenne 0 et une matrice de covariance 12 x 12 où la diagonale est de 1 et la corrélation est de rho (ρ) = 0.3 :

$$\begin{bmatrix} 1 & \cdots & \rho \\ \vdots & \ddots & \vdots \\ \rho & \cdots & 1 \end{bmatrix}.$$

À partir des variables ainsi créées, X1 àX6 restent continues. On crée ensuite des variables catégorielles pour X7 à X12 selon :

$\boldsymbol{Pour\ i = 7, 8}$:
$si\ Xi \leq 0 \ \Rightarrow Xi =$ "V0"
$if\ Xi > 0 \ \Rightarrow Xi =$ "V1"

$\boldsymbol{Pour\ i = 9, 10}$:
$si\ Xi \leq -0.5 \ \Rightarrow Xi =$ "V0"
$si\ -0.5 < Xi < 0.5 \ \Rightarrow Xi =$ "V1"
$si\ Xi \geq 0.5 \ \Rightarrow \ Xi =$ "V2"

$\boldsymbol{Pour\ i = 11, 12}$:
$si\ Xi \leq -1 \ \Rightarrow \ Xi =$ "V0"
$si\ -1 < Xi < 0 \ \Rightarrow \ Xi =$ "V1"
$si\ 0 \leq Xi < 1 \ \Rightarrow \ Xi =$ "V2"
$si\ Xi \geq 1 \ \Rightarrow Xi =$ "V3"

Ainsi, X7 et X8 sont binaires, X9 et X10 prennent 3 valeurs, et X11 et X12 prennent 4 valeurs. La moyenne de la variance réponse, me, est générée de la manière suivante.

Pour scénario = 1 : $me =$

$$\frac{X_1-\overline{X_1}}{\sigma_{X_1}} + \frac{X_2-\overline{X_2}}{\sigma_{X_2}} + \frac{{X_3}^2-\overline{{X_3}^2}}{\sigma_{{X_3}^2}} + \frac{{X_4}^2-\overline{{X_4}^2}}{\sigma_{{X_4}^2}} + \frac{{X_5}^3-\overline{{X_5}^3}}{\sigma_{{X_5}^3}} + \frac{{X_6}^3-\overline{{X_6}^3}}{\sigma_{{X_6}^3}} + \frac{X_1X_2-\overline{X_1X_2}}{\sigma_{X_1X_2}} + \frac{X_2X_5-\overline{X_2X_5}}{\sigma_{X_2X_5}}$$

Pour scénario = 2 : $me =$

$$I_{\{X_7=\text{"V0"}\}} + I_{\{X_8=\text{"V1"}\}} + I_{\{X_9=\text{"V0"}\}} + I_{\{X_{10}=\text{"V0"}\}} + I_{\{X_{10}=\text{"V3"}\}} + I_{\{X_{11}=\text{"V1"}\}} + I_{\{X_{12}=\text{"V0"}\}} + I_{\{X_{12}=\text{"V1"}\}} + I_{\{X_7=\text{"V0" \& } X_9=\text{"V0"}\}} + I_{\{X_9=\text{"V2"}\}} + I_{\{X_9=\text{"V0" \& } X_{11}=\text{"V1" \& } X_{11}=\text{"V3"}\}}$$

Pour scénario = 3 : $me =$

[idem que scénario 1] +

[idem que scénario 2] +

$$\left(\frac{X_1-\overline{X_1}}{\sigma_{X_1}}\right) * I_{\{X_7=\text{"V0"}\}} + \left(\frac{{X_3}^2-\overline{{X_3}^2}}{\sigma_{{X_3}^2}}\right) * I_{\{X_{10}=\text{"V0"}\}} + I_{\{X_{10}=\text{"V3"}\}}$$

Pour les situations avec présences de variables explicatives supplémentaires (noise = 1**),** elles sont générées ainsi.

Si noise = 1:

$\boldsymbol{Pour\ i = 13:22}$:

$Xi \sim N(0,1)$

$\boldsymbol{Pour\ i = 23:32}$:

$Xi \sim Bin(2, p = 0.5)$

Ainsi, pour le scénario 1, seulement des variables explicatives continues sont liées à la variable réponse. Pour le scénario 2, seulement des variables explicatives catégorielles sont liées à la variable réponse. Finalement, pour le scénario 3, un mélange de variables continues et catégorielles sont liées à la variable réponse. Le code utilisé en R pour générer les données est présenté en annexe 1.

Préparation des données

Les ensembles de données réels qui nécessitaient des manipulations ou modifications ont été préparés avant que les deux algorithmes soient testés. Peu de modifications ont été apportées outre le retrait des variables d'identification (pour BC et SMO) et l'imputation des valeurs manquantes.

Imputation des valeurs manquantes

Parmi les ensembles de données réels, quatre contenaient des valeurs manquantes : BC, Ozone, Soybean et HV. Nous avons dans ces cas utilisé la fonction `complete` du package `mice` : Multivariate Imputation by Chained Equations (van Buuren 2017) pour imputer les valeurs manquantes à partir des autres variables explicatives.

Paramètres et packages utilisés

`rpart::rpart`

Nous avons utilisé la version 4.1-12 du package `rpart` : Recursive Partitioning and Regression Rrees (Thernau, Atkinson, and Ripley 2018), mis à jour en janvier 2018,

pour construire les arbres classiques CART. "*Description*. Recursive partitioning for classification, regression and survival trees. An implementation of most of the functionality of the 1984 book by Breiman, Friedman, Olshen and Stone" (Thernau, Atkinson, and Ripley 2018, 1).

party::ctree

Nous avons utilisé la version 1.2-4 du package `party`: A Laboratory for Recursive Partytioning (Hothorn et al. 2016), mis à jour en décembre 2017, pour construire les arbres 'non-biaisés' d'inférence conditionnelle.

> *Description*. A computational toolbox for recursive partitioning. The core of the package is ctree(), an implementation of conditional inference trees which embed tree-structured regression models into a well defined theory of conditional inference procedures. This non-parametric class of regression trees is applicable to all kinds of regression problems, including nominal, ordinal, numeric, censored as well as multivariate response variables and arbitrary measurement scales of the covariates (Hothorn and Zeileis 2015, 1).

randomForestSRC::rfsrc

Nous avons utilisé la version 2.4.2 du package `randomForestSRC` : Random Forests for Survival, Regression and Classification (Ishwaran and Kogalur 2017), mis à jour en mars 2017, pour construire les forêts 'classiques' développées à partir de l'algorithme CART.

> `rfsrc`: *Description.* A random forest (Breiman, 2001) is grown using user supplied training data. Applies when the response (outcome) is numeric, categorical (factor), or right-censored (including competing risk), and yields regression, classification, and survival forests, respectively. The resulting forest, informally referred to as a RF-SRC object, contains many useful values which can be directly extracted by the user and/or parsed using additional functions (see the examples below). This is the main entry point to the randomForestSRC package (Ishwaran and Kogalur 2017, 41).

Partykit::cforest

Nous avons utilisé la version 1.1-1 du package `partykit` : A Toolkit for Recursive Partytioning (Hothorn and Zeileis 2015), mis à jour en septembre 2016, pour construire les forêts 'non-biaisées' développées à partir de l'algorithme CTREE.

"`cforest`: An implementation of the random forest and bagging ensemble algorithms utilizing conditional inference trees as base learners" (Hothorn and Zeileis 2015, 2).

Pour tous les ensembles de données réels et des simulations inspirées de la littérature, nous avons laissé le paramètre sélectionnant le nombre de variables présélectionnées à chaque nœud (`mtry`) par défaut. Nous avons tenté de produire une forêt avec 500 arbres (`ntree`) pour chaque ensemble de données et compiler les résultats par validation-croisée à 10 groupes. Cependant, pour certain des ensembles de données, le package `partykit` n'arrivait pas à compiler la commande (maximum de mémoire atteinte). Nous avons donc, dans ces cas, réduit le nombre d'arbres ou le nombre de groupes pour les deux algorithmes. Nous indiquerons le nombre d'arbres et de groupes dans le tableau récapitulatif des ensembles de données utilisés. Il faut ici mentionner qu'un avantage notable de `randomForestSRC` sur `partykit` est qu'il est nettement plus rapide et permet d'utiliser un plus grand nombre d'arbres sans obstruer la mémoire maximale de l'ordinateur. Cela provient du fait que `randomForestSRC` est codé en C tandis que `partykit` est codé entièrement en R.

Chapitre 4
Résultats et comparaison des deux algorithmes

Méthode d'évaluation de la performance pour tous les jeux de données réels ou les simulations inspirées de la littérature

Nous souhaitions utiliser la validation croisée avec 10 groupes mutuellement exclusifs pour évaluer la performance de prédiction des arbres et des forêts construites à partir de 500 arbres. Cependant, pour les forêts, certains ensembles de données demandaient trop de mémoire pour que l'algorithme `partykit::cforest` arrive à produire un résultat. Dans le cas de `randomForestSRC::rfsrc` nous aurions pu utiliser 500 arbres et 10 groupes pour tous les ensembles de données sans que cela ne pose un problème. Dans les cas problématiques nous avons réduit le nombre d'arbres, puis le nombre de groupes pour la validation croisée, afin de pouvoir effectuer la comparaison de la capacité prédictive tout en gardant les mêmes paramètres pour chacun des deux algorithmes. Le nombre d'arbres utilisés et de groupes pour la validation-croisée est donc indiqué dans les tableaux de comparaison qui suivent.

Dans le cas des ensembles de données avec une variable-réponse catégorielle ou binaire, nous avons mesuré le taux de bonnes classifications pour chacun des groupes.

Dans le cas des ensembles de données avec variable-réponse continue, nous avons plutôt mesuré la distance moyenne entre le résultat et la valeur réelle élevée au carré.

Pour déterminer l'algorithme offrant la meilleure performance prédictive, nous avons compilé la moyenne du taux de bonnes classification ou d'erreur de prédiction pour les dix groupes.

Tableaux des résultats

4.1. Tableau comparatif du taux de bonnes classifications pour les ensembles de données avec variable-réponse catégorielle

Dans tous les cas, pour les classifications, nous avons utilisé 10 groupes (k) pour la validation-croisée.

Jeu de données	Observations (n)	Variables (n)	Taux de bonnes classifications (%)							
			taux CART (%)	taux ctree (%)	Meilleur taux arbres	Différence CART et ctree (%)	taux rfsrc (%)	taux cforest (%)	Meilleur taux forêts	Différence rfsrc et cforest (%)
BC	699	10	94.86	95.57	ctree	-0.74	96.86	94.86	rfsrc	2.11
DNA	3186	181	91.76	91.91	ctree	-0.16	95.74	94.86	rfsrc	0.93
Glass	214	10	72.27	68.64	CART	5.29	75.91	66.82	rfsrc	13.60
Ionosphere	351	36	89.72	89.17	CART	0.62	91.39	91.11	rfsrc	0.31
PID	768	9	70.42	70.71	ctree	-0.41	77.27	75.06	rfsrc	2.94
Soybean	683	36	78.80	80.00	ctree	-1.50	92.17	85.51	rfsrc	7.80
LR	20000	17	82.96	80.67	CART	2.84	96.25	90.54	rfsrc	6.31
Satellite	6435	37	87.00	85.77	CART	1.43	91.80	88.87	rfsrc	3.30
TAE	151	6	46.88	50.00	ctree	-6.24	58.75	51.25	rfsrc	14.63
CMC	1473	10	55.88	55.34	CART	0.98	52.16	55.61	cforest	-6.20
SMO	2854	13	69.41	69.41	=	0.00	66.50	70.31	cforest	-5.42
VH	846	19	69.00	68.00	CART	1.47	76.24	72.94	rfsrc	4.52
HV	435	17	96.36	96.56	ctree	-0.21	97.05	96.14	rfsrc	0.95
Sonar	208	61	74.29	75.71	ctree	-1.88	82.38	78.10	rfsrc	5.48
GC	1000	21	49.60	48.60	CART	2.06	74.60	74.00	rfsrc	0.81
Ecoli	336	8	85.88	85.88	=	0.00	86.47	84.41	rfsrc	2.44
wf (sim.)	1800	22	76.33	75.33	CART	1.33	82.78	82.89	cforest	-0.13
t2n (sim.)	1800	21	83.22	79.50	CART	4.68	96.00	95.39	rfsrc	0.64
t3n (sim.)	1800	21	71.00	68.67	CART	3.39	86.56	84.33	rfsrc	2.64
rn (sim.)	1800	21	84.67	82.89	CART	2.15	94.22	90.00	rfsrc	4.69
		minimum	**46.88**	**48.60**		**-6.24**	**52.16**	**51.25**		**-6.20**
		maximum	**96.36**	**96.56**		**5.29**	**97.05**	**96.14**		**14.63**
		médiane	**77.57**	**77.61**		**0.80**	**86.52**	**84.37**		**2.54**
		moyenne	**76.52**	**75.92**		**0.75**	**83.56**	**81.15**		**3.12**

4.2. Tableau comparatif des résultats de l'erreur moyenne au carré pour les ensembles de données avec variable-réponse continue

Pour ce qui est des ensembles de données avec variable-réponse continue, dans tous les cas, pour les arbres, nous avons utilisé 10 groupes (k) pour la validation-croisée. Pour les forêts nous avons également utilisé 10 groupes, à l'exception du cas d'Ozone où nous avons utilisé 4 groupes car nous n'arrivions pas à exécuter cforest avec 10.

Jeu de données	Observations (n)	Variables (n)	Erreur quadratique moyenne							
			CART	ctree	Plus faible erreur	Différence CART et ctree (%)	rfsrc	cforest	Plus faible erreur	Différence rfsrc et cforest (%)
BH	**506**	**14**	22.72	22.85	CART	0.57	11.38	17.93	rfsrc	57.56
Ozone	**366**	**13**	31.77	26.45	ctree	-16.75	18.33	18.62	rfsrc	1.58
LD	**345**	**6**	0.23	0.24	CART	4.35	0.75	0.74	cforest	-1.52
Baseball	**337**	**17**	551992	557598	CART	1.02	466540	539330	rfsrc	15.60
f1 (sim.)	**1800**	**11**	6.93	7.85	CART	13.28	3.48	5.09	rfsrc	46.26
f2 (sim.)	**1800**	**5**	20837	20785	ctree	-0.25	17142	16950	cforest	-1.12
f3 (sim.)	**1800**	**5**	0.022	0.023	CART	5.40	0.01	0.02	rfsrc	21.32
minimum			**0.02**	**0.02**		**-16.75**	**0.01**	**0.02**		**-1.52**
maximum			**551992**	**557598**		**13.28**	**466540**	**539330**		**57.56**
médiane			**22.72**	**22.85**		**1.02**	**11.38**	**17.93**		**15.60**
moyenne			**81842**	**82634**		**1.09**	**69102**	**79475**		**19.96**

Comparaison de la performance des deux algorithmes sur les jeux de données réels et simulations inspirés de la littérature

En observant les performances prédictives des deux modèles d'arbres à l'étude, nous constatons que la différence entre les deux méthodes n'est pas très grande comme l'indiquent les tableaux des résultats.

Pour ce qui est des jeux de données avec variable-réponse catégorielle (tableau 4.1.), 11/20 obtiennent une meilleure prédiction avec CART et 7/20 avec ctree. Les deux autres cas obtiennent des prévisions identiques. La mesure de la différence entre CART et ctree montre qu'il n'y a que 7 cas où la différence est de + que 2% ou encore de – de 2%. La différence moyenne du taux de bonnes classifications entre les deux modèles est de 0.75% et la médiane de 0.80% en faveur de CART ce qui n'est pas significatif.

Pour ce qui est des jeux de données avec variable-réponse continue (tableau 4.2.), les arbres CART obtiennent un meilleur résultat dans 5 cas sur 7. Cependant un des cas, Ozone, la différence est beaucoup plus significative en faveur de ctree (-16.75%). Dans

un autre cas, f1, la différence est beaucoup plus significative en faveur de CART (+13.28%). La moyenne de la différence entre les erreurs quadratique est de 1.09% et la médiane de 1.02% en faveur de CART, ce qui n'est pas très grand.

Les deux algorithmes semblent donc avoir une capacité de prévision semblable pour ce qui est des arbres, avec un léger avantage pour CART.

Cependant, pour ce qui est des forêts aléatoires, en observant les deux tableaux, on comprend rapidement que la performance prédictive des forêts classiques basées sur des arbres CART (rfsrc) surpasse majoritairement celle des forêts aléatoires non biaisées basées sur des arbres d'inférence conditionnelle (cforest).

Pour ce qui est des jeux de données avec variable-réponse catégorielle (tableau 4.1.), 17/20 obtiennent une meilleure prédiction avec rfsrc (12 cas de plus de 2%) et 3/20 avec cforest (2 cas de plus de 2%). La différence moyenne du taux de bonnes classifications entre les deux modèles est de 3.12% et la différence médiane de 2.54% en faveur de rfsrc.

Pour ce qui est des jeux de données avec variable-réponse continue (tableau 4.2.), les forêts 'rfsrc' obtiennent une meilleure prédiction que les 'cforest' 5/7 fois. Pour les deux autres cas, la différence de prédiction penche en faveur de cforest, mais avec moins de 2% de différence. La différence moyenne de 20% et médiane de 15,6% de l'erreur quadratique moyenne est ici beaucoup plus élevée en faveur de rfsrc.

Il est à noter également que, bien que cela ne soit pas l'objectif de cette recherche, rfsrc produit un résultat beaucoup plus rapidement que cforest. À performance égale, nous privilégierions donc l'utilisation de cette première méthode. Ceci étant dit, la performance prédictive des forêts aléatoires construites à partir des arbres CART (rfsrc) n'est pas égale, mais nettement supérieure en moyenne et dans la forte majorité des cas.

Si une forêt est construite dans un but de maximiser le résultat de prédiction, le choix d'algorithme parmi les deux analysés ici est donc pour le moment évident. L'intérêt est donc maintenant de comprendre par l'évaluation des résultats sur les jeux de données dont les paramètres ont été contrôlés, dans quels cas précis l'algorithme cforest est exceptionnellement meilleur en prédiction que rfsrc.

Méthode d'évaluation de la performance pour tous les jeux de données créés artificiellement

Chacun des douze jeux de données artificiels sera étudié avec un nombre différent d'observations composant le jeu de données d'entraînement : 200; 500; 1000 ou 5000. De plus, le nombre de variables choisi aléatoirement pour la construction des arbres parmi lesquelles les meilleurs points de coupure seront choisi sera de 4 ou de 11. Il y aura donc au total 96 différentes évaluations des jeux de données (12 jeux de données X 2 options de nombre de variables choisies aléatoirement X 4 taille de données dans le jeu d'entraînement).

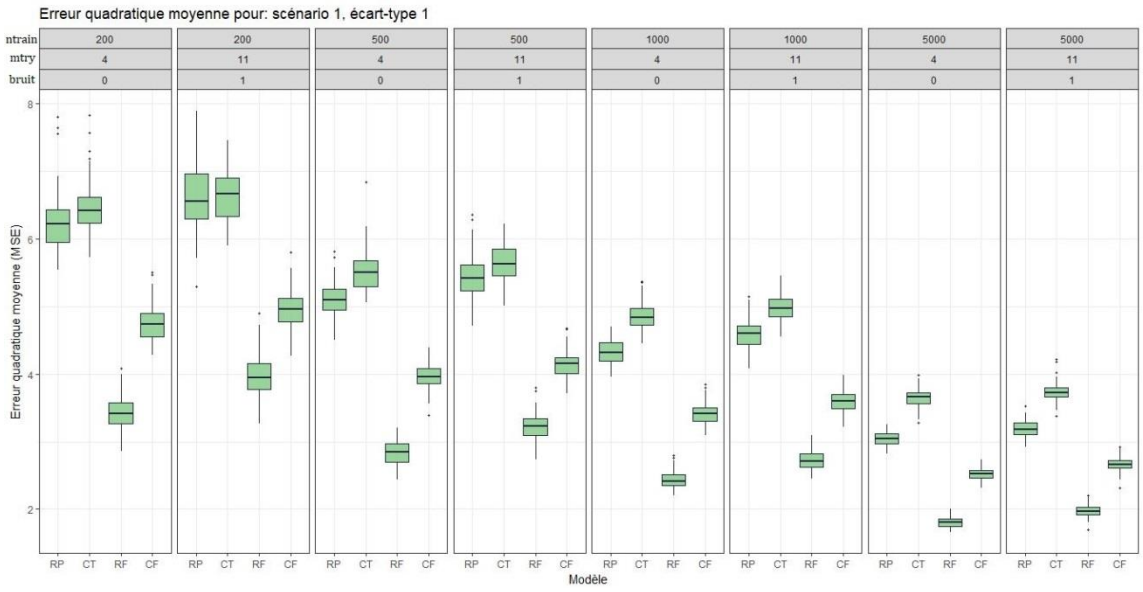
scénario	bruit	Nombre total de X continus	Nombre total de X catégoriels	Nombre de X continus liés à Y	Nombre de X catégoriels liés à Y
1	0	6	6	6	0
1	1	16	16	6	0
2	0	6	6	0	6
2	1	16	16	0	6
3	0	6	6	6	6
3	1	16	16	6	6

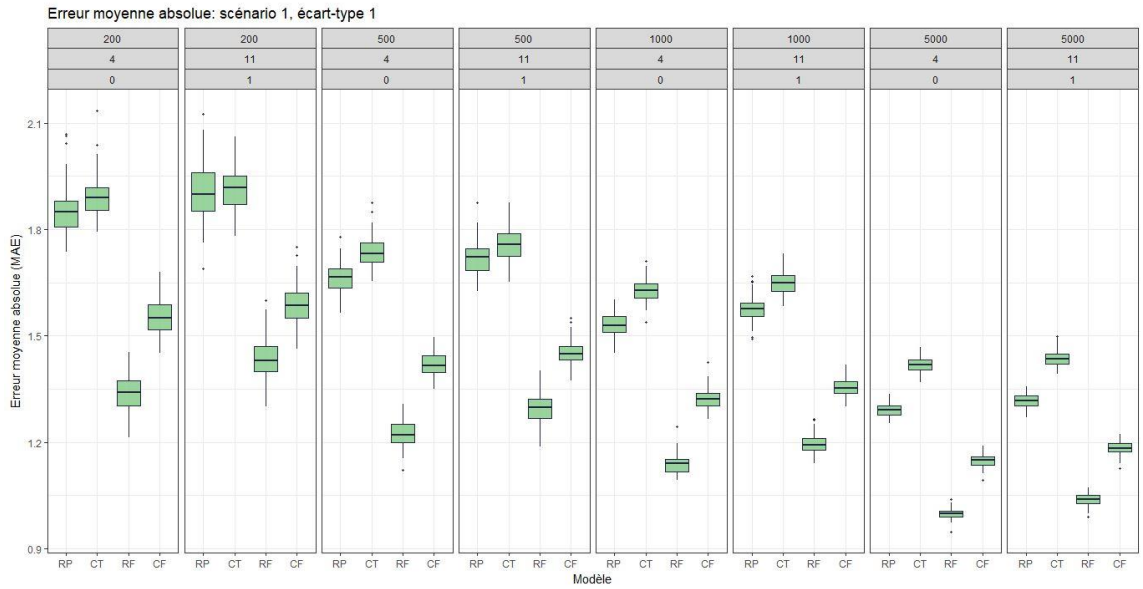
100 simulations différentes seront effectuées à partir de chaque ensemble de paramètres et les résultats de performances seront mesurés en utilisant deux mesures : l'erreur quadratique moyenne et l'erreur moyenne absolue et ce pour les arbres construits à partir de CART (rpart), les arbres construits à partir d'inférence conditionnelle (ctree),

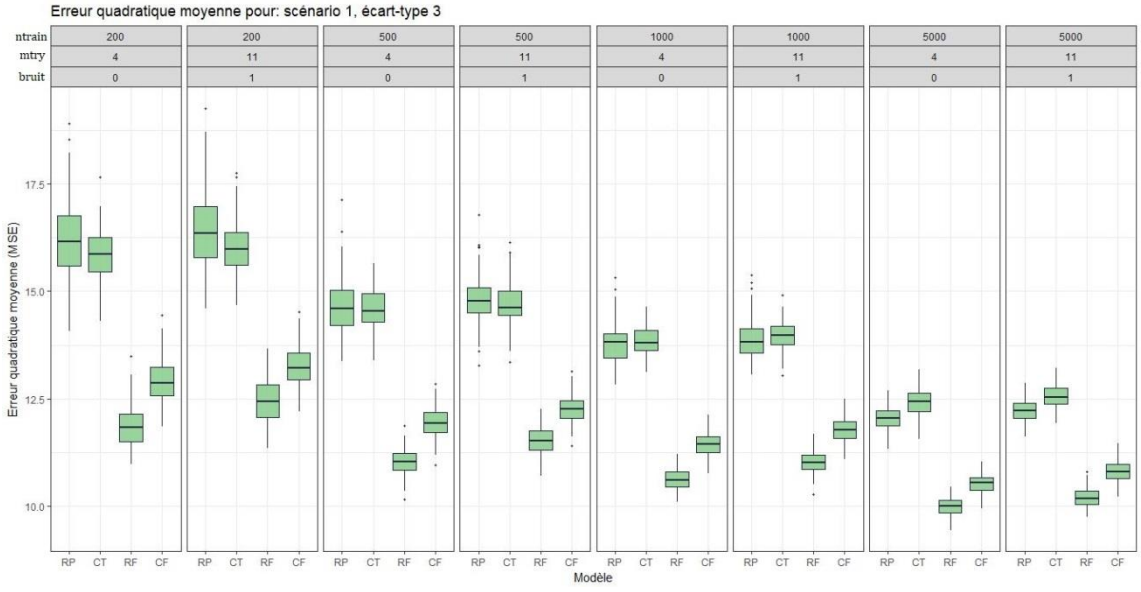
les forêts classiques (RFSRC - RF) et les forêts alternatives (cforest- CF). Les résultats sont donc ici présentés sous forme de tracés en boîtes dans douze graphes (3 scénarios X écart-type de 1 ou de 3 X 2 mesures). Le détail du code utilisé dans R pour générer les scénarios est disponible en annexe 1.

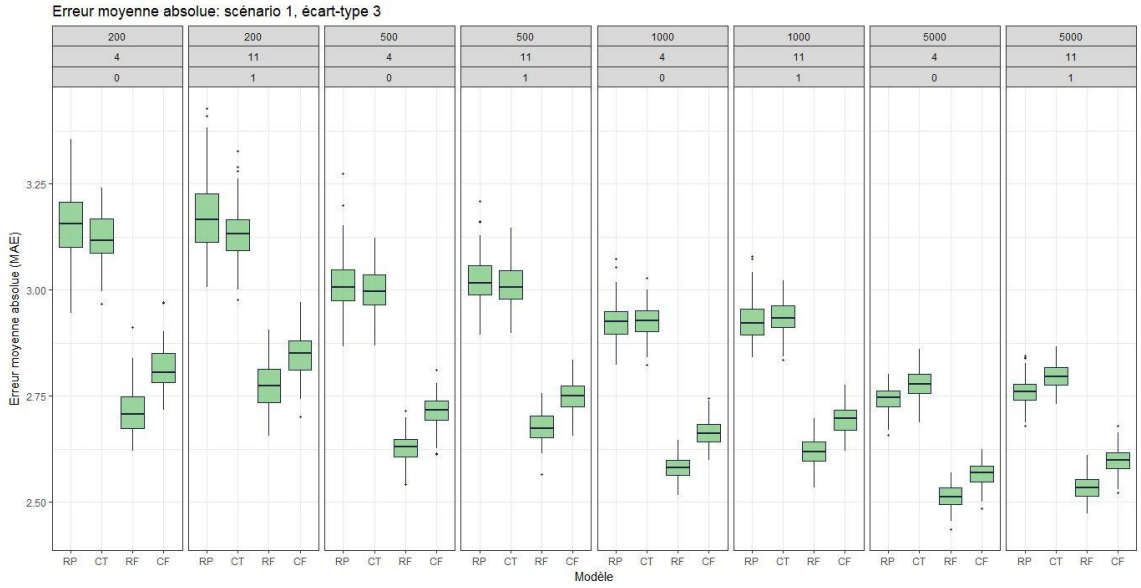
4.5. Graphes des résultats de trois scénarios de simulation évalués par deux méthodes

Scénario 1

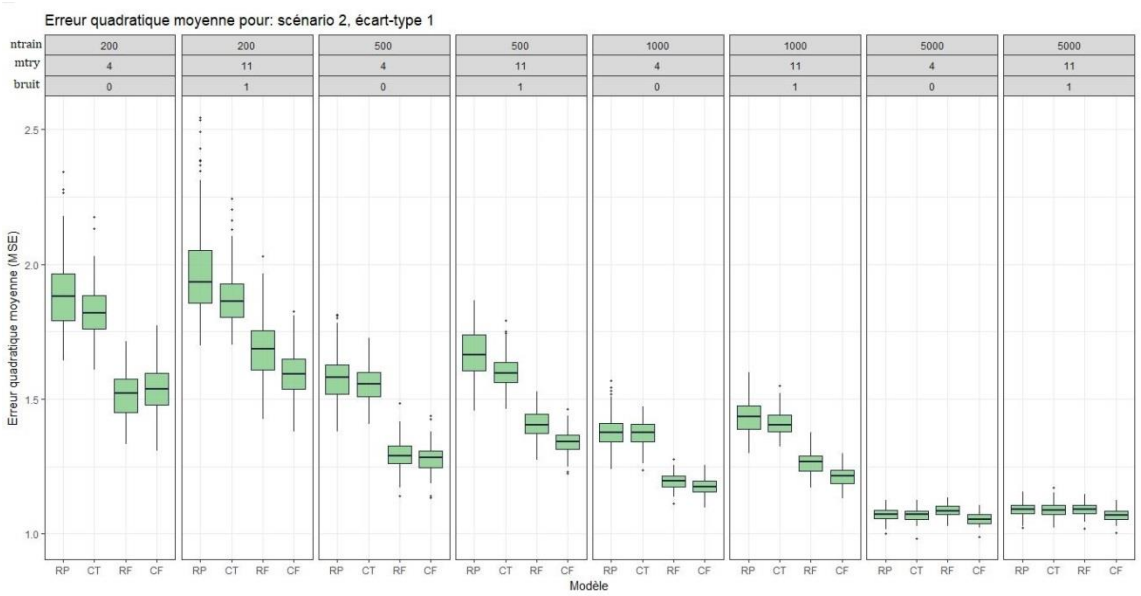


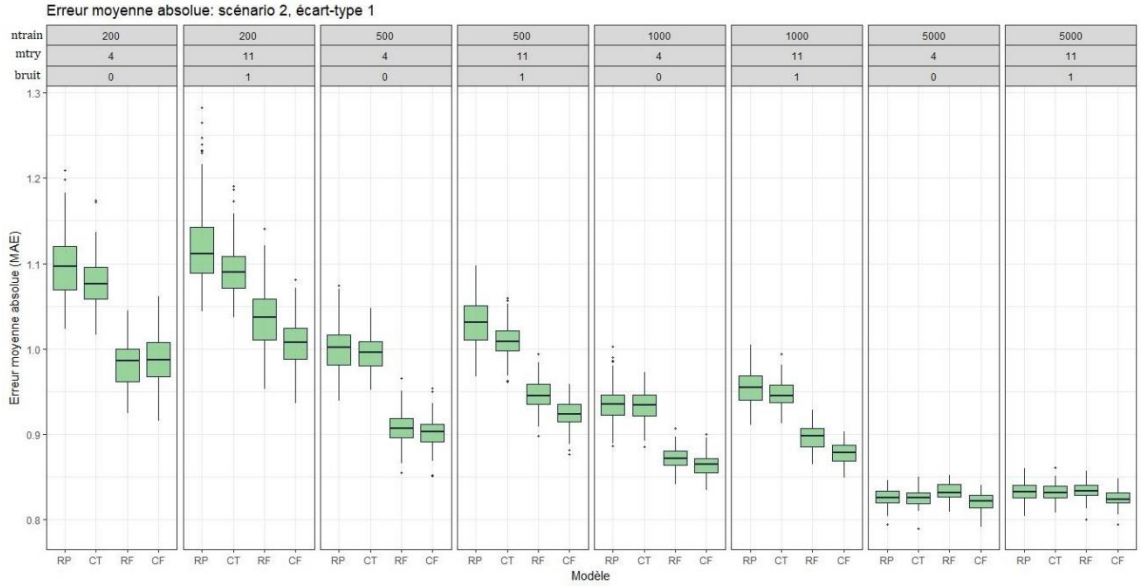


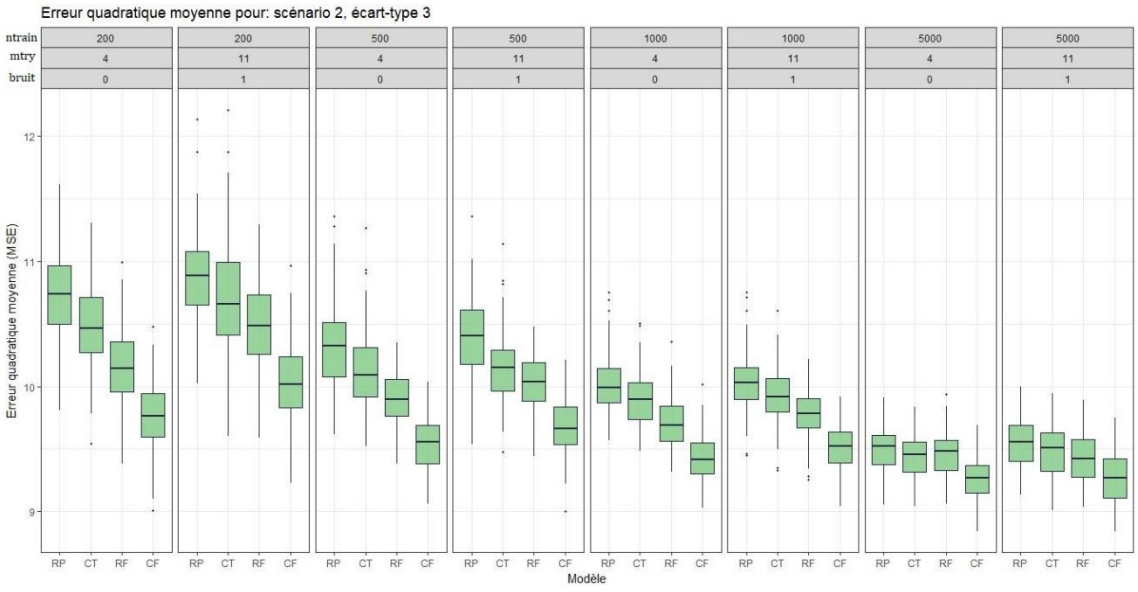

Erreur quadratique moyenne pour: scénario 1, écart-type 3
ntrain
mtry
bruit
200
4
0
200
11
1
500
4
0
500
11
1
1000
4
0
1000
11
1
5000
4
0
5000
11
1
Erreur quadratique moyenne (MSE)
17.5
15.0
12.5
10.0
RP
CT
RF
CF
Modèle

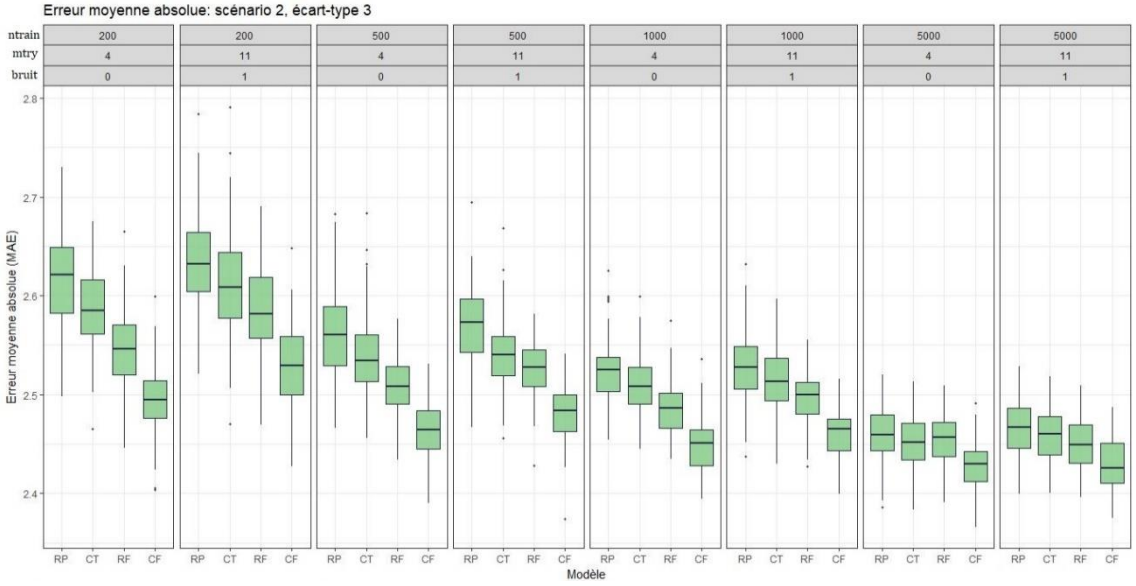

Erreur moyenne absolue: scénario 1, écart-type 3
200
4
0
200
11
1
500
4
0
500
11
1
1000
4
0
1000
11
1
5000
4
0
5000
11
1
Erreur moyenne absolue (MAE)
3.25
3.00
2.75
2.50
RP
CT
RF
CF
Modèle

Scénario 2

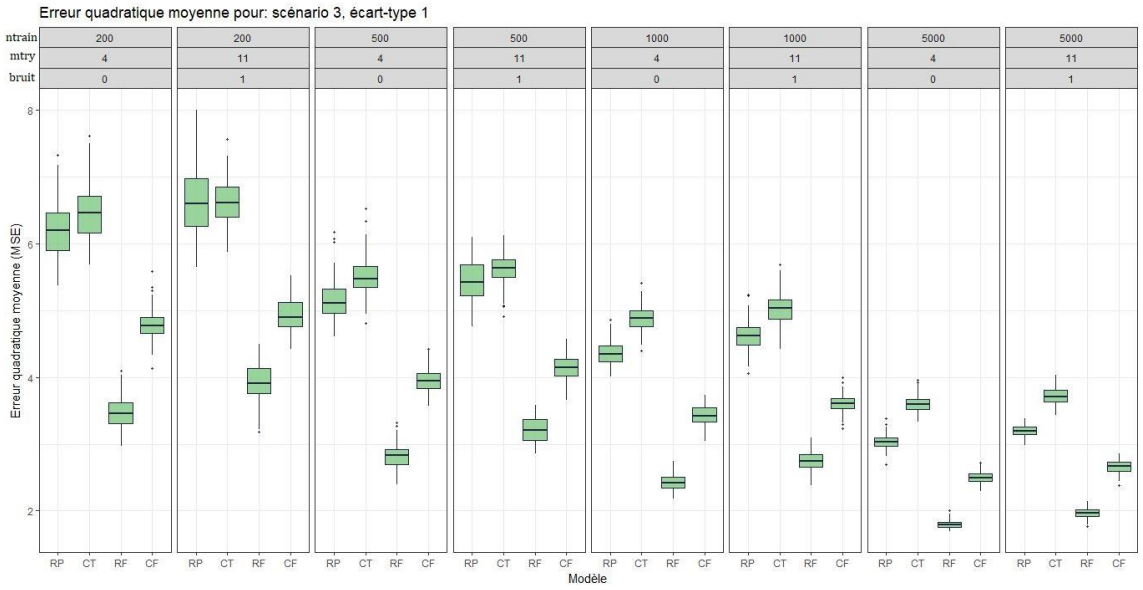

Erreur quadratique moyenne pour: scénario 2, écart-type 1
ntrain
mtry
bruit
200
4
0
200
11
1
500
4
0
500
11
1
1000
4
0
1000
11
1
5000
4
0
5000
11
1
Erreur quadratique moyenne (MSE)
2.5
2.0
1.5
1.0
RP CT RF CF
Modèle

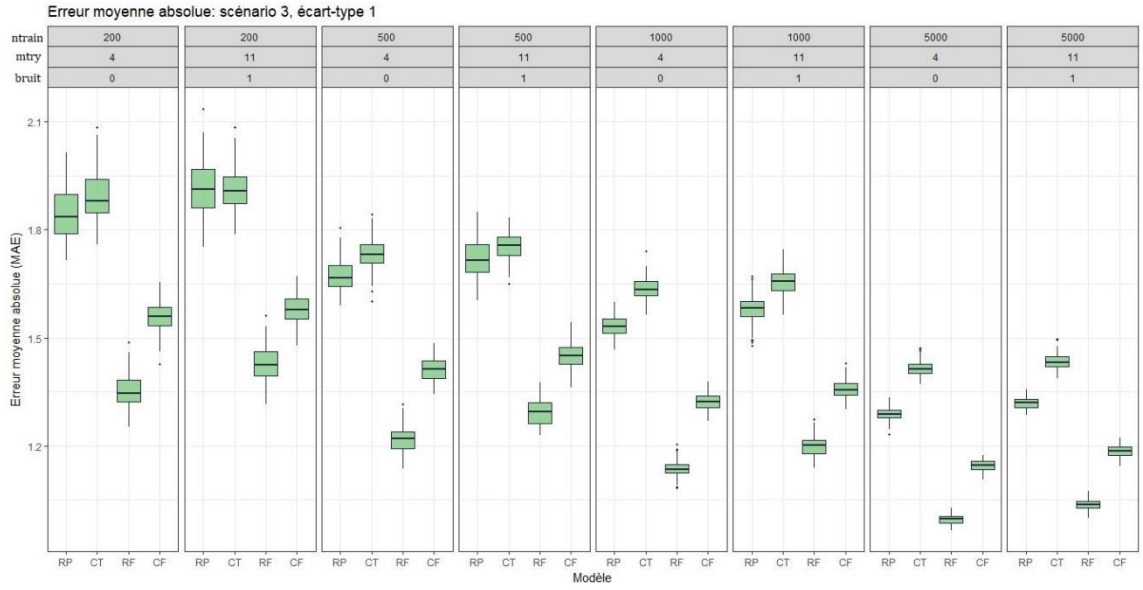

Erreur moyenne absolue: scénario 2, écart-type 1
ntrain
mtry
bruit
200
4
0
200
11
1
500
4
0
500
11
1
1000
4
0
1000
11
1
5000
4
0
5000
11
1
Erreur moyenne absolue (MAE)
1.3
1.2
1.1
1.0
0.9
0.8
RP CT RF CF
Modèle

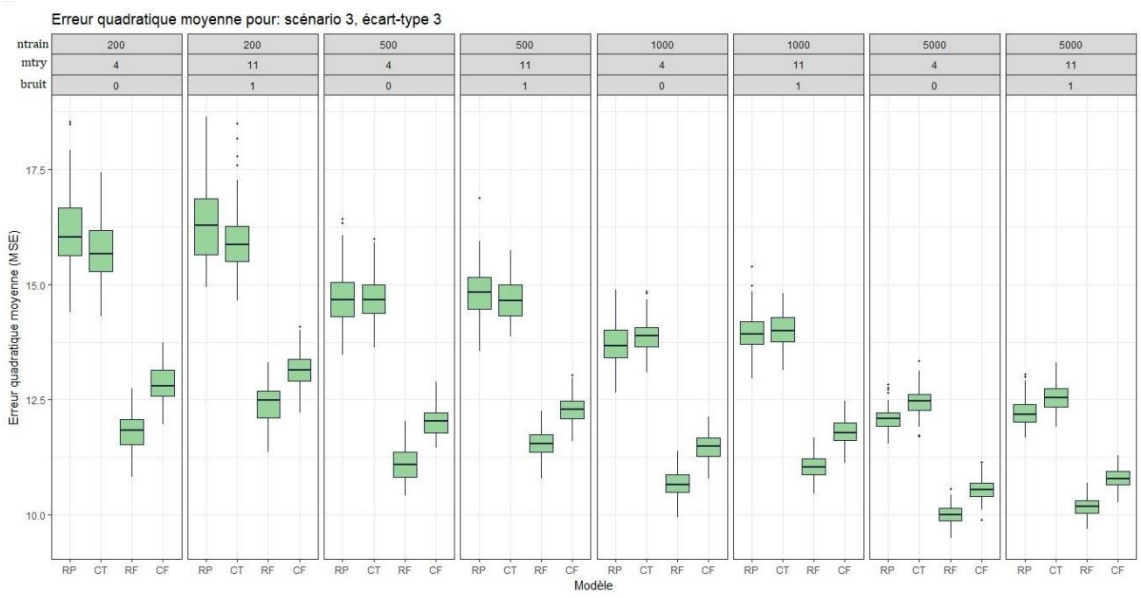

Erreur quadratique moyenne pour: scénario 2, écart-type 3
ntrain
mtry
bruit
200
4
0
200
11
1
500
4
0
500
11
1
1000
4
0
1000
11
1
5000
4
0
5000
11
1
Erreur quadratique moyenne (MSE)
12
11
10
9
RP
CT
RF
CF
Modèle

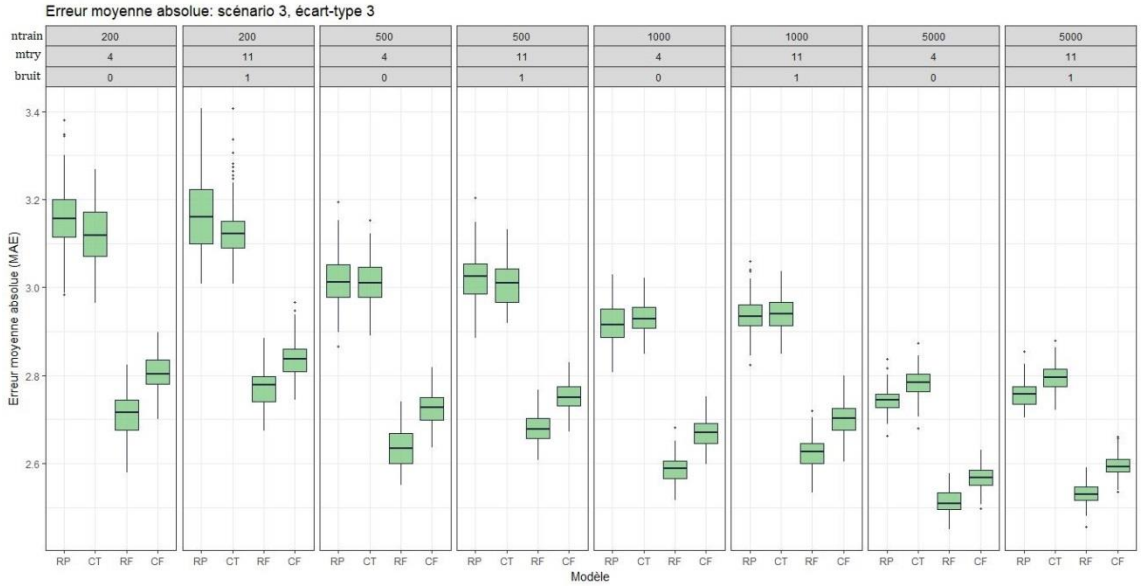

Erreur moyenne absolue: scénario 2, écart-type 3
ntrain
mtry
bruit
200
4
0
200
11
1
500
4
0
500
11
1
1000
4
0
1000
11
1
5000
4
0
5000
11
1
Erreur moyenne absolue (MAE)
2.8
2.7
2.6
2.5
2.4
RP
CT
RF
CF
Modèle

Scénario 3


Erreur quadratique moyenne pour: scénario 3, écart-type 1
ntrain
mtry
bruit
200
4
0
200
11
1
500
4
0
500
11
1
1000
4
0
1000
11
1
5000
4
0
5000
11
1
Erreur quadratique moyenne (MSE)
8
6
4
2
RP
CT
RF
CF
Modèle


Erreur moyenne absolue: scénario 3, écart-type 1
ntrain
mtry
bruit
200
4
0
200
11
1
500
4
0
500
11
1
1000
4
0
1000
11
1
5000
4
0
5000
11
1
Erreur moyenne absolue (MAE)
2.1
1.8
1.5
1.2
RP
CT
RF
CF
Modèle





Discussion des résultats : comparaison de la performance sur les jeux de données créés artificiellement avec contrôle des paramètres

Comme attendu, l'erreur diminue lorsque ntrain augmente, et ce, pour chacun des quatre modèles à l'étude ici (CART, ctree, rfsrc et cforest). De plus, les forêts aléatoires

obtiennent de meilleures performances prédictives que les arbres, ce qui était également prévisible.

Comme attendu également, l'erreur augmente lorsque des variables explicatives ne créant que du bruit dans les données sont introduites. Contrairement à ce que nous prévoyions cependant, cette augmentation est assez légère.

Comme nous l'avions déjà vu dans les cas des jeux de données réels, la différence de performance prédictive entre les arbres CART et ctree est plus petite qu'entre les forêts RF et cforest.

Pour ce qui est du scénario 1, contenant des variables explicatives continues uniquement, les forêts aléatoires classiques performent persuasivement mieux que les forêts 'non-biaisées' cforest en termes de prédiction. Le troisième quartile des résultats de RF est régulièrement plus performant que le premier quartile des résultats de CF. C'est le cas dans toutes les options testées et selon les deux mesures de performance utilisées.

Les résultats du scénario 2 dans lequel les variables explicatives sont toutes catégorielles sont très différents. Les forêts non-biaisées cforest performent ici mieux que ceux des forêts classiques (RF). Lorsque l'écart-type est de 1 pour la variable-réponse, les résultats sont légèrement supérieurs. Cependant, lorsque l'écart-type est de 3, la performance prédictive de cforest dépassent nettement celle des forêts classiques (RF). La performance prédictive des arbres non-biaisés est alors également supérieure que celle des arbres classiques CART. On voit donc ici possiblement l'effet du fait que les arbres CART, et par extension donc les forêts aléatoires bâties sur ceux-ci, ont tendance à sélectionner des variables avec plus de points de coupures possibles (les variables explicatives continues ici). Comme seulement les variables explicatives catégorielles sont liées à la variable à prédire dans le scénario 2, la performance en souffre un peu

lorsque l'écart-type est de 3. Cependant, dans le scénario 1, cette tendance aide les arbres et forêts (CART) car justement ce sont les variables explicatives continues qui sont liées à la variable à prédire.

Pour ce qui est du scénario 3 où il existe un mélange de variables explicatives catégorielles et continues, les résultats sont sans ambiguïté. Les forêts aléatoires classiques (RF) performent significativement mieux que les forêts non-biaisées cforest (CF). En effet, dans tous les cas ici, le troisième quartile des résultats de RF est plus performant que le premier quartile des résultats de CF.

Pour ce qui est des jeux de données construits artificiellement, les forêts aléatoires classiques ont une performance prédictive supérieure aux forêts d'inférence conditionnelle dans les scénarios 1 et 3 où certaine ou la totalité des variables explicatives sont continues. Cependant, dans le cas du scénario 2 où les variables explicatives sont toutes catégorielles, cforest offre de meilleures prédictions.

Chapitre 5
Conclusion et discussion

Quel type de forêt aléatoire offre la meilleure performance prédictive? Une forêt 'classique' basée sur des arbres CART ou une forêt basée sur des arbres 'non-biaisés' d'inférence conditionnelle? Telle était le questionnement à l'origine de la recherche présentée ici. Après avoir étudié les prédictions sur les jeux de données réels et simulés à partir d'exemples tirés de la littérature, la conclusion de cette étude semble manifeste : dans une proportion de 85%, l'algorithme RF a mieux performé que CF au niveau de la capacité de prédiction, en plus de requérir moins de performance machine. Pour ce qui est des arbres cependant, l'algorithme qui réussit à obtenir le meilleur résultat varie : CT et CART obtiennent tour à tour de meilleures performances.

Cependant, en ajoutant des simulations construites à partir de paramètres contrôlés, un scénario a permis de trouver un cas où les forêts non-biaisées performent mieux que les forêts classiques. En effet, si toutes les variables explicatives sont catégorielles, alors CF offre une meilleure performance prédictive que RF. Dans les deux autres scénarios cependant, RF surpasse les prédictions de CF.

L'apport de cette étude dépasse le monde de la recherche universitaire. Au niveau pratique, plusieurs utilisateurs de R cherchent la différence d'utilisation entre les deux types de forêts. Par exemple, sur le site web 'StackExchange', beaucoup utilisé en programmation, nous avons noté deux séries de questions suivies de commentaires et réponses. La première, publiée en 2011, avec commentaires en 2013, puis en 2015, traite avec surprise de la différence entre la faible erreur de classification produite par `randomForest` (forêts classiques basées sur arbres CART) et l'erreur plus importante en utilisant `cforest` pour un cas précis alors que l'erreur est semblable dans des jeux de données contenant plus d'observations:

https://stats.stackexchange.com/questions/6855/cforest-and-randomforest-classification-prediction-error[6].

La seconde, publiée en 2014, avec commentaires en 2014, puis 2016 s'interrogeant sur la différence dans le choix des variables encore une fois entre `randomForest` et `cforest :`

https://stats.stackexchange.com/questions/36324/randomforest-vs-cforest-can-i-get-partial-dependence-plots-and-percent-varianc [7].

Nous avons également trouvé une troisième discussion sur le blogue 'Analytics Vidhya' datant de 2015 cherchant à comparer pourquoi `cforest` était supposément mieux que `randomForest:`

https://discuss.analyticsvidhya.com/t/difference-between-cforest-and-random-forest-in-r/5455 [8].

Cette recherche peut donc contribuer, quelques années plus tard, à éclaircir un public utilisant R sur la question.

Ce mémoire de maîtrise comprend plusieurs limites. D'abord, nous n'expliquons pas pourquoi les forêts aléatoires classiques prédisent généralement mieux que les forêts conditionnelles ni pourquoi, dans certains cas, l'inverse se produit. Il s'agit d'une démonstration pratique sans explication théorique autre que celle de la différence entre avoir une totalité de variables explicatives catégorielles ou non. Nous ne connaissons pas les raisons du succès prédictif des forêts classiques. Ensuite, vingt-sept ensembles de données ont choisi d'être utilisé après avoir été noté comme ayant préalablement été utilisés dans la littérature. Il est possible qu'un type spécifique d'ensemble de données

[6] Page consultée le 5 novembre 2017.
[7] Page consultée le 5 novembre 2017.
[8] Page consultée le 5 novembre 2017.

ne s'y trouve pas et que les résultats sur ce type particulier de données soient différents. De plus, nos jeux de données créés artificiellement avaient des caractéristiques contrôlées, mais toutes les possibilités ne s'y trouvaient pas évidemment. La présence d'une totalité de variables explicatives catégorielle n'est peut-être pas le seul cas de figure où l'algorithme CF surpasse en performance prédictives RF.

Il aurait aussi été intéressant de comparer d'autres algorithmes de forêts 'non-biaisés' également, il s'agit d'ailleurs d'une possibilité pour de futures recherches sur le sujet. GUIDE (Loh 2009) serait particulièrement intéressant à comparer comme algorithme de forêt aléatoires 'non-biaisées', bien qu'il soit plus complexe à utiliser puisque n'étant pas directement intégré via un package dans R ou Python.

La capacité des forêts aléatoires classiques, basées sur la méthode développée par Leo Breiman au début du siècle, est difficile à expliquer. Continuons néanmoins de l'utiliser pour obtenir d'excellentes prédictions.

Bibliographie

Amaratunga, D., J. Cabrera, and Y.-S. Lee. 2008. “Enriched Random Forests.” *Bioinformatics* 24 (18): 2010–14. https://doi.org/10.1093/bioinformatics/btn356.

Breiman, Leo, ed. 1984. *Classification and Regression Trees*. Repr. Boca Raton: Chapman & Hall [u.a.].

———. 1996a. “Stacked Regressions.” *Machine Learning* 24 (1): 49–64. https://doi.org/10.1007/BF00117832.

———. 1996b. “Bagging Predictors.” *Machine Learning* 24 (2): 123–40. https://doi.org/10.1007/BF00058655.

———. 2001. “Random Forests.” *Machine Learning* 45: 5–32.

Breiman, Leo, and W. S. Meisel. 1976. “General Estimates of the Intrinsic Variability of Data in Nonlinear Regression Models.” *Journal of the American Statistical Association* 71 (354): 301–7.

Buuren, Stef van. 2017. “Package ‘Mice’: Multivariate Imputation by Chained Equations.” https://cran.r-project.org/web/packages/mice/index.html.

Cutler, Adele. 2010. “Remembering Leo Breiman.” *The Annals of Applied Statistics* 4 (4): 1621–33. https://doi.org/10.1214/10-AOAS427.

Forsyth, Richard S., and BUPA Medical Research Ltd. 1990. “Liver Disorders Data Set.” *UCI Machine Learning Repository* (blog). 1990. https://archive.ics.uci.edu/ml/datasets/Liver+Disorders.

Friedman, Jerome H. 1991. “Multivariate Adaptive Regression Splines.” *The Annals of Statistics*, 1–67.

Hamza, Mounir, and Denis Larocque. 2005. “An Empirical Comparison of Ensemble Methods Based on Classification Trees.” *Journal of Statistical Computation and Simulation* 75 (8): 629–43. https://doi.org/10.1080/00949650410001729472.

Hofman, Pr. Dr. Hans, and Institut für Statistik und Ökonometrie, Universität Hamburg. 1994. “Statlog (German Credit Data) Data Set.” *UCI Machine Learning Repository* (blog). 1994. https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data).

Horton, Paul, and Nakai, Kenta. 1994. “Ecoli: Protein Localization Sites.” *UCI Machine Learning Repository* (blog). 1994. https://archive.ics.uci.edu/ml/datasets/Ecoli.

Hothorn, Torsten, Kurt Hornik, Carolin Strobl, and Achim Zeileis. 2016. “Package ‘Party’: A Laboratory for Recursive Partytioning.” http://party.R-forge.R-project.org.

Hothorn, Torsten, Kurt Hornik, and Achim Zeileis. 2006. “Unbiased Recursive Partitioning: A Conditional Inference Framework.” *Journal of Computational and Graphical Statistics* 15 (3): 651–74. https://doi.org/10.1198/106186006X133933.

Hothorn, Torsten, and Achim Zeileis. 2015. “Partykit: A Modular Toolkit for Recursive Partytioning in R.” *Journal of Machine Learning Research*.

Ishwaran, Hemant. 2015. “The Effect of Splitting on Random Forests.” *Machine Learning* 99 (1): 75–118. https://doi.org/10.1007/s10994-014-5451-2.

Ishwaran, Hemant, and Udaya B. Kogalur. 2017. “Package ‘randomForestSRC’: Random Forests for Survival, Regression and Classification.” https: //cran.r-project.org/package=randomForestSRC.

Kass, G. V. 1980. “An Exploratory Technique for Investigating Large Quantities of Categorical Data.” *Applied Statistics* 29 (2): 119–27.

Kim, Hyunjoong, and Wei-Yin Loh. 2001. “Classification Trees With Unbiased Multiway Splits.” *Journal of the American Statistical Association* 96 (454): 589–604. https://doi.org/10.1198/016214501753168271.

Lange, Nicholas, Louise Ryan, and Lynne Billard. 1994. “Case Studies in Biometry.”

Leisch, Friedrich, and Evgenia Dimitriadou. 2015. “Package ‘Mlbench’: Machine Learning Benchmark Problems.” https://cran.r-project.org/web/packages/mlbench/index.html.
Liaw, Andy, and Matthew Wiener. 2002. “Classification and Regression by randomForest.” *R News* 2 (3): 18–22.
———. 2015. “Package ‘randomForest’: Breiman and Cutler’s Random Forests for Classification and Regression.” https://www.stat.berkeley.edu/~breiman/RandomForests/.
Lichman, M. 2013. “Teaching Assistant Evaluation Data Set.” *UCI Machine Learning Repository* (blog). 2013. http://archive.ics.uci.edu/mll/datasets/Teaching+Assistant+Evaluation.
Lim, Tjen-Sien. 1997. “Contraceptive Method Choice Data Set.” *UCI Machine Learning Repository* (blog). 1997. https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice.
Lim, Tjen-Sien, Wei-Yin Loh, and Yu-Shan Shih. 2000. “A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms.” *Machine Learning* 40 (3): 203–28.
Loh, Wei-Yin. 2002. “Regression Trees With Unbiased Variable Selection and Interaction Detection.” *Statistica Sinica* 12: 361–86.
———. 2009. “Improving the Precision of Classification Trees.” *The Annals of Applied Statistics* 3 (4): 1710–37. https://doi.org/10.1214/09-AOAS260.
———. 2014. “Fifty Years of Classification and Regression Trees: Fifty Years of Classification and Regression Trees.” *International Statistical Review* 82 (3): 329–48. https://doi.org/10.1111/insr.12016.
Loh, Wei-Yin, and Yu-Shan Shih. 1997. “Split Selection Methods for Classification Trees.” *Statistica Sinica* 7 (4): 815–40.
Messenger, R., and L. Mandell. 1972. “A Modal Search Technique for Predictive Nominal Scale Multivariate Analysis.” *Journal of American Statistical Association* 67: 754–177.
Michie, D., D. J. Spiegelhalter, and C. C. Taylor. 1994. “The Statlog Project.” In *Machine Learning, Neural and Statistical Classification*. 4.
Milborrow, Stephen. 2017. “Package ‘Rpart.plot’: Plot ‘Rpart’ Models.” https://cran.r-project.org/web/packages/rpart.plot/rpart.plot.pdf.
Mola, F., and R. Siciliano. 1999. “A General Splitting Criterion for Classification Trees.” *Metron* 57: 155–71.
Morgan, J.N., and J.A. Sonquist. 1963. “Problems in the Analysis of Survey Data, and a Proposal.” *Journal of American Statistical Association* 58: 415–34.
NC State University, and Watnik, Mitchell R. 1996. “Baseball Salaries Data.” 1996. http://www4.stat.ncsu.edu/~boos/var.select/baseball.html.
Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge University Press. Cambridge.
Shelley B. Bull. 1994. “Analysis of Attitudes towards Workplace Smoking Restrictions Data Set.” *StatLib: Case Studies in Biometry, ch13* (blog). 1994. http://lib.stat.cmu.edu/datasets/csb/.
Shih, Yu-Shan. 2001. “Selecting the Best Splits for Classification Trees with Categorical Variables.” *Statistics & Probability Letters*, no. 54: 341–45.
———. 2004. “A Note on Split Selection Bias in Classification Trees.” *Computational Statistics & Data Analysis* 45 (3): 457–66. https://doi.org/10.1016/S0167-9473(03)00064-1.
Strobl, Carolin. 2004. “Variable Selection Bias in Classification Trees.” München: Ludwig-Maximilians-Universität.
———. 2008. “Statistical Issues in Machine Learning: Towards Reliable Split Selection and Variable Importance Measures.” München: Ludwig-Maximilians-Universität.

Strobl, Carolin, Anne-Laure Boulesteix, and Thomas Augustin. 2007. “Unbiased Split Selection for Classification Trees Based on the Gini Index.” *Computational Statistics & Data Analysis* 52 (1): 483–501. https://doi.org/10.1016/j.csda.2006.12.030.

Strobl, Carolin, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. “Conditional Variable Importance for Random Forests.” *BMC Bioinformatics* 9 (1): 307. https://doi.org/10.1186/1471-2105-9-307.

Strobl, Carolin, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. 2007. “Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution.” *BMC Bioinformatics* 8 (25): 1–21.

Thernau, Terry, Beth Atkinson, and Brian Ripley. 2018. “Package ‘Rpart’: Recursive Partitioning and Regression Trees.” https://cran.r-project.org/web/packages/rpart/rpart.pdf.

White, Allan P., and Wei Zhong Liu. 1994. “Bias in Information-Based Measures in Decision Tree Induction.” *Machine Learning* 15 (3): 321–29. https://doi.org/10.1007/BF00993349.

Annexe 1 :

Code utilisé pour générer les jeux de données construits artificiellement

```
dgp=function(n,scenario,sigma,noise){

  # n = sample size
  # scenario =
            # 1 = only continuous predictors
            # 2 = only categorical predictors
            # 3 = mix of both
  # sigma = STD of the error term
  # noise = add 20 noise predictors (1) or not (0).  10 continuous and
10 categorical.

# output
# x1,...,x12 = 12 covariates
# y = response

library(mvtnorm)

rho=.3
sig=matrix(rho,12,12)-(rho-1)*diag(12)
x=rmvnorm(n, mean = rep(0, 12), sigma = sig)
dat=data.frame(x)

dat$X7=0*(dat$X7<=0)+1*(dat$X7>0)
dat$X8=0*(dat$X8<=0)+1*(dat$X8>0)
dat$X9=0*(dat$X9<=-.5)+1*(dat$X9>-.5)*(dat$X9<.5)+2*(dat$X9>=.5)
dat$X10=0*(dat$X10<=-.5)+1*(dat$X10>-.5)*(dat$X10<.5)+2*(dat$X10>=.5)
dat$X11=0*(dat$X11<=-1)+1*(dat$X11>-
1)*(dat$X11<0)+2*(dat$X11>=0)*(dat$X11<1)+3*(dat$X11>=1)
dat$X12=0*(dat$X12<=-1)+1*(dat$X12>-
1)*(dat$X12<0)+2*(dat$X12>=0)*(dat$X12<1)+3*(dat$X12>=1)

for(j in 7:12)
{
for(i in 1:n)
{
dat[i,j]=paste("v",dat[i,j],sep="")
}
}

for(j in 7:12)
{
dat[,j]=factor(dat[,j])
}

if(scenario==1)
{
me=
(dat$X1-mean(dat$X1))/sd(dat$X1)+(dat$X2-mean(dat$X2))/sd(dat$X2)+
```

```
(dat$X3^2-mean(dat$X3^2))/sd(dat$X3^2)+(dat$X4^2-
mean(dat$X4^2))/sd(dat$X4^2)  +
(dat$X5^(3)-mean(dat$X5^(3)))/sd(dat$X5^(3)) +(dat$X6^(3)-
mean(dat$X6^(3)))/sd(dat$X6^(3))
(dat$X1*dat$X3-mean(dat$X1*dat$X3))/sd(dat$X1*dat$X3)+
(dat$X2*dat$X5-mean(dat$X2*dat$X5))/sd(dat$X2*dat$X5)

}

if(scenario==2)
{
me=
(dat$X7=="v0")+(dat$X8=="v1")+
(dat$X9=="v0")+(dat$X10=="v0")+(dat$X10=="v3")+
(dat$X11=="v1")+(dat$X12=="v0")+(dat$X12=="v4")+
(dat$X7=="v0")*(dat$X9=="v0")+(dat$X9=="v2")+
(dat$X8=="v1")*(dat$X11=="v1")*(dat$X11=="v3")
}

if(scenario==3)
{
me=
(dat$X1-mean(dat$X1))/sd(dat$X1)+(dat$X2-mean(dat$X2))/sd(dat$X2)+
(dat$X3^2-mean(dat$X3^2))/sd(dat$X3^2)+(dat$X4^2-
mean(dat$X4^2))/sd(dat$X4^2)  +
(dat$X5^(3)-mean(dat$X5^(3)))/sd(dat$X5^(3)) +(dat$X6^(3)-
mean(dat$X6^(3)))/sd(dat$X6^(3))
(dat$X1*dat$X3-mean(dat$X1*dat$X3))/sd(dat$X1*dat$X3)+
(dat$X2*dat$X5-mean(dat$X2*dat$X5))/sd(dat$X2*dat$X5)+

(dat$X7=="v0")+(dat$X8=="v1")+
(dat$X9=="v0")+(dat$X10=="v0")+(dat$X10=="v3")+
(dat$X11=="v1")+(dat$X12=="v0")+(dat$X12=="v4")+
(dat$X7=="v0")*(dat$X9=="v0")+(dat$X9=="v2")+
(dat$X8=="v1")*(dat$X11=="v1")*(dat$X11=="v3")+

(dat$X1-mean(dat$X1))/sd(dat$X1)*(dat$X7=="v0")+
(dat$X3^2-mean(dat$X3^2))/sd(dat$X3^2)*(dat$X10=="v0")+(dat$X10=="v3")

}

if(noise==1)
{
dat$X13=rnorm(n)
dat$X14=rnorm(n)
dat$X15=rnorm(n)
dat$X16=rnorm(n)
dat$X17=rnorm(n)
dat$X18=rnorm(n)
dat$X19=rnorm(n)
dat$X20=rnorm(n)
dat$X21=rnorm(n)
dat$X22=rnorm(n)

dat$X23=rbinom(n,2,.5)
dat$X24=rbinom(n,2,.5)
dat$X25=rbinom(n,2,.5)
```

```
dat$X26=rbinom(n,2,.5)
dat$X27=rbinom(n,3,.5)
dat$X28=rbinom(n,3,.5)
dat$X29=rbinom(n,3,.5)
dat$X30=rbinom(n,4,.5)
dat$X31=rbinom(n,4,.5)
dat$X32=rbinom(n,4,.5)

for(j in 23:32)
{
for(i in 1:n)
{
dat[i,j]=paste("v",dat[i,j],sep="")
}
}

for(j in 23:32)
{
dat[,j]=factor(dat[,j])
}


}


dat$y=me+sigma*rnorm(n)

dat

}
```