

ARBRES DE DECISION et MÉTHODES D'ENSEMBLES (Random Forests, gradient boosting

Arbres et forêts aléatoires

Famille : *"machine learning"*

Type: *Approches basées sur les observations (non sur un modèle théorique)*

Spécificité: *Partitionnements répétés*

Arbres et méthodes d'ensembles

- Arbres de décision :

Différents algorithmes : CART (Breiman *et al.*, 1984), C4.5 (Quinlan, 1993),
CHAID (Kas, 1980)
CI Tree (Othorn *et al.*, 2006), GUIDE (Loh *et al.*, 2009)....

Le plus populaire: **CART (Classification And Regression Tree)**

dichotomies successives du jeu de données

Vers 1996 - 2001, Breiman introduit les notions de communautés/ensembles d'arbres :

- Bagging

- Random Forests

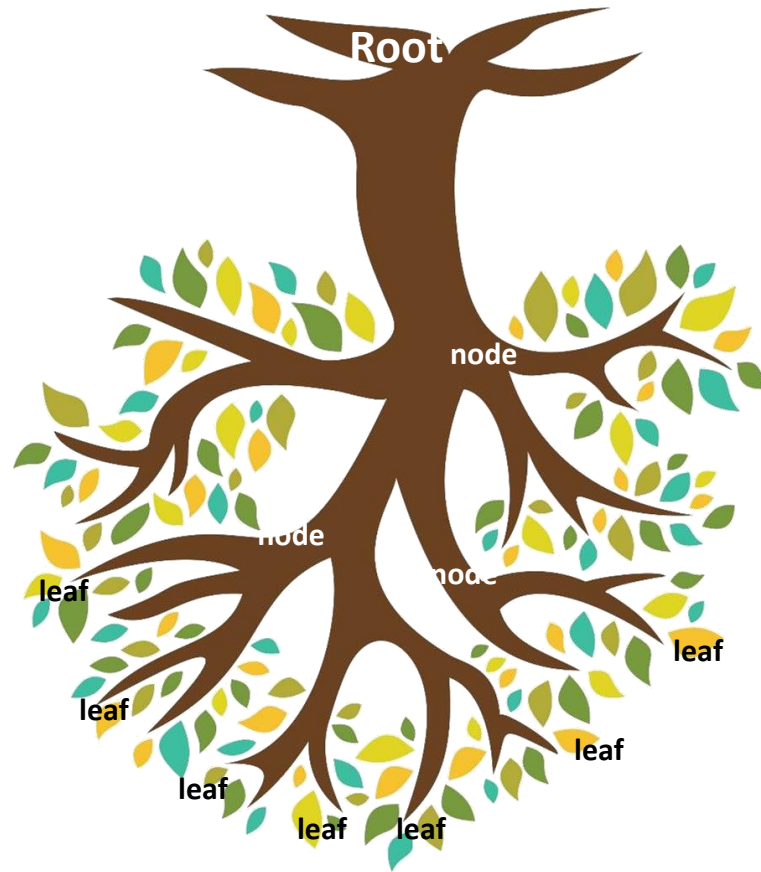
Parallèlement, des approches dites de « boosting » se sont développées (sous différentes formes) dont :

- Gradient boosting

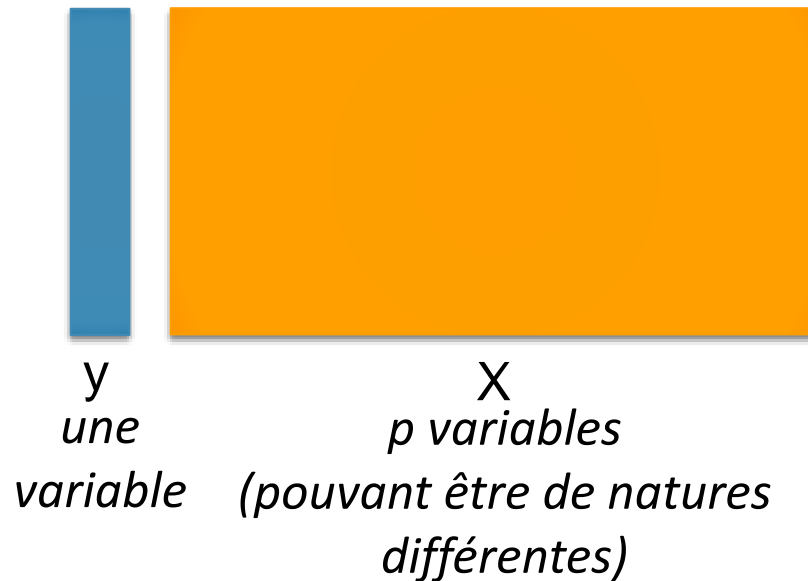
Arbre de décision



Arbre de décision



CART



- **Arbre de classification** : réponse y qualitative
- **Arbre de régression** : réponse y quantitative

Ex. Arbre de décision (CART)

Hitters (ISLR package)

Major League Baseball Data from the 1986 and 1987 seasons.

261 players

y : *annual salary (10^3 dollars)*

X: * *#Years (Major League),*

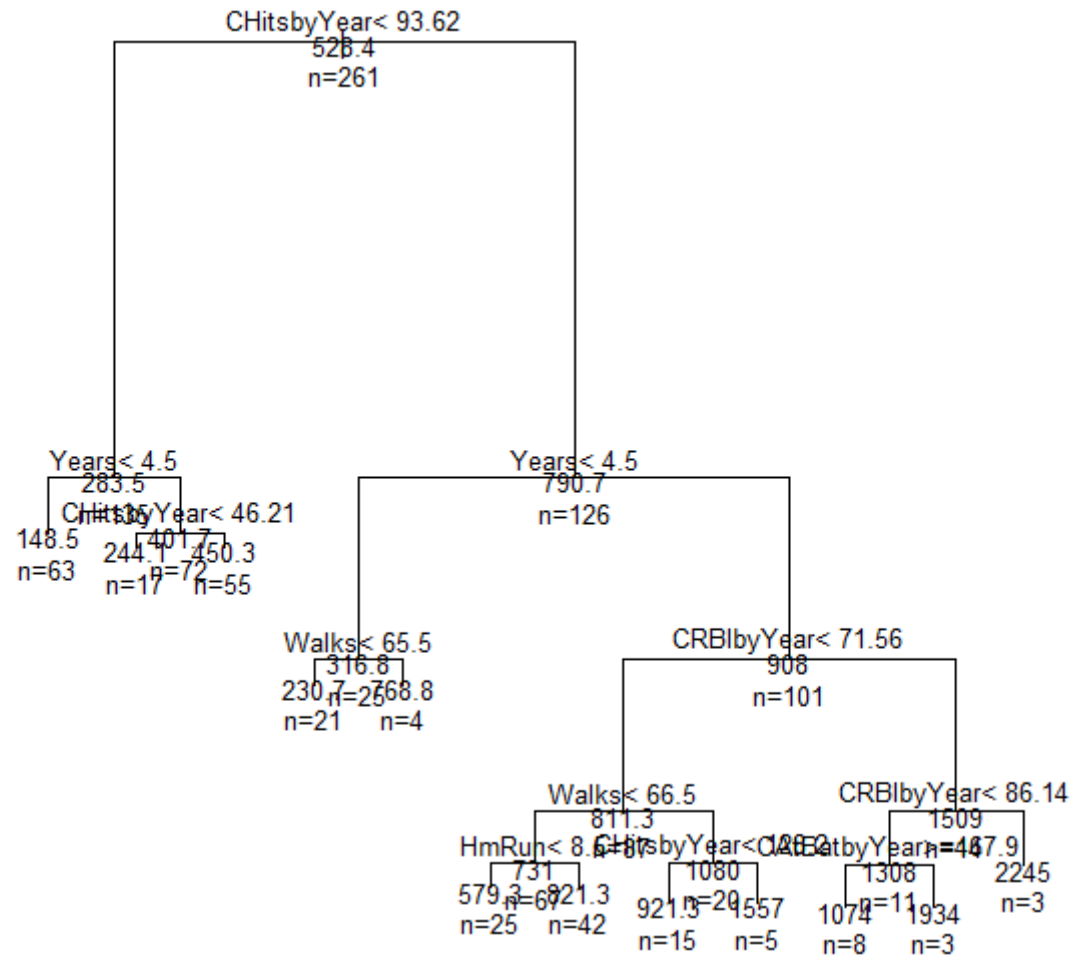
* *performances in the year (#AtBats, #Hits, #HomeRuns, #Runs, #RBI, #Walks),*

* *past performances per year (#CAtBatsbyYear, #CHitsbyYear, #CHmRunsbyYear, #CRunsbyYear, #CRBIbyYear, #CWalksbyYear)*

Lexique des statistiques au baseball : <http://www.argancy-baseball.com/lexique-statistiques/>

Ex. Arbre de décision(CART)

Hitters
y: annual salary

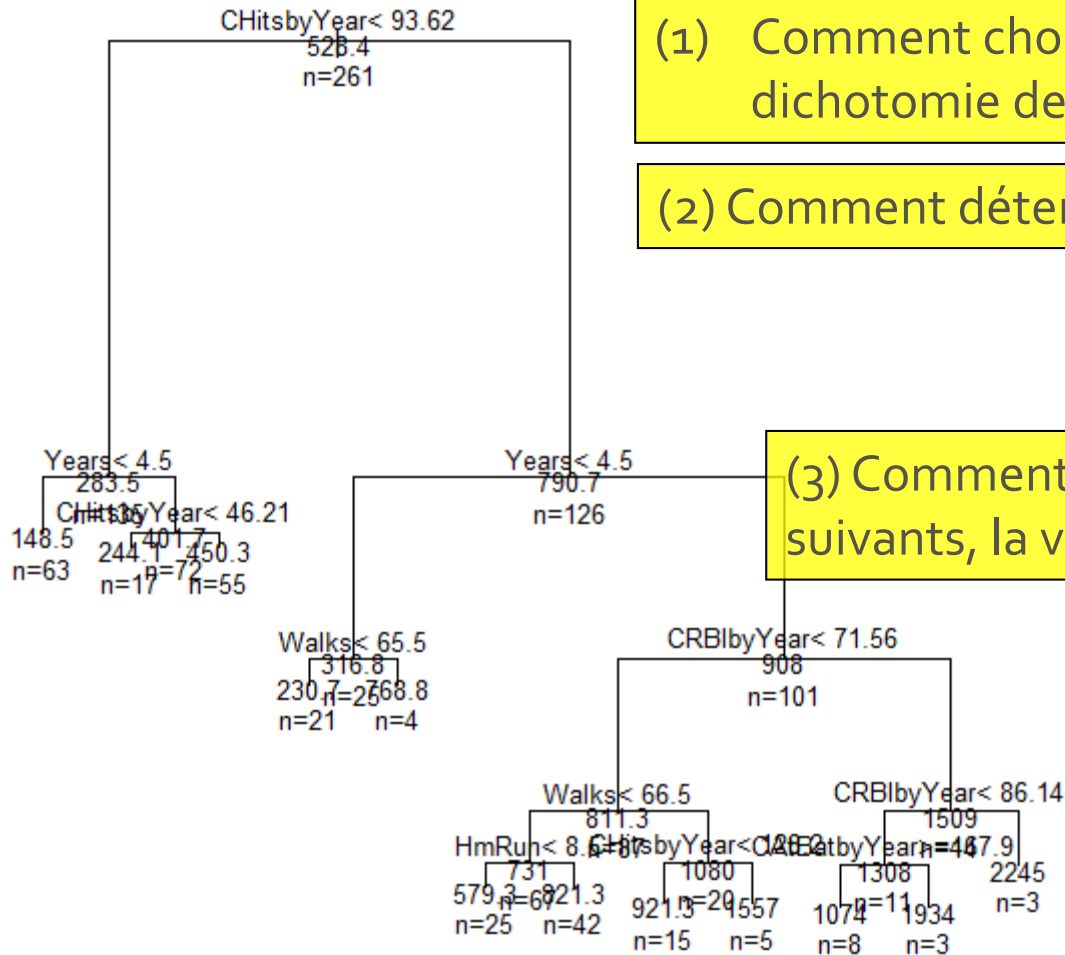


Arbre de décision

(1) Comment choisir la variable pour la première dichotomie des données ?

(2) Comment déterminer le seuil pour couper ?

(3) Comment choisir, à chacun des noeuds suivants, la variable et le niveau de coupure ?

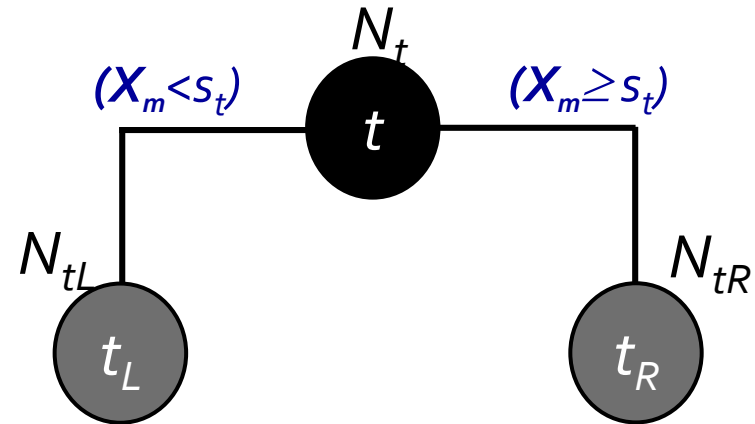


(4) A quel moment arrêter ?

CART

Objectif : A chaque nœud,
le jeu d'apprentissage est coupé en deux
en choisissant une variable X_m et un niveau de coupure s ,
de sorte à
maximiser la diminution (globale) de l'impureté

CART: Critère



A chaque nœud t , 2 paramètres (X_m, s_t) sont déterminés de sorte à maximiser

$$\Delta i(s, t) = i(t) - \frac{N_{tL}}{N_t} i(t_L) - \frac{N_{tR}}{N_t} i(t_R)$$

$i(t)$: mesure d'impureté

CART: Critère

$i(t)$: mesure d'impureté

Arbre de Classification

Réponse \mathbf{y} , ayant K modalités

Gini index

ou
$$i(t) = G = \sum_{k=1}^K p_{tk}(1 - p_{tk})$$

Shannon Entropy

$$i(t) = D = - \sum_{k=1}^K p_{tk} \log(p_{tk})$$

p_{tk} : proportion des obs. au noeud t présentant la modalité k .

Plus les p_{tk} sont proches de 0 ou de 1, plus G (ou D) est petit.

Arbre de Régression

Variance de \mathbf{y} pour les obs. au noeud t

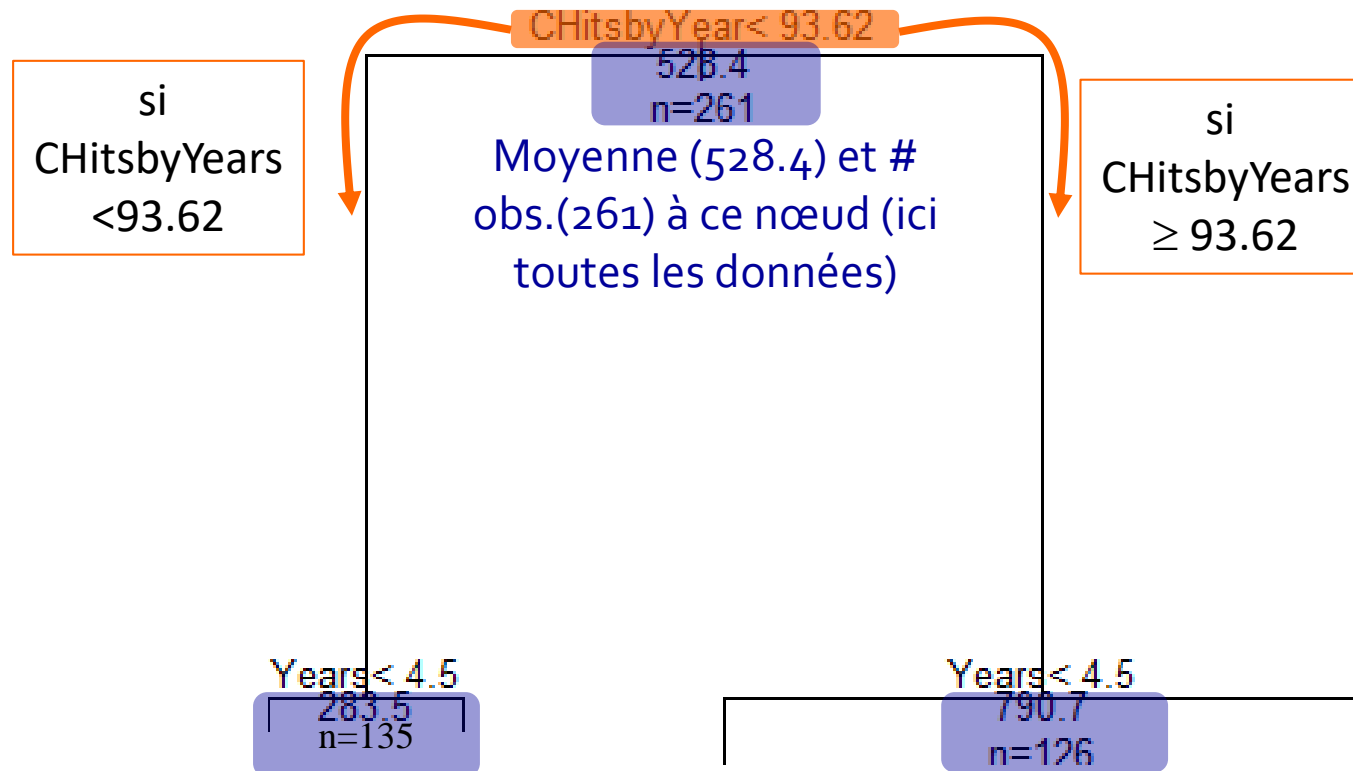
$$i(t) = \sum_{l=1}^{N_t} (y_{m,l} - \bar{y}_{m/l \in t})^2 / N_t$$

Minimisation de la variance intra, à chaque noeud
 \Leftrightarrow Maximisation de la variance inter.

Ex. Arbre de Régression

Hitters
y: annual salary

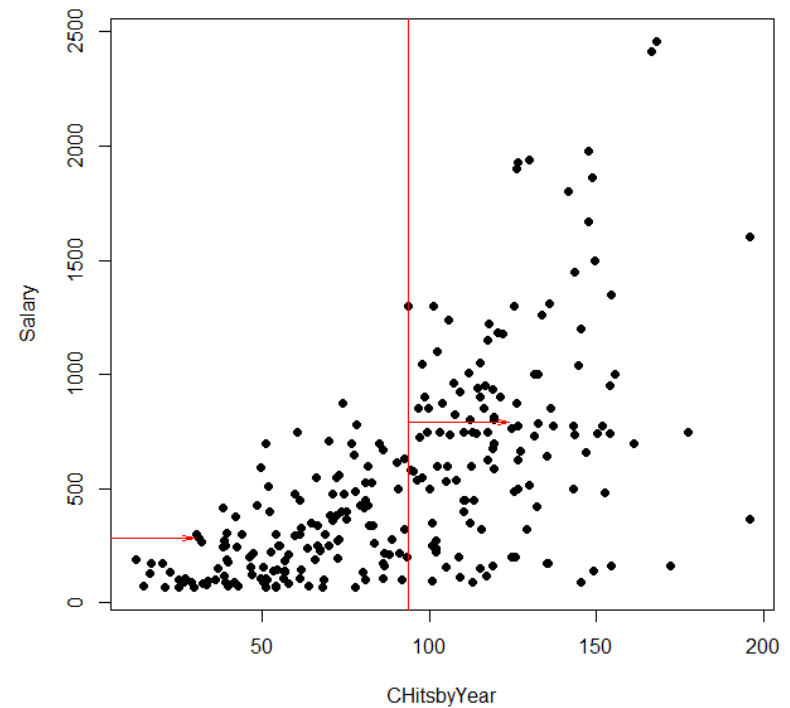
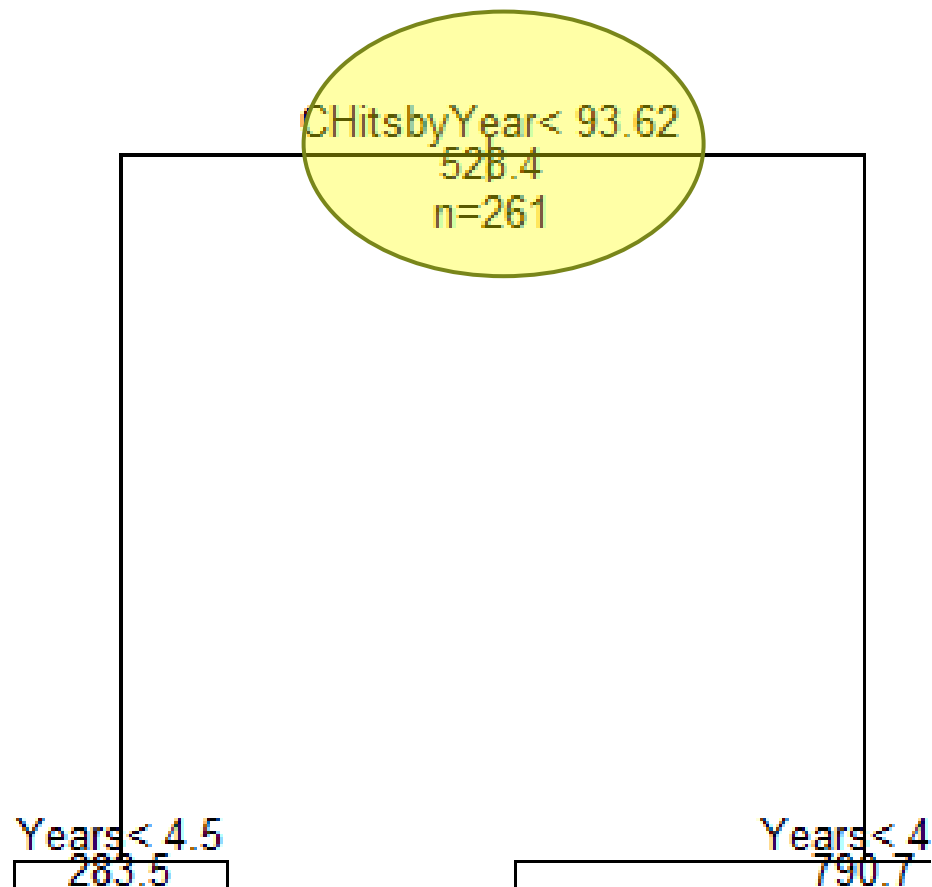
Variable pour la coupure et seuil



Ex. Arbre de Régression

Hitters
y: annual salary

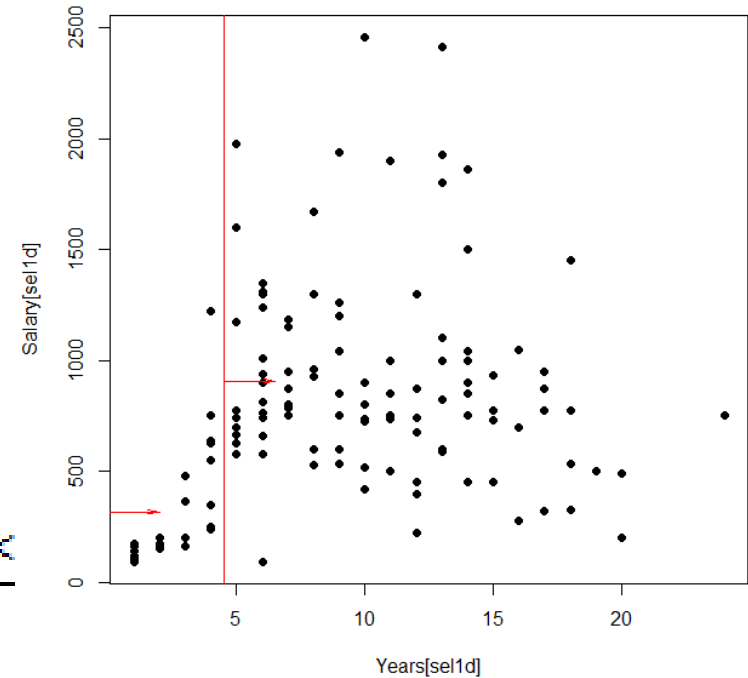
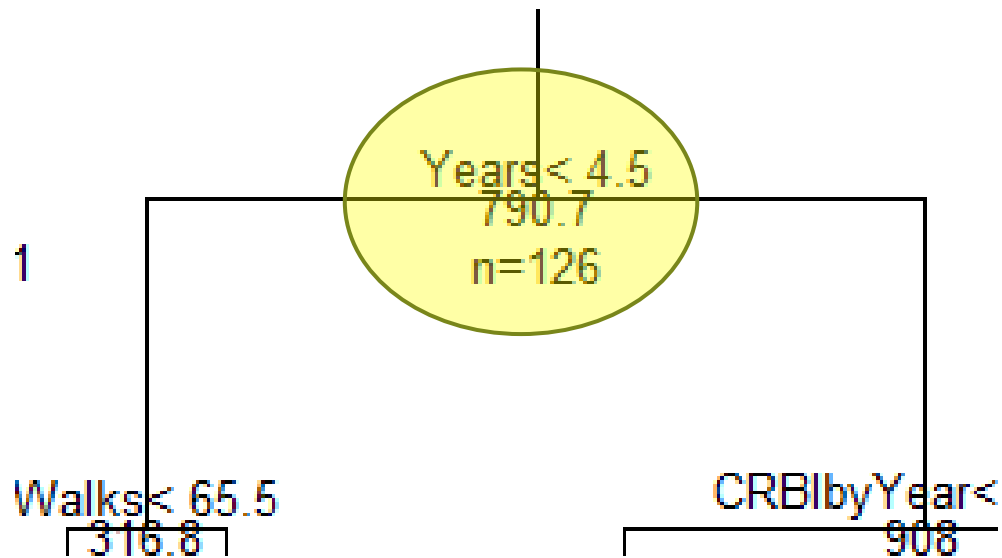
1^{er} noeud



Ex. Arbre de Régression

Hitters
y: annual salary

2^{ième} noeud



Ex. Arbre de Classification

OJ (ISLR package)

Citrus Hill or Minute Maid Orange Juice purchase.

1070 observations

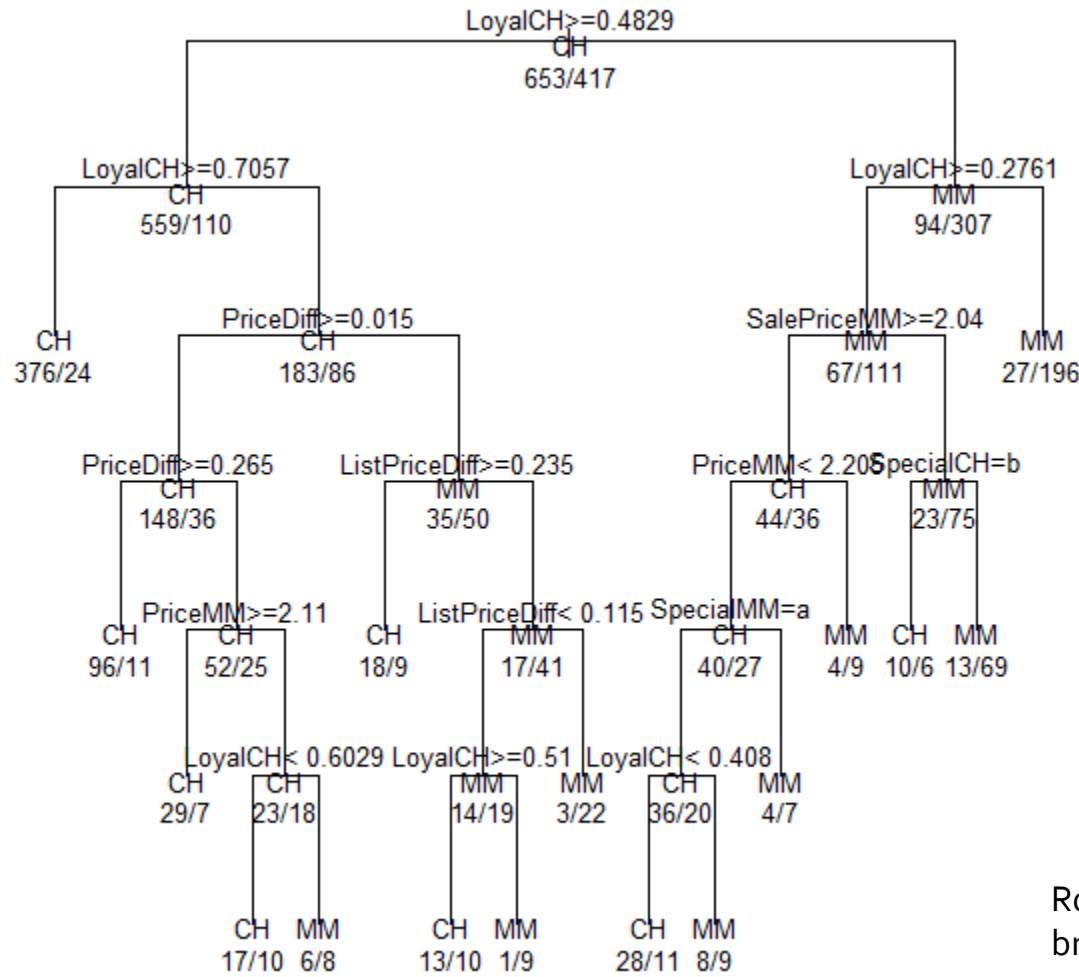
y : #Purchase

CH	MM
653	417

X: "Price", "Discount", "Special", "SalePrice", "PctDisc" for CH and MM
"PriceDiff", "ListPriceDiff",
"LoyalCH" (Customer Brand Loyalty for CH),
"STORE" (Which of 5 possible stores the sale occurred at)

Ex. Arbre de Classification

OJ (ISLR package)
y : #purchase

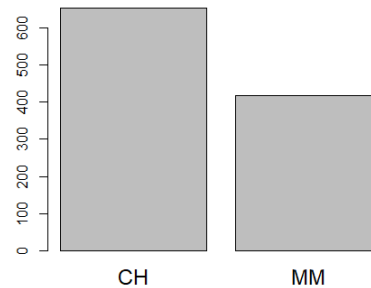


Rq. Longueur des
branches identique pour
une meilleure lisibilité

Ex. Arbre de Classification

OJ (ISLR package)
y : #purchase

root (1070)



si $\text{LoyalCH} \geq 0.4829$

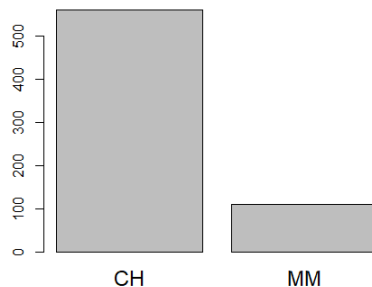
si $\text{LoyalCH} < 0.4829$

$\text{LoyalCH} \geq 0.4829$
CH
653/417

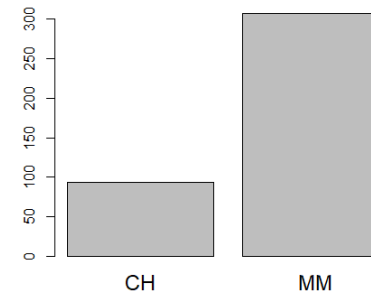
CH
559/110

MM
94/307

node 1 left (669)



node 1 right (401)



CART: prédiction

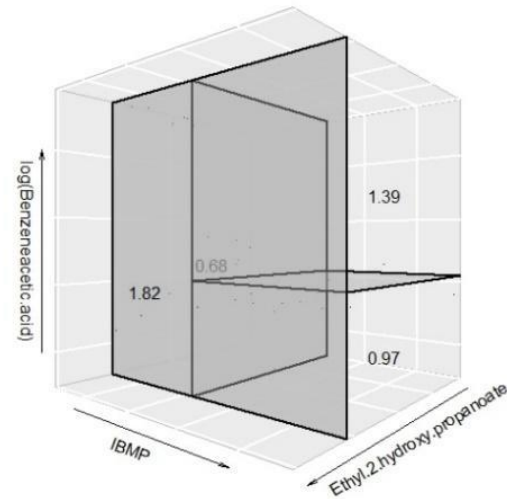
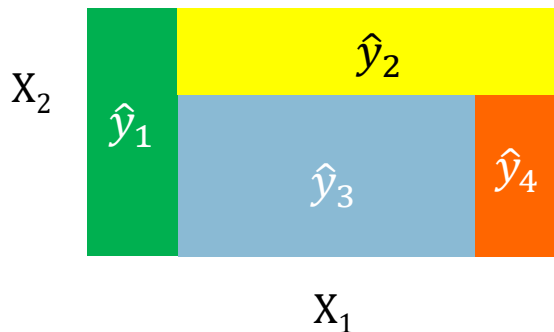
Pour toute (nouvelle) observation, l'arbre définit une règle de décision qui permet de prédire une valeur pour la variable de réponse, y .

Arbre de Classification => catégorie ayant la plus grande fréquence au niveau de la feuille considérée (sur la base du jeu d'apprentissage).

Arbre de Régression => moyenne des valeurs y pour les observations (du jeu d'apprentissage) atteignant une feuille donnée.

CART: prédiction

Un arbre de décision (CART) produit un découpage en régions (hyper-cubes) de l'espace des prédicteurs, et chaque région est associée à une fonction de prédiction constante.



Conséquence:

- Un arbre peu profond, simple, avec peu de nœuds terminaux, n'offrira qu'un nombre limité de prédictions possibles => faible capacité prédictive.
- Un arbre profond, complexe, permettre de prédire parfaitement les observations du jeu d'apprentissage => mais risque de sur-ajustement.

CART: règles d'arrêt

Plusieurs règles sont possibles:

a) Tous les nœuds terminaux sont purs : Les observations atteignant un nœud terminal appartiennent à la même catégorie (y qualitative) ou ont la même valeur pour y (y quantitative).

b1) on fixe un seuil pour le nombre minimal d'observations dans chaque nœud terminal. paramètre *minbucket* (package R "rpart")

ou

b2) on fixe un seuil pour le nombre minimal d'observations dans un nœud pour qu'il puisse être soumis à coupure. paramètre *minsplit* (package R "rpart")
si *minbucket* fixé \Rightarrow $\text{minsplit} = \text{minbucket} * 3$ Si *minsplit* fixé \Rightarrow $\text{minbucket} = \text{minsplit} / 3$

c) Seuil minimal pour le paramètre de complexité ("poids" entre la mesure d'impureté, qui décroît quand l'arbre augmente, et sa profondeur)
cp = *complexity parameter* (package R "rpart"), 0.01 by default.

CART: élagage

Objectif :

construire un arbre T le moins complexe possible (moins de nœuds terminaux)
mais avec la meilleure capacité prédictive possible (évaluation par cross-validation).

Elagage

A partir d'un arbre "complet", T_0 ,

- supprimer les niveaux les plus bas, et
considérer des sous-arbres, T , indexés par un niveau de complexité cp ($\leftrightarrow \alpha$)

Pour chaque valeur de α (nb fini d'intervalles pour les valeurs possibles de α) on détermine l'arbre T ayant le minimum pour :

$$R_{\alpha}(T) = \underbrace{R(T)}_{\text{"Risque" associé à } T \leftrightarrow i(T)} + \underbrace{\alpha |T|}_{\text{"coût" de l'addition d'une nouvelle dichotomie (complexity parameter)}} \rightarrow \# \text{ nœuds terminaux } \leftrightarrow df$$

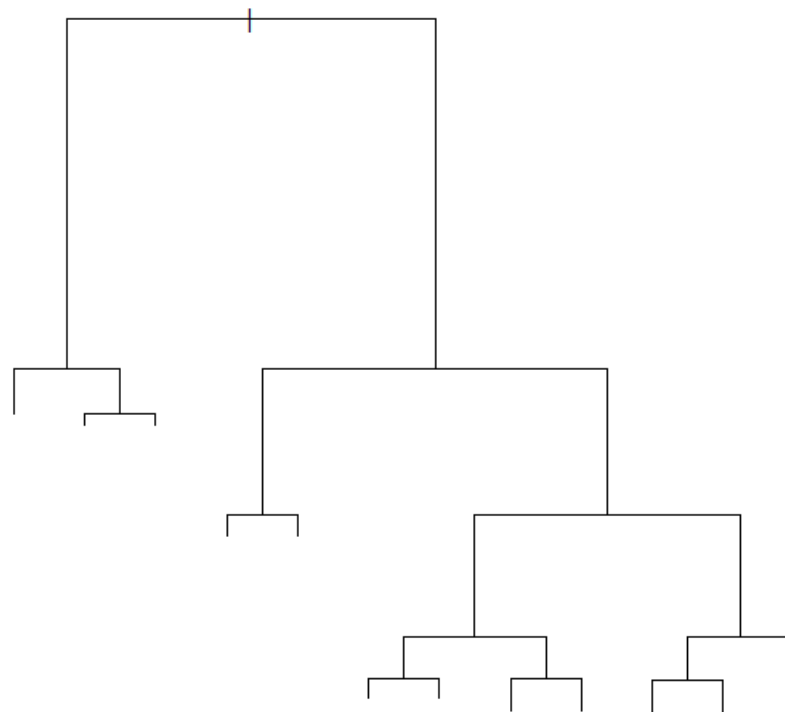
- choisir, par cross-validation, l'arbre T qui fournit l'erreur de prédiction minimale.

CART: élaguage

package R "rpart"

	CP	nsplit	rel.error	xerror	xstd
1	0.331092	0	1.00000	1.00457	0.13982
2	0.138351	1	0.66891	0.71230	0.10209
3	0.115914	2	0.53056	0.80461	0.14069
4	0.042523	3	0.41464	0.68608	0.12685
5	0.040836	4	0.37212	0.67270	0.12708
6	0.037103	5	0.33129	0.65890	0.12760
7	0.031867	6	0.29418	0.72285	0.13561
8	0.029937	7	0.26232	0.70274	0.13368
9	0.019215	8	0.23238	0.68624	0.13083
10	0.018133	9	0.21316	0.69288	0.13196
11	0.010909	10	0.19503	0.66454	0.12905
12	0.010000	11	0.18412	0.67035	0.12921

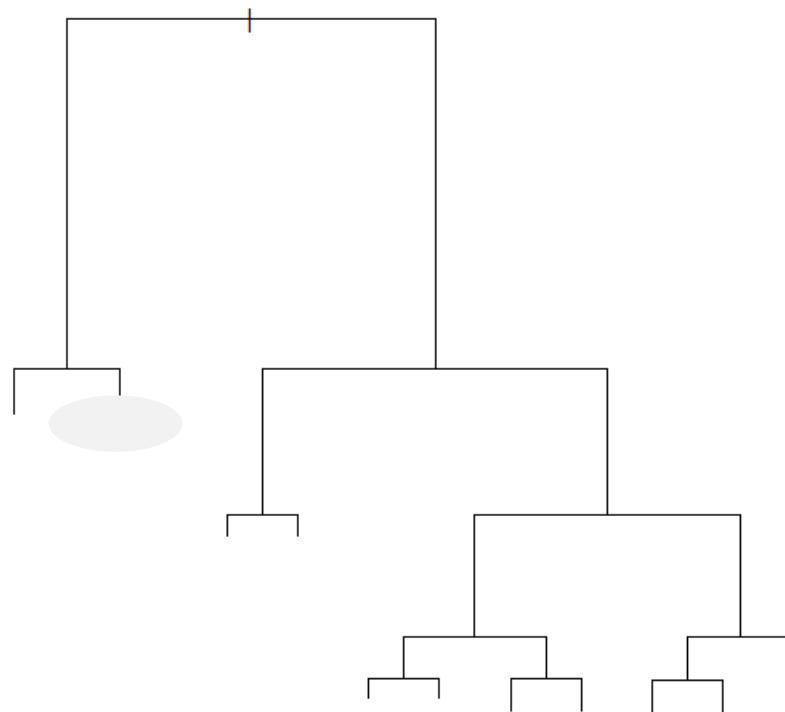
Nb nœuds terminaux



CART: élaguage

package R "rpart"

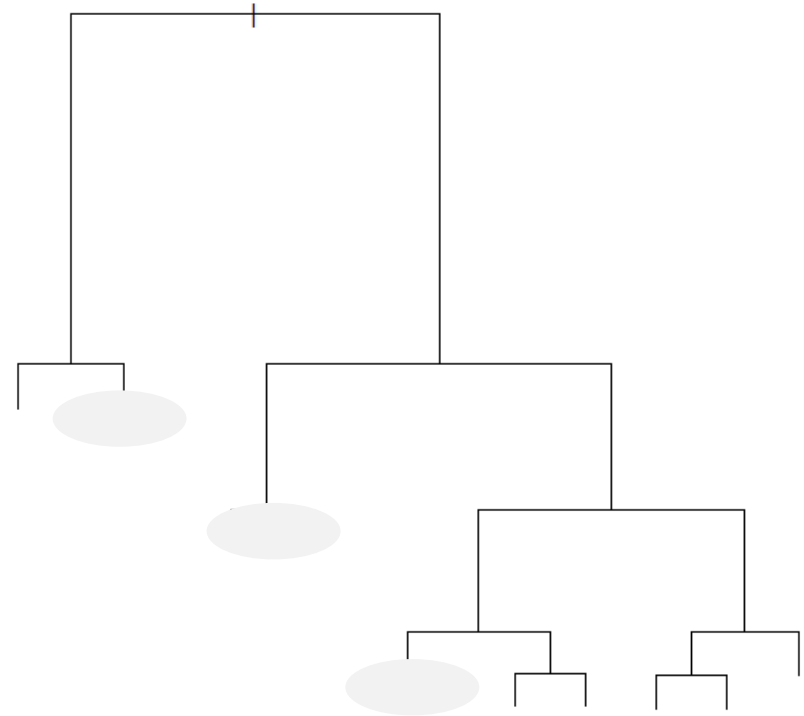
	CP	nsplit	rel.error	xerror	xstd
1	0.331092	0	1.00000	1.00457	0.13982
2	0.138351	1	0.66891	0.71230	0.10209
3	0.115914	2	0.53056	0.80461	0.14069
4	0.042523	3	0.41464	0.68608	0.12685
5	0.040836	4	0.37212	0.67270	0.12708
6	0.037103	5	0.33129	0.65890	0.12760
7	0.031867	6	0.29418	0.72285	0.13561
8	0.029937	7	0.26232	0.70274	0.13368
9	0.019215	8	0.23238	0.68624	0.13083
10	0.018133	9	0.21316	0.69288	0.13196
11	0.010909	10	0.19503	0.66454	0.12905
12	0.010000	11	0.18412	0.67035	0.12921



CART: élaguage

package R "rpart"

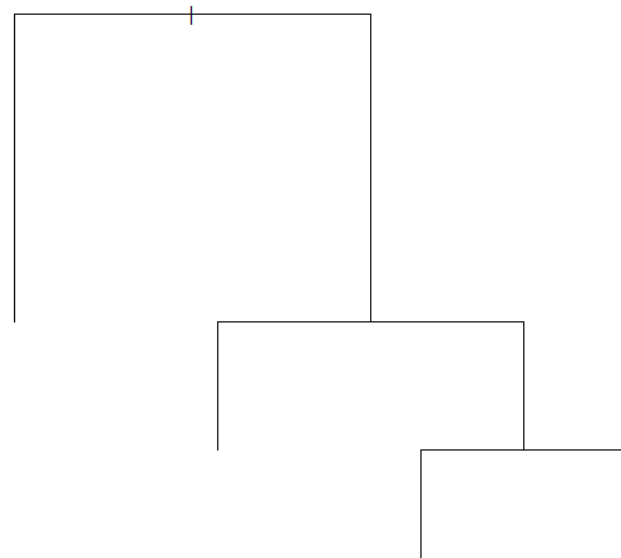
	CP	nsplit	rel.error	xerror	xstd
1	0.331092	0	1.00000	1.00457	0.13982
2	0.138351	1	0.66891	0.71230	0.10209
3	0.115914	2	0.53056	0.80461	0.14069
4	0.042523	3	0.41464	0.68608	0.12685
5	0.040836	4	0.37212	0.67270	0.12708
6	0.037103	5	0.33129	0.65890	0.12760
7	0.031867	6	0.29418	0.72285	0.13561
8	0.029937	7	0.26232	0.70274	0.13368
9	0.019215	8	0.23238	0.68624	0.13083
10	0.018133	9	0.21316	0.69288	0.13196
11	0.010909	10	0.19503	0.66454	0.12905
12	0.010000	11	0.18412	0.67035	0.12921



CART: élaguage

package R "rpart"

	CP	nsplit	rel.error	xerror	xstd
1	0.331092	0	1.00000	1.00457	0.13982
2	0.138351	1	0.66891	0.71230	0.10209
3	0.115914	2	0.53056	0.80461	0.14069
4	0.042523	3	0.41464	0.68608	0.12685
5	0.040836	4	0.37212	0.67270	0.12708
6	0.037103	5	0.33129	0.65890	0.12760
7	0.031867	6	0.29418	0.72285	0.13561
8	0.029937	7	0.26232	0.70274	0.13368
9	0.019215	8	0.23238	0.68624	0.13083
10	0.018133	9	0.21316	0.69288	0.13196
11	0.010909	10	0.19503	0.66454	0.12905
12	0.010000	11	0.18412	0.67035	0.12921



Ex. CART

R package rpart

Hitters
y: annual salary

```
>library(rpart)
```

```
# arbre CART
```

```
>fit.rpart <- rpart(Salary~ ., data=DATA,  
control=rpart.control(minsplit=10, cp=0.01, xval=10),)
```

"Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. For instance, with anova splitting, this means that the overall R-squared must increase by cp at each step."

Lower cp values return deeper trees.

"number of cross-validations."
Herein, used in order to prune the tree.

Ex. CART

R package rpart

Hitters
y: annual salary

```
> printcp(fit.rpart)
```

Regression tree:

```
rpart(formula = Salary ~ ., data = DATA, control = rpart.control(minsplit = 10,  
  cp = 0.01, xval = 10))
```

Variables actually used in tree construction:

CatBatbyYear CHitsbyYear CRBIbyYear HmRun Walks Years

Root node error: $50624071/261 = 193962$ **SCEy/n**
n= 261

	CP	nsplit	rel error of the model
1	0.331092	0	1.00000
2	0.138351	1	0.66891
3	0.115914	2	0.53056
4	0.042523	3	0.41464
5	0.040836	4	0.37212
6	0.037103	5	0.33129
7	0.031867	6	0.29418
8	0.029937	7	0.26232
9	0.019215	8	0.23238
10	0.018133	9	0.21316
11	0.010909	10	0.19503
12	0.010000	11	0.18412

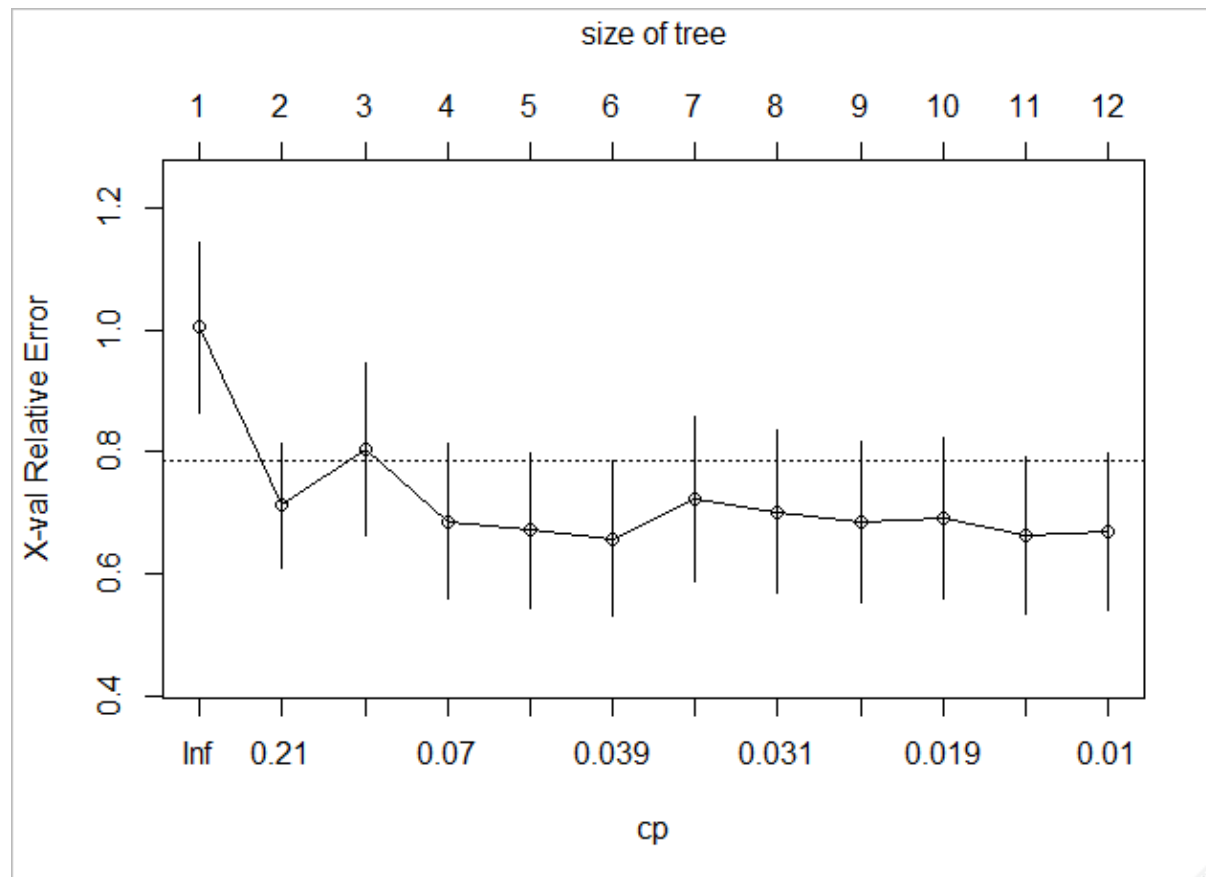
xerror CV	xstd CV
1.00457	0.13982
0.71230	0.10209
0.80461	0.14069
0.68608	0.12685
0.67270	0.12708
0.65890	0.12760
0.72285	0.13561
0.70274	0.13368
0.68624	0.13083
0.69288	0.13196
0.66454	0.12905
0.67035	0.12921

Ex. CART

R package rpart

Hitters
y: annual salary

```
> plotcp(fit.rpart)
```

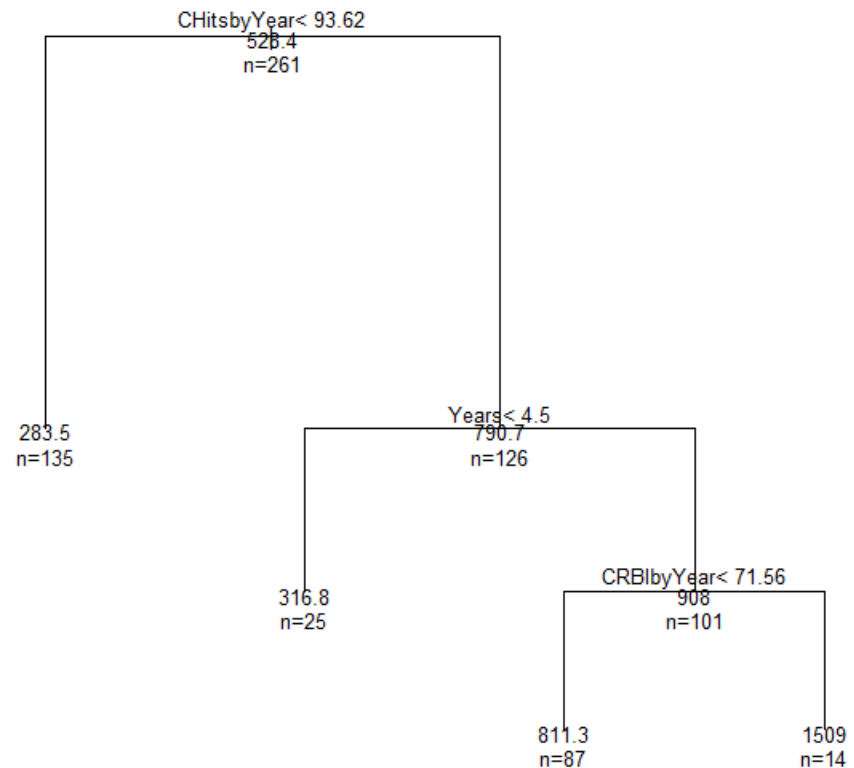


Ex. CART après élaguage

Hitters
y: annual salary

```
> choix=4  
> cp = fit.rpart$cp[choix,"CP"]  
> fit.rpart.prune = prune(fit.rpart, cp=cp)
```

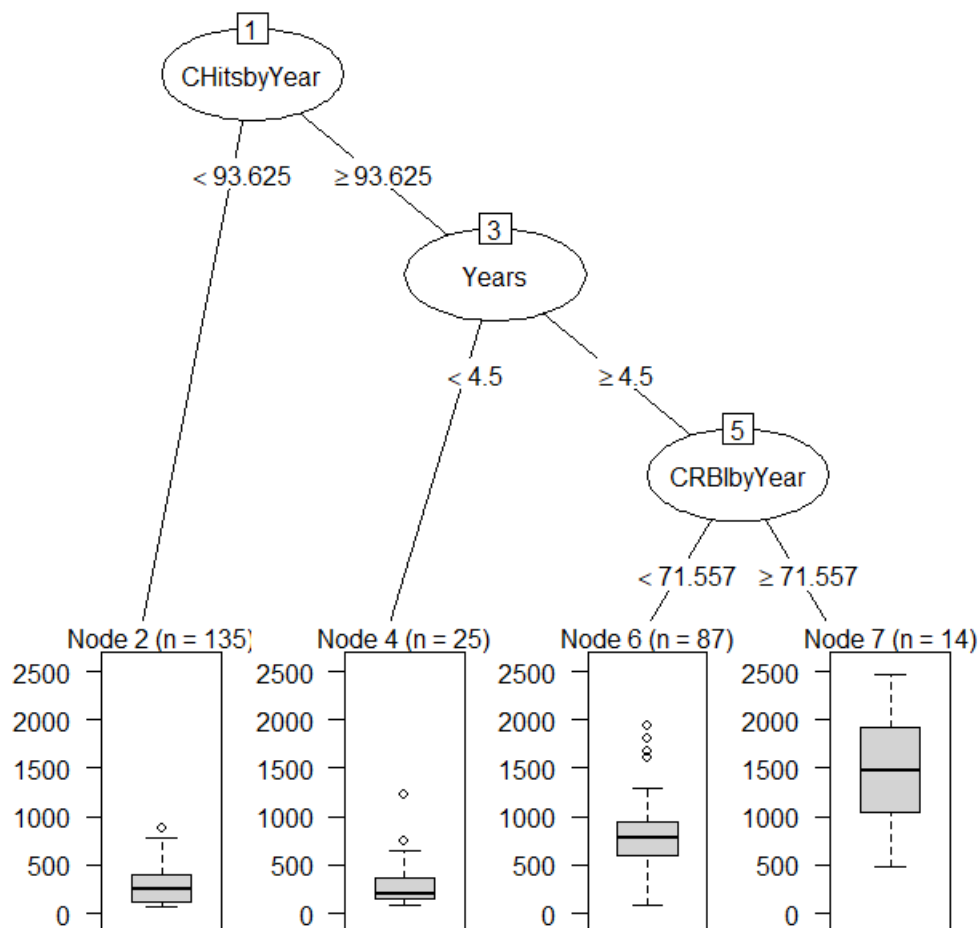
*function "plot"
in "rpart" package*



Ex. CART après élaguage

Hitters
y: annual salary

Le même arbre que précédemment mais avec la fonction "plot" du package "partykit".



Arbres de décision

Points positifs

- 😊 Aspect graphique,
- 😊 Interprétabilité du modèle,
- 😊 Fournit d'une règle de décision,
- 😊 Aptitude à tenir compte d'éventuels relations non-linéaires et effets d'interaction,
- 😊 Possibilité d'utiliser des prédicteurs qualitatifs et quantitatifs (attention au biais de sélection. Strobl et al. 2007) ,
- 😊 Pas d'hypothèses sur les distributions des variables,
-

Cependant

- Un petit nombre seulement de prédicteurs sont utilisés comme variables de coupure,
- 👉 MAIS d'autres prédicteurs pourraient aussi donner des résultats de qualité équivalente (variables concurrentes)
- 👉 Un modèle de type arbre peut être différent s'il y a de petites modifications dans le jeu de données (instabilité),
- 👉 La structure de l'arbre peut être fortement modifiée s'il y a un changement pour le 1^{er} niveau.

Par conséquent

- 👉 Prédictions entachées d'une forte variabilité.

Ex. CART : variables concurrentes

Hitters
y: annual salary

variables concurrentes au premier noeud

Node number 1: 261 observations, complexity param=0.3310922
mean=528.357, MSE=193962
left son=2 (135 obs) right son=3 (126 obs)
Primary splits:

CHitsbyYear	< 93.625	to the left,	improve=0.3310922,	(0 missing)
CAtBatbyYear	< 340.1875	to the left,	improve=0.3004466,	(0 missing)
CRunsbyYear	< 58.63333	to the left,	improve=0.2981556,	(0 missing)
Years	< 4.5	to the left,	improve=0.2891033,	(0 missing)
CRBIbyYear	< 48.28571	to the left,	improve=0.2890407,	(0 missing)

Variation relative
d'impureté. Ici
$$\left(\frac{MSE_0 - MSE_1}{MSE_0} \right)$$

variables concurrentes au nœud-fils n°3

Node number 3: 126 observations, complexity param=0.1383506
mean=790.6667, MSE=230605.7
left son=6 (25 obs) right son=7 (101 obs)
Primary splits:

Years	< 4.5	to the left,	improve=0.2410447,	(0 missing)
Walks	< 66.5	to the left,	improve=0.1889800,	(0 missing)
CRBIbyYear	< 69.11538	to the left,	improve=0.1593549,	(0 missing)
RBI	< 78.5	to the left,	improve=0.1486289,	(0 missing)
CRunsbyYear	< 58.63333	to the left,	improve=0.1026206,	(0 missing)

Ex. CART : variables concurrentes

OJ
y: #Purchase

variables concurrentes au premier noeud

```
Node number 1: 1070 observations,      complexity param=0.5107914
predicted class=CH expected loss=0.3897196 P(node) =1
class counts:    653    417
probabilities: 0.610 0.390
left son=2 (669 obs) right son=3 (401 obs)
```

Primary splits:

```
LoyalCH      < 0.48285   to the right, improve=181.21710, (0 missing)
STORE        splits as LRRRL, improve= 52.93961, (0 missing)
PriceDiff    < 0.015    to the right, improve= 27.89651, (0 missing)
SalePriceMM  < 1.84     to the right, improve= 24.01200, (0 missing)
ListPriceDiff < 0.255    to the right, improve= 22.67383, (0 missing)
```

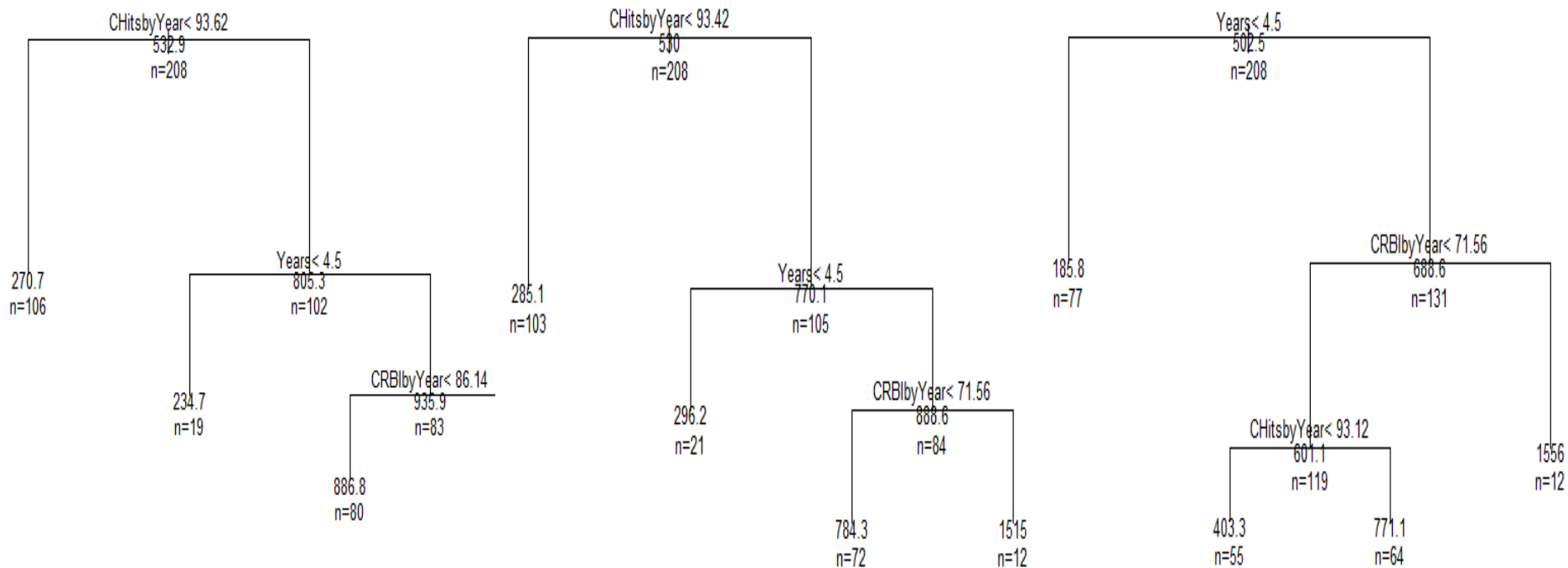
*STORE=0 ou 4 → branche gauche,
STORE=1, 2, 3 → branche droite*

n * variation d'impureté (Gini):
 $n \{ i(\text{root}) - p_L * i(L) - p_R * i(R) \}$

Ex. CART : instabilité des arbres

Hitters
y: annual salary

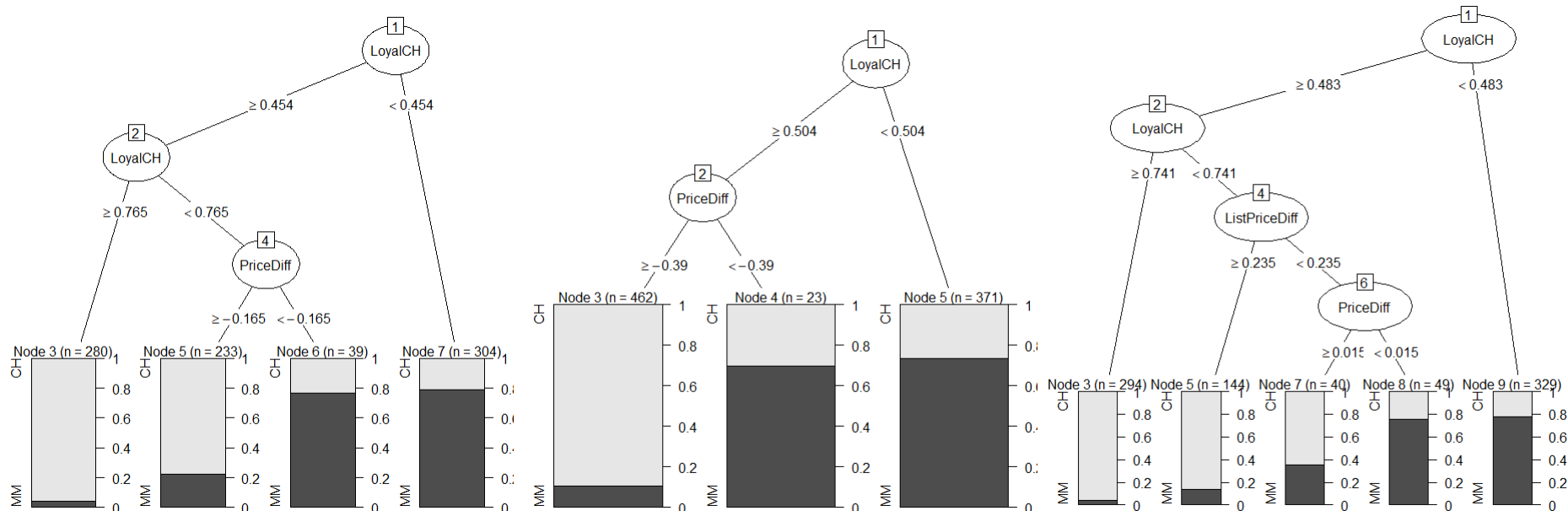
80% du jeu de données est sélectionné au hasard
4 nœuds terminaux considérés



Ex. CART : instabilité des arbres

80% du jeu de données est sélectionné au hasard
zieme niveau de complexité considéré.

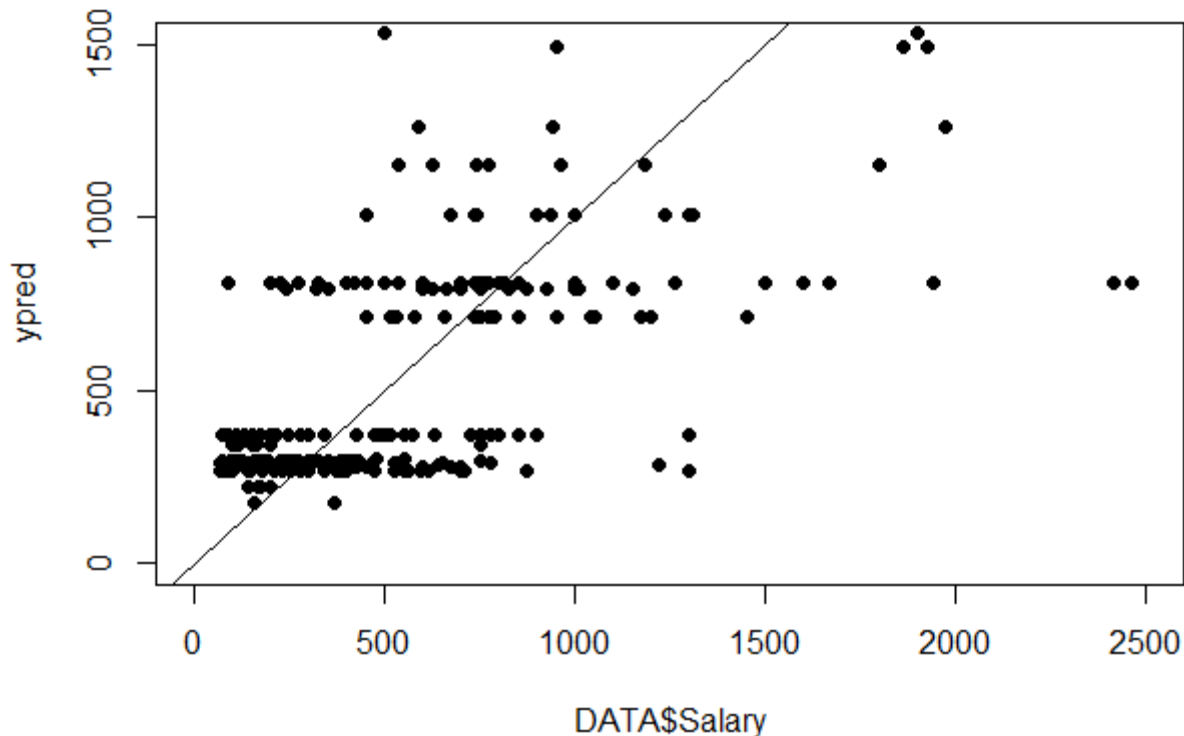
OJ
y: #Purchase



Ex. CART : Prediction (CV)

Jeu de données découpé en 5 segments (intercalés)
4 noeuds terminaux

Hitters
y: annual salary



$RMSE_{CV} = 347.25$

$RMSE_{\text{null model}} = 440.41$
 $RMSE_{CV_null \text{ model}} = 442.58$

Ex. CART : Prédiction (CV)

OJ
y: #Purchase

Jeu de données découpé en 5 segments (intercalés)
3ieme niveau de complexité considéré

Purchase	<i>classpred</i>		<i>class error</i>
	1	2	
CH	540	113	17.3%
MM	90	327	21.6%

Taux d'erreur 19.0%

Rand= 0.69

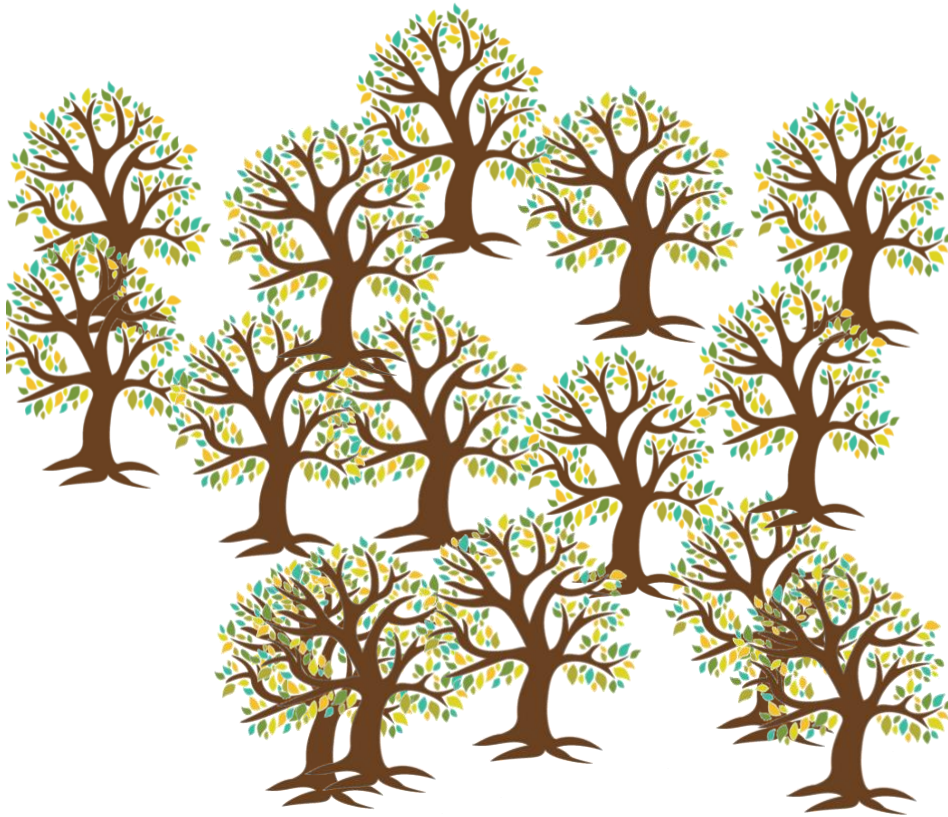
AdjustedRand=0.38

Taux erreur_"modèle constant" (root) =39.0%

Approches basées sur des ensembles d'arbres



Forêts Aléatoire (Breiman, 2001)



Une forêt
aléatoire est un
ensemble d'arbres
de décision
(*e.g.* arbre CART)

Forêts aléatoires = *Random Forests*

Random

- Randomisation des observations

B échantillons *Bootstrap*

Bagging (bootstrap aggregation) Leo Breiman, 1996

But: Améliorer la précision des prédictions par agrégation

- Sélection au hasard d'un sous-ensemble de prédicteurs, à chaque nœud, et choix de la variable de coupure au sein de ce sous-ensemble.

paramètre *mtry* : # variables dans le sous-ensemble

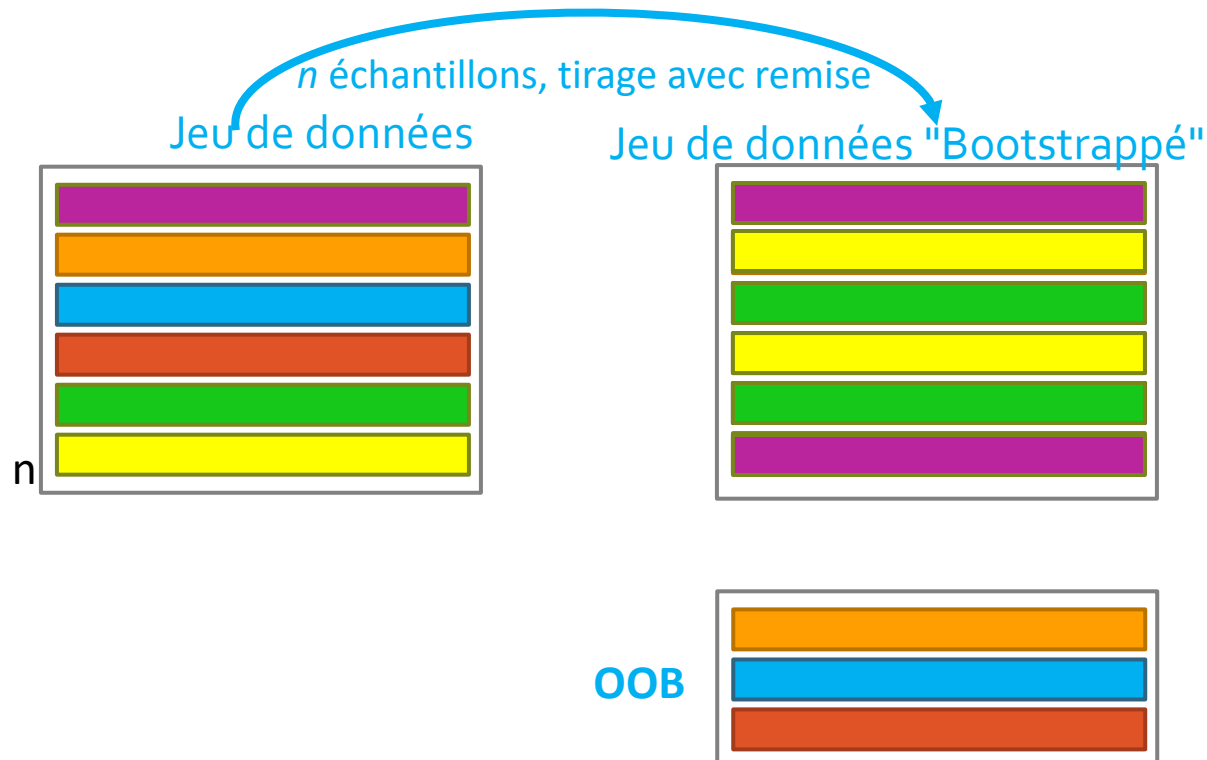
But: rendre les arbres de la forêt le plus indépendants possible

Algorithme de construction d'une forêt aléatoire

- ✓ Pour $b=1, \dots, B = \text{ntree}$ (# arbres dans la forêt)
 - Sélection d'un **échantillon Bootstrap** (tirage avec remise de n parmi n)
 - Construction d'un arbre CART maximal (sans élaguage) avec
A chaque nœud, choix de la meilleure variable candidate pour la coupure du jeu de données parmi q variables sélectionnées **au hasard** parmi les p prédicteurs (par défaut, $q = \text{mtry} = p/3$ pour les arbres de régression, \sqrt{p} pour les arbres de classification).
- ✓ Collecte de tous les arbres (toutes les règles de décision) de la forêt
➔ prédiction finale: valeur moyenne/classe majoritaire observée sur l'ensemble des prédictions individuelles.

Observations Out Of Bag (OOB)

Pour chaque arbre, ré-échantillonnage « bootstrap »: → obs. sélectionnées
→ obs. OOB



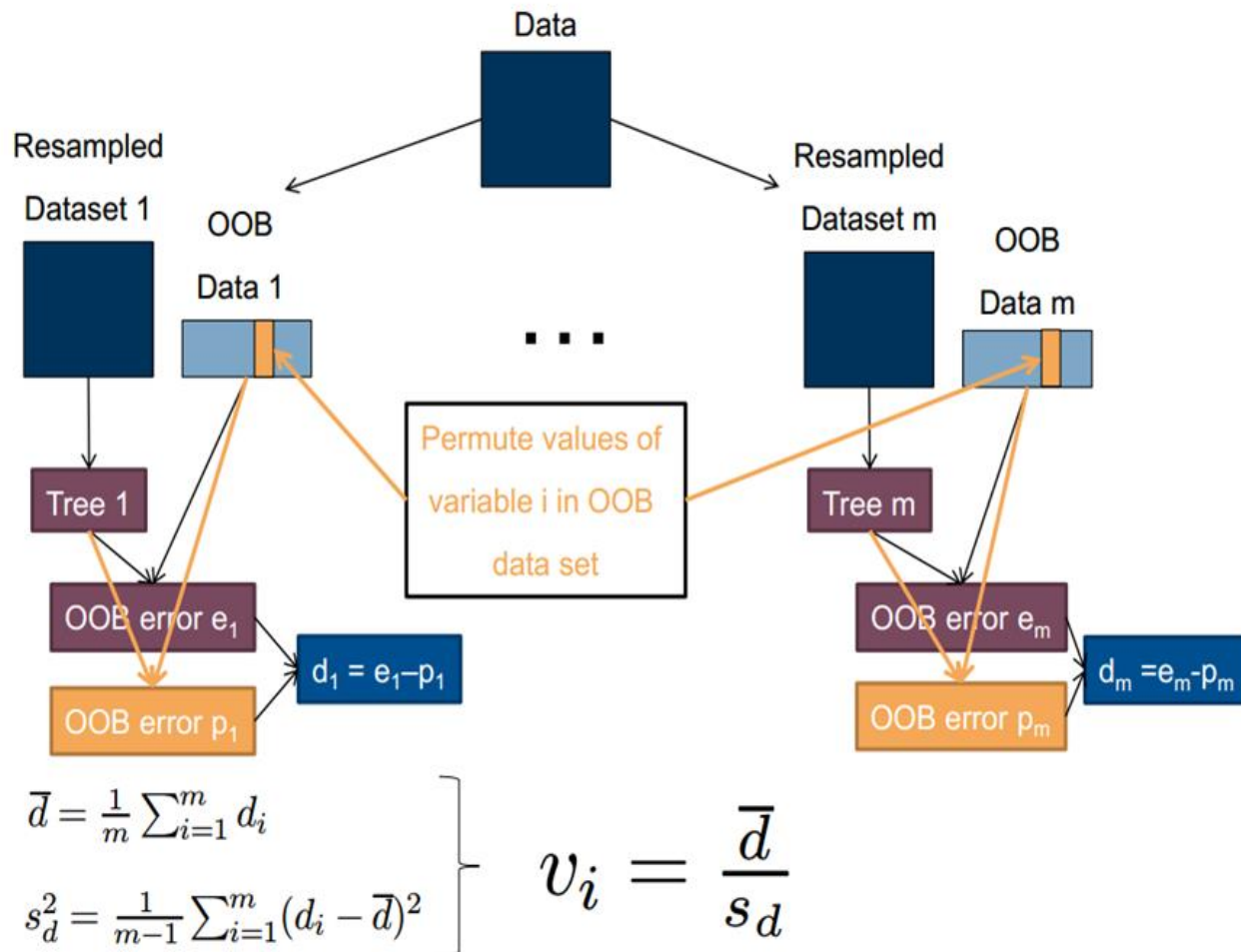
Rq: La probabilité qu'une observation appartienne au sous-ensemble OOB est environ de $1/e = 0.368$ ($n \rightarrow \infty$)

Observations Out Of Bag (OOB)

D'un grand intérêt pour:

- Estimer l'erreur de prédiction : OOB error
- Evaluer l'importance des variables (V.I.),
à l'aide d'un indicateur tel que le "MDA" (Mean
Decrease in Accuracy)

Erreur OOB et critère V.I.



Plus d'écart d est grand, plus la variable est "importante"

Ex. Forêt Aléatoire

Hitters
y: annual salary

R Package "randomForest"

```
> valntree=2000
> valmtry= max(floor(p/3))
> valnodesize=5    # by default 5 if y quantitative, but not very sensible
> rdf=randomForest(formula=Salary~., data=DATA,
                    ntree=valntree, mtry=valmtry, nodesize = valnodesize,
                    importance=TRUE)
```

Ex. Forêt Aléatoire

Hitters
y: annual salary

Call:

```
randomForest(formula = Salary ~ ., data = DATA, ntree = 2000, mtry = 4,  
nodesize = 5, importance = TRUE, proximity = TRUE, nPerm = 1)
```

Type of random forest: regression

Number of trees: 2000

No. of variables tried at each split: 4

Mean of squared residuals: 69776.65

% Var explained: 64.03

$$\text{MSE}_{\text{OOB}} = n^{-1} \sum_1^n \{y_i - \hat{y}_i^{\text{OOB}}\}^2,$$

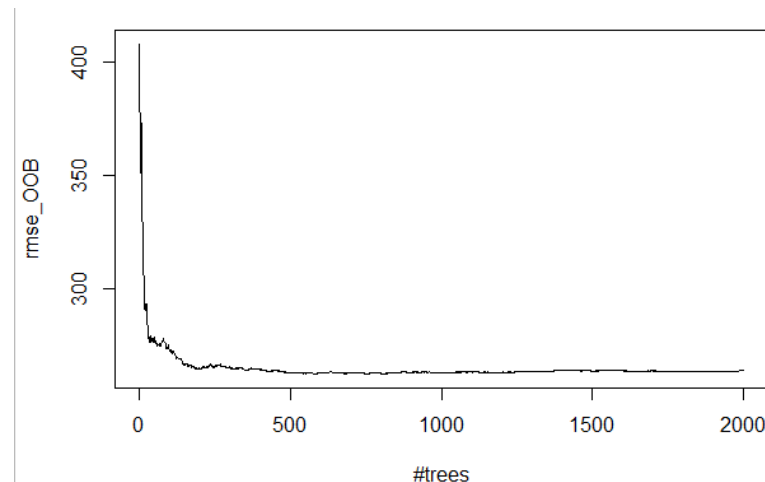
$$1 - \frac{\text{MSE}_{\text{OOB}}}{\hat{\sigma}_y^2},$$

Ex. Forêt Aléatoire

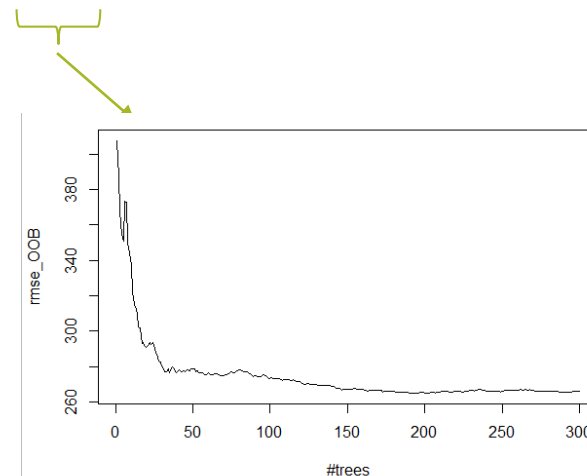
Hitters
y: annual salary

Evolution du RMSE_OOB en fonction de nb d'arbres dans la forêt

Dans ce cas, il n'est pas forcément nécessaire de construire 2000 arbres. 200-300 arbres, par ex., seraient suffisants.



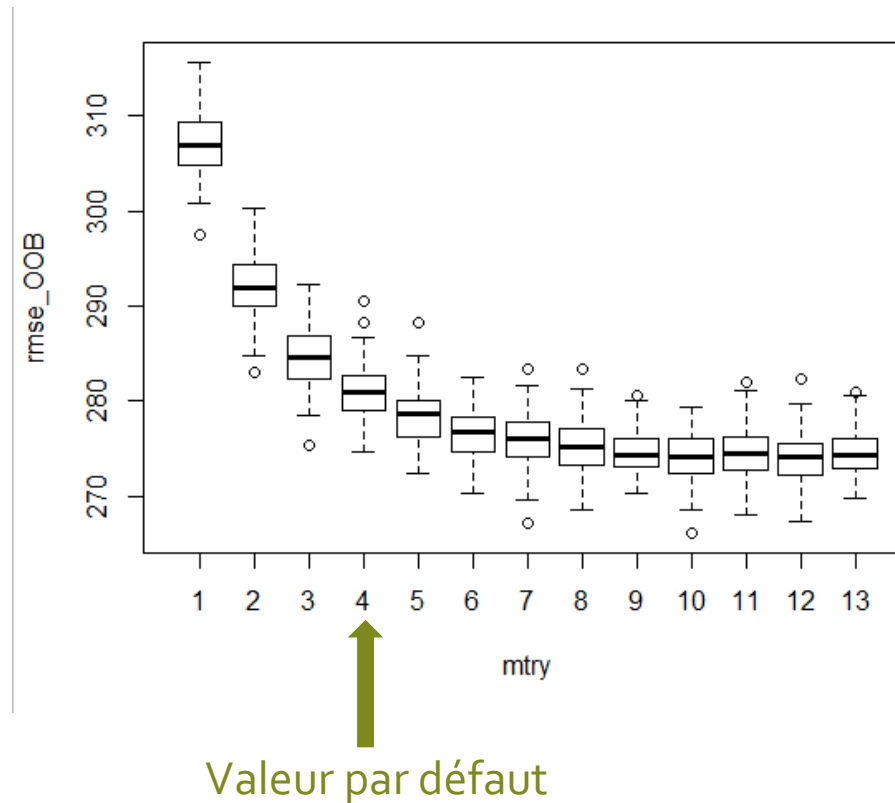
Remarque : avec trop peu d'arbres (<10 par ex), prédictions médiocres, et grande sensibilité aux observations appartenant aux jeux OOB .



Ex. Forêt Aléatoire

Hitters
y: annual salary

Evolution du RMSE_OOB en fonction du nb de variables sélectionnées au hasard à chaque noeud (paramètre *mtry*)



boxplots pour
100 essais

Ex. Forêt Aléatoire

Hitters
y: annual salary

Forêt aléatoire avec : $n_{tree}=300$ et $m_{try}=p/3=4$

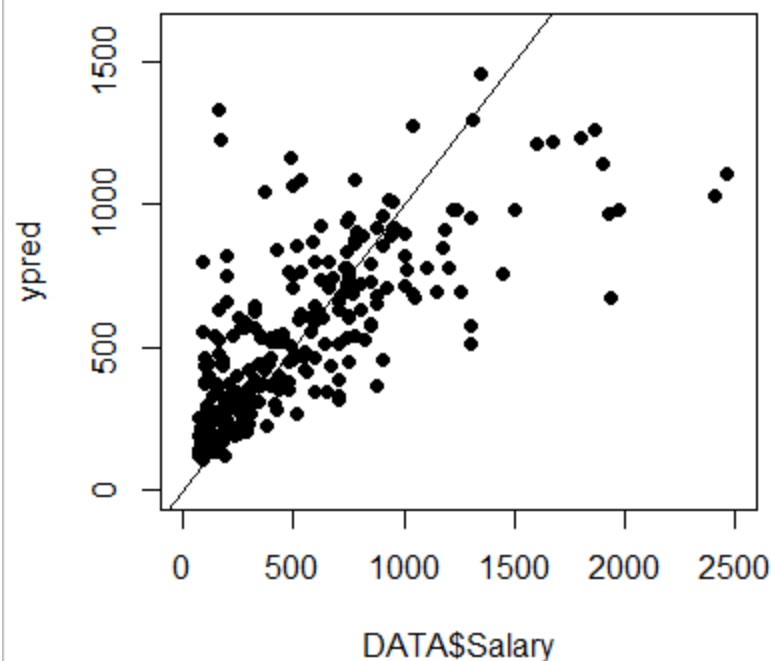
$RMSE = 121.28$ (learning set)

$RMSE_{OOB} = 284.44$



$RMSE_{CV}^* = 292.53$

* 5 segments



Rappel

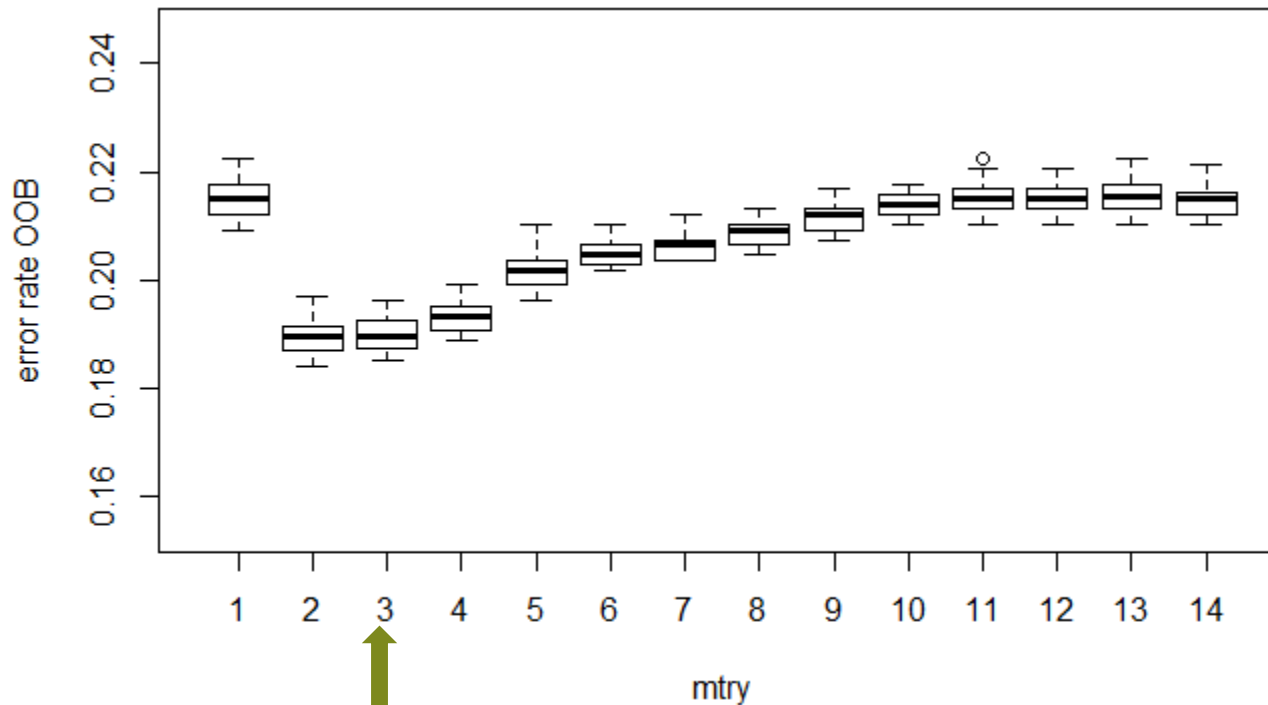
Arbre CART : $RMSE_{CV} = 347.25$

modèle "constant" : $RMSE_{CV_0} = 442.58$

Ex. Forêt Aléatoire

OJ
y: #Purchase

Evolution du taux d'erreur _OOB en fonction du nb de variables sélectionnées au hasard à chaque noeud (paramètre *mtry*)



boxplots pour
100 essais

Valeur par défaut

Ex. Forêt Aléatoire

OJ
y: #Purchase

Forêt aléatoire avec : $n_{tree}=500$ et $m_{try}=\sqrt{p}=3$

Err.rate = 10.0% (learning set)

Err.rate_{OOB} = 18.8%



Confusion matrix:

	<i>classpred</i>		<i>class.error</i>
	<i>CH</i>	<i>MM</i>	
<i>CH</i>	562	91	13.9%
<i>MM</i>	110	307	26.4%

Err.rate_{CV}* = 19.9%

* 5 segments

Rappel

Arbre CART : Err.rate_{CV} = 19.0%

modèle "constant" : Err.rate = 39.0%

Forêt Aléatoire- V.I.

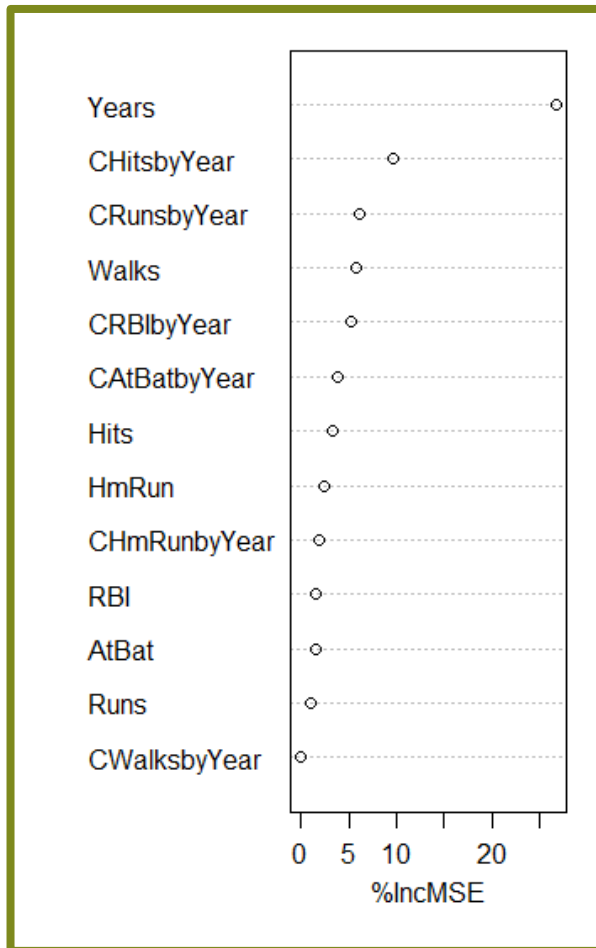
Variable Importance:
un critère pour la sélection de
variables

Ex. Forêt Aléatoire- V.I.

ntree=300 mtry=p/3=4

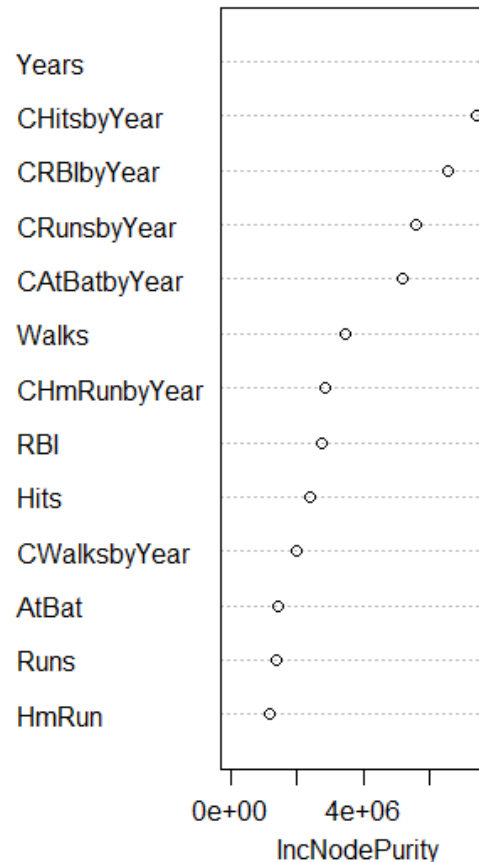
Hitters
y: annual salary

Mean Decrease in Accuracy

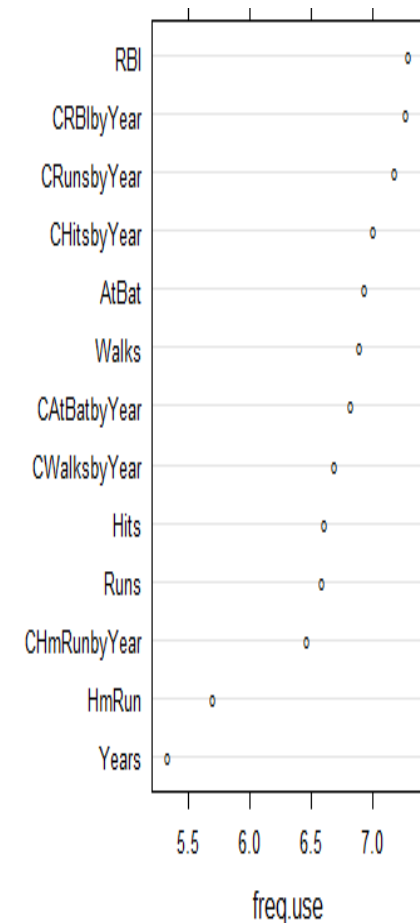


Mean Decrease in node Impurity

=somme des $\Delta i(s, t)$ pour tous les arbres



Frequency of use



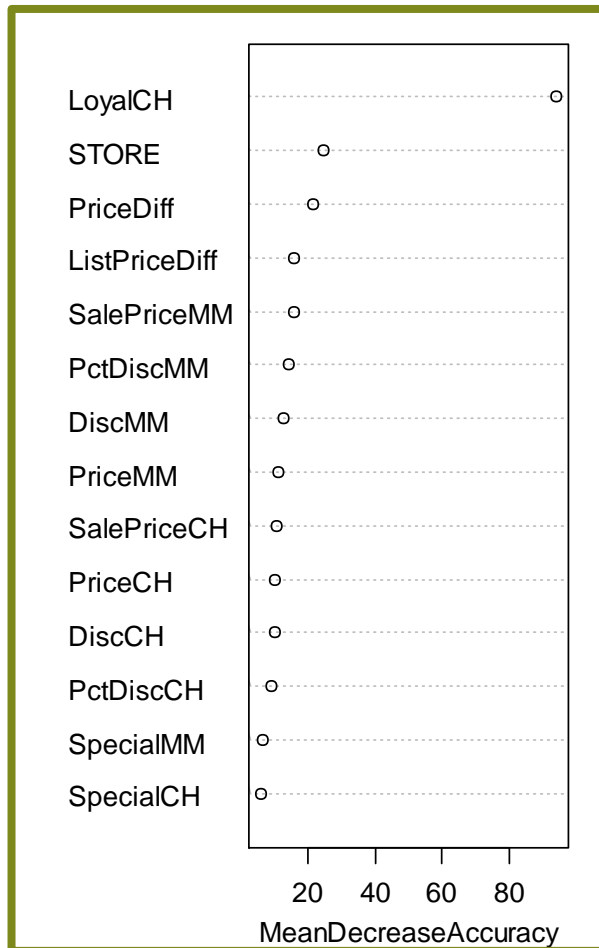
Ex. Forêt Aléatoire- V.I.

ntree=500 mtry=sqrt(p)=3

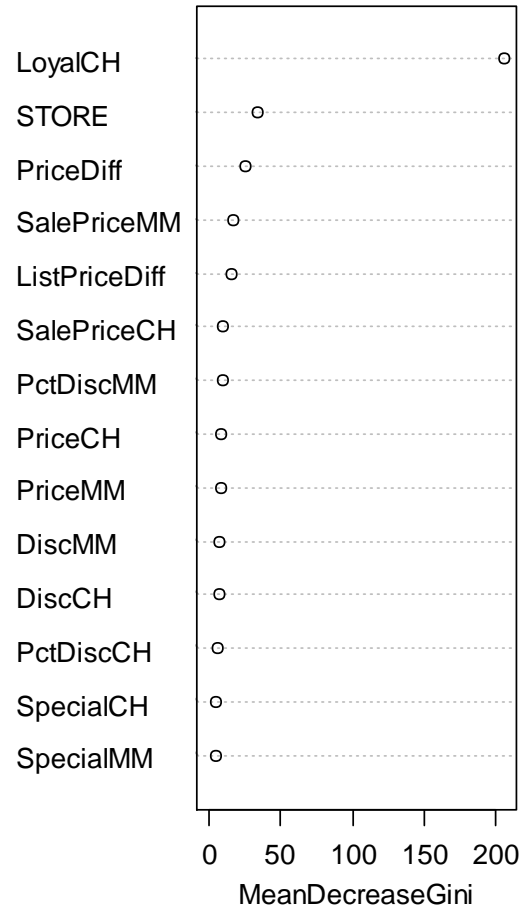
OJ

y: #Purchase

Mean Decrease in Accuracy



Mean Decrease Gini



Frequency of use



Stratégie pour la sélection de variables

Proposition

- Step 1** - Construction d'une forêt aléatoire avec $ntree$ arbres ($mtry$ fixé)
- Répétition de la procédure => $R(=50)$ forêts construites
 - Classement des variables en fonction de leur importance (MDA)
 - Pré-sélection de K variables

Stratégie pour la sélection de variables

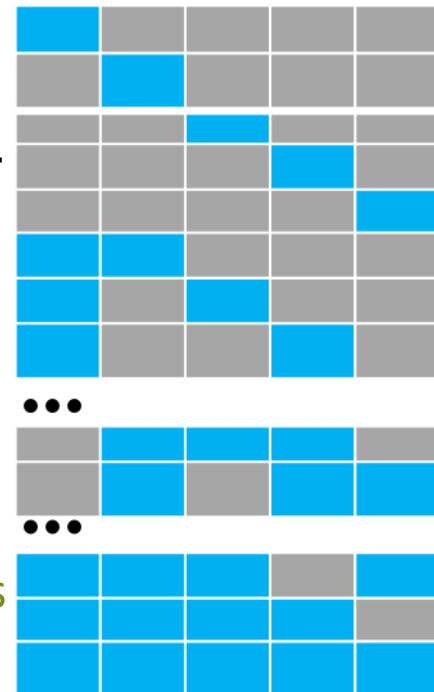
Proposition

- Step 1** - Construction d'une forêt aléatoire avec $ntree$ arbres ($mtry$ fixé)
- Répétition de la procédure => $R(=50)$ forêts construites
 - Classement des variables en fonction de leur importance (MDA)
 - Pré-sélection de K variables

- Step 2** Pour chaque sous-ensemble "s" of 1, 2, 3... variables, prises parmi les K variables pré-sélectionnées.

- Construction d'une forêt à partir de "s"
- Evaluation de $RMSE/Err_{OOB,s}$ et/ou $RMSE/Err_{CV,s}$

Sélection du meilleur sous-ensemble de variables



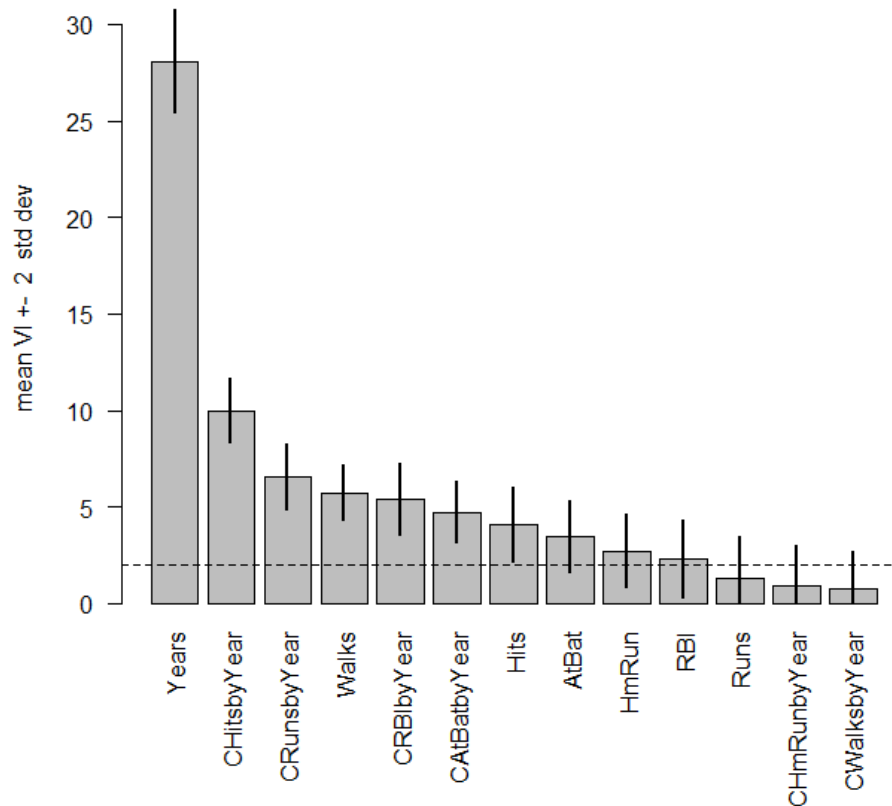
Ex. Forêt Aléatoire -sélection de variables

Hitters
y: annual salary

Step ①

ntree=300 mtry=p/3=4

Moyenne et intervalle de confiance (à ± 2 écart-types) du MDA (Mean Decrease in Accuracy) pour 50 répétitions (50 forêts)



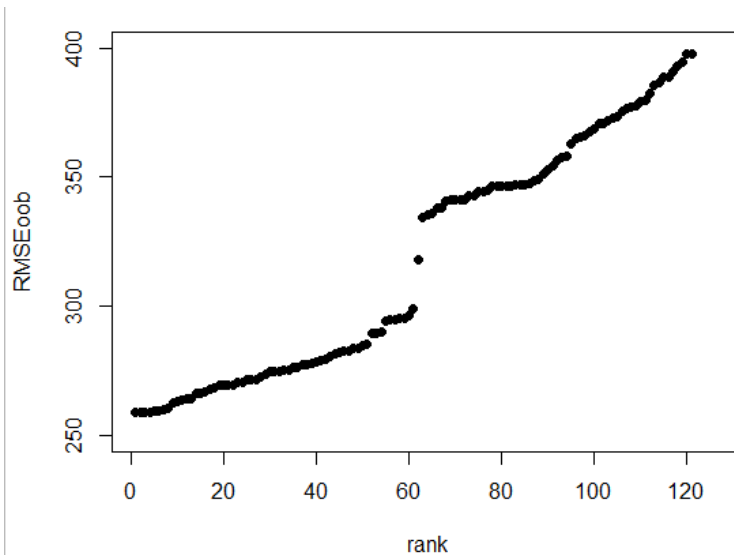
Ex. Forêt Aléatoire - sélection de variables

Hitters
y: annual salary

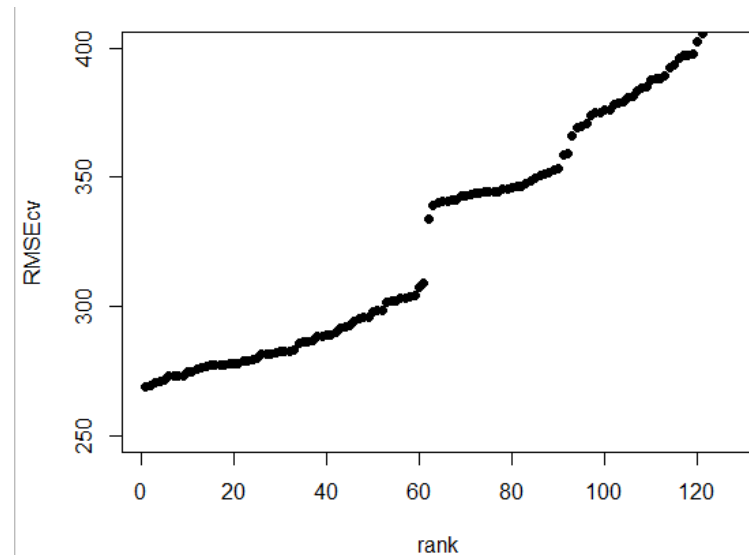
Step ②

pré-sélection de 7 variables \Rightarrow 127 combinaisons

RMSE_{OOB}



RMSE_{CV}



Ex. Forêt Aléatoire - sélection de variables

Hitters
y: annual salary

Step 2 Liste des meilleures combinaisons, sur la base du $RMSE_{CV}$ (avec pré-sélection)

```
=====
"Years"          "CHitsbyYear" "Walks"          "CRBIbyYear"
269.1662

"=====
"Years"          "Walks"          "CRBIbyYear" "Hits"
269.4038

=====
"Years"          "Walks"          "CRBIbyYear"  "CAtBatbyYear" "Hits"
270.7624

=====
Years"          "Walks"          "CRBIbyYear"  "CAtBatbyYear"
270.9282
... •
```

Avec les 7 variables les plus "importantes"

$RMSE_{CV}=279.10$

Avec les 13 variables de départ

$RMSE_{CV}=292.53$

Stratégie pour la sélection de variables

VSURF (Genuer *et al.*, 2010 and 2015)

Stratégies en plusieurs étapes :

- ❶ Classement des variables en fonction de leur mesure d'importance et élimination des variables "non importantes".

VI : (not standardized) MDA. Average over 50 RdF runs.

Elimination of the unimportant variables based on the ordered sequence of the standard deviation of the VI. A new RdF is built with the sd as Y and ranks as X. Determination of the minimum prediction value=> q variables retained.

- ❷ Sélection de variables

- ❷a. Dans une optique d'**interprétation**:

From the ordered list of the variables (mean VI value), 25-50 (typically) RdF are built.

For nested models (from 1 to q), chosen the list leading to the smallest OOB error.

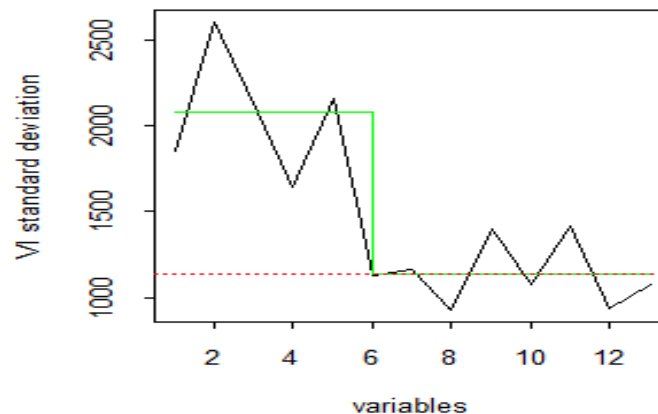
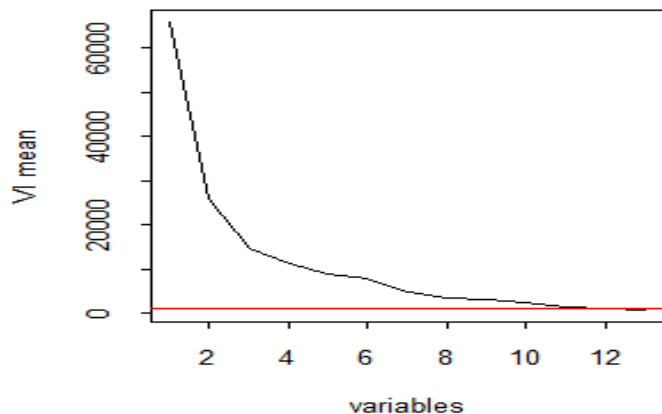
- ❷b. Dans une optique de **prédiction**:

From the ordered list of the variables (mean VI value), 25-50 (typically) RdF are built.

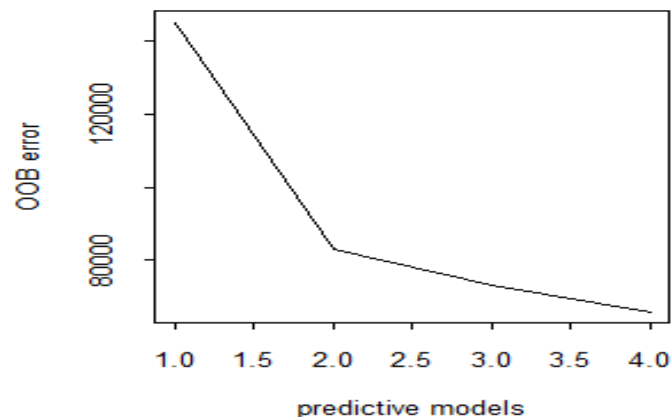
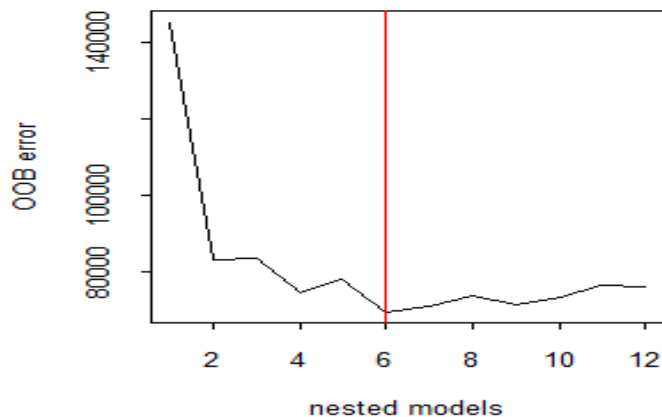
Variables are introduced sequentially, and a new variable is added if the OOB error decrease is significantly greater than a threshold.

Ex. VSURF

Hitters
y: annual salary



`$vselect.thres: 12 variables ("CWalksbyYear" exclu)`



`vselect.interp : 6 var`

"Years", "CHitsbyYear", "CRunsbyYear",
"CRBIbyYear", "CAtBatbyYear", "Walks"

`vselect.pred : 4 var`

"Years", "CHitsbyYear",
"CRBIbyYear", "Walks"

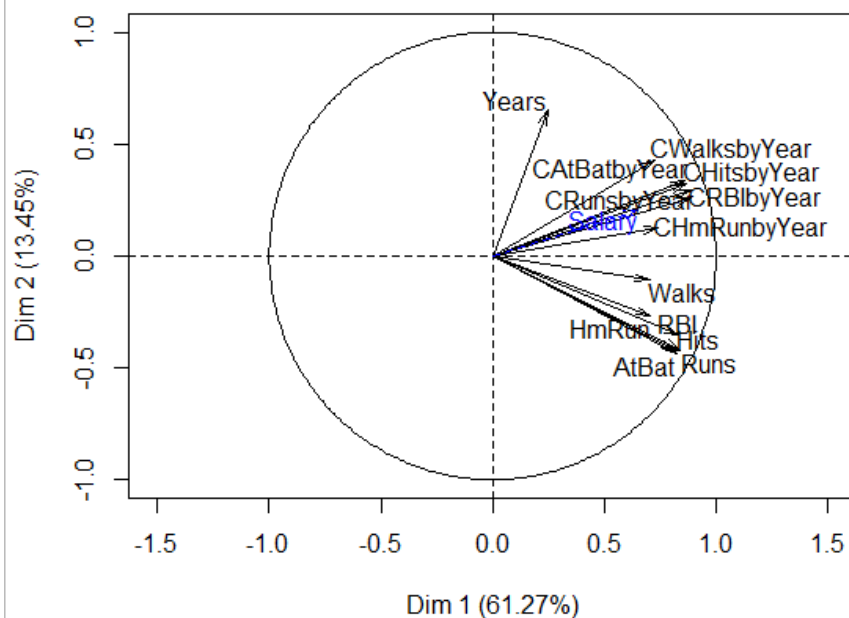
idem

Ex. Analyse descriptive

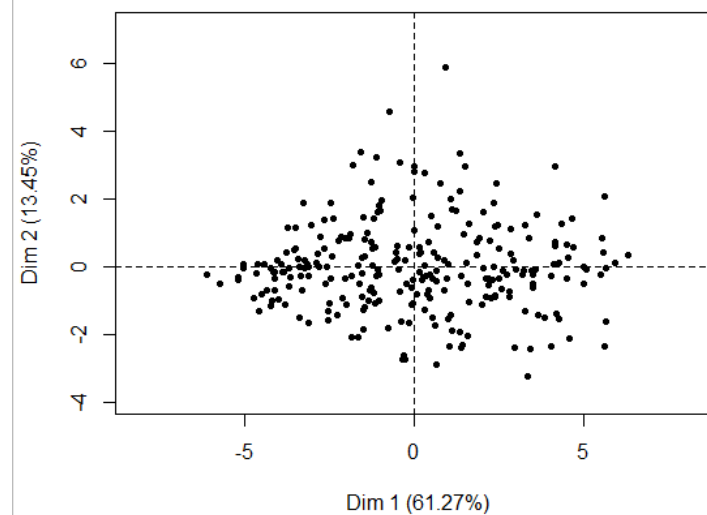
Hitters
y: annual salary

ACP avec les 13 variables de départ

Variables factor map (PCA)



Individuals factor map (PCA)



Ex. arbre de régression avec les 4 variables retenues

Hitters
y: annual salary

Interprétation

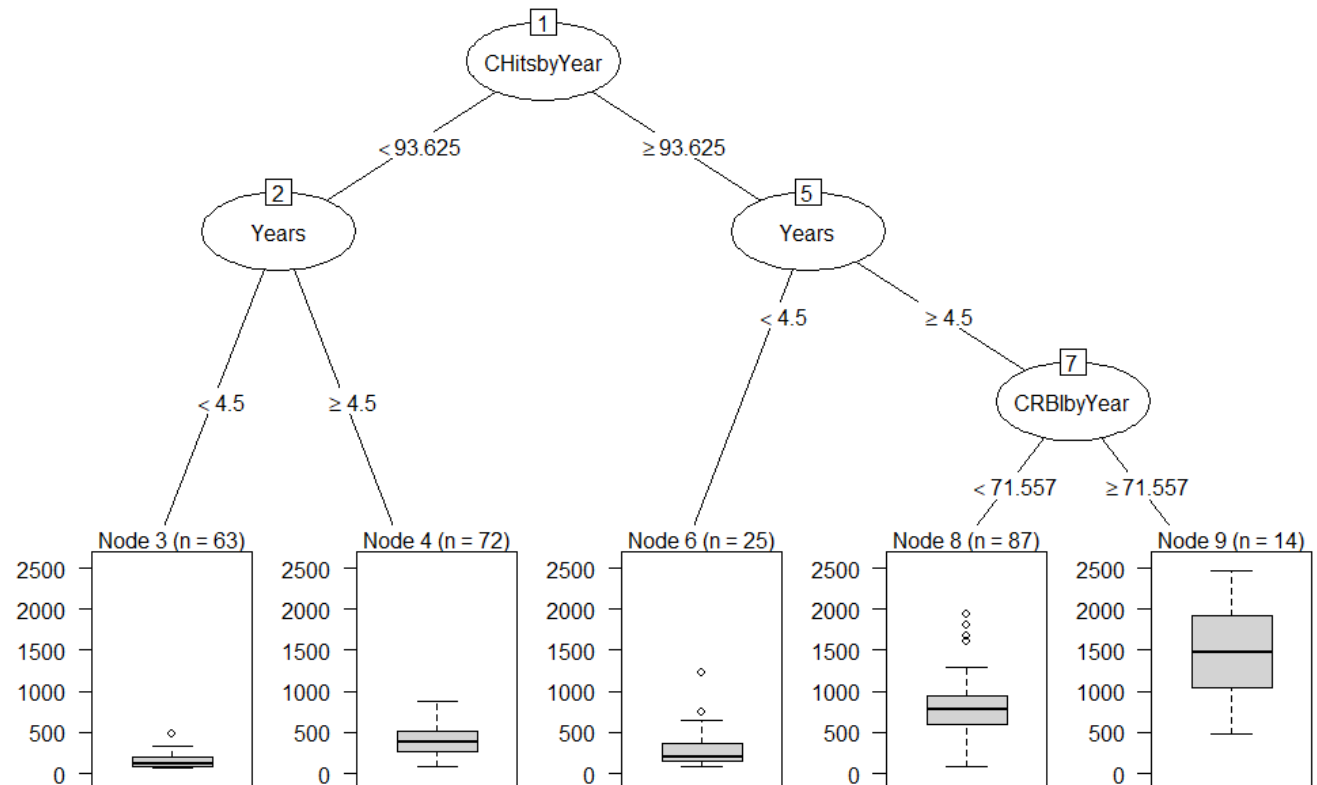


Schéma susceptible de varier légèrement en fonction des paramètres utilisés pour l'élagage et le jeu d'apprentissage

Ex. comparaison CART et RdF

Prédiction

CART , 13 variables

$RMSE_{CV}=347.25$ (pseudo $R^2=38.44$)

CART , 4 variables retenues

$RMSE_{CV}= 315.85$ (pseudo $R^2=49.07\%$)

Random Forest , 13 variables

$RMSE_{CV}=292.53$ (pseudo $R^2=56.31\%$)

Random Forest , 4 variables retenues

$RMSE_{CV}= 269.72$ (pseudo $R^2=62.86\%$)

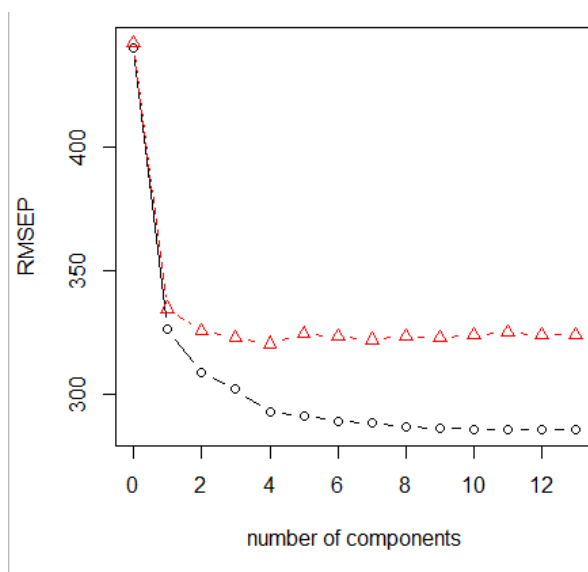
Ex. Comparaison: Régression PLS

Rq. standardisation des variables "X"

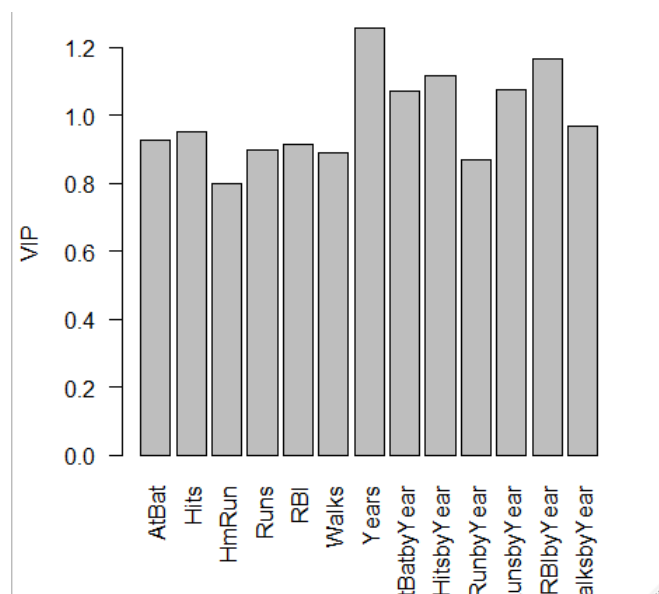
Hitters

y: annual salary

graphique des erreurs en prédiction



critère VIP pour chaque variable



4 composantes PLS retenues

variables pour lesquelles $VIP > 1$

"Years" "CAAtBatbyYear" "CHitsbyYear"
"CRunsbyYear" "CRBIbyYear"

Ex. Bilan comparaison

Hitters
y: annual salary

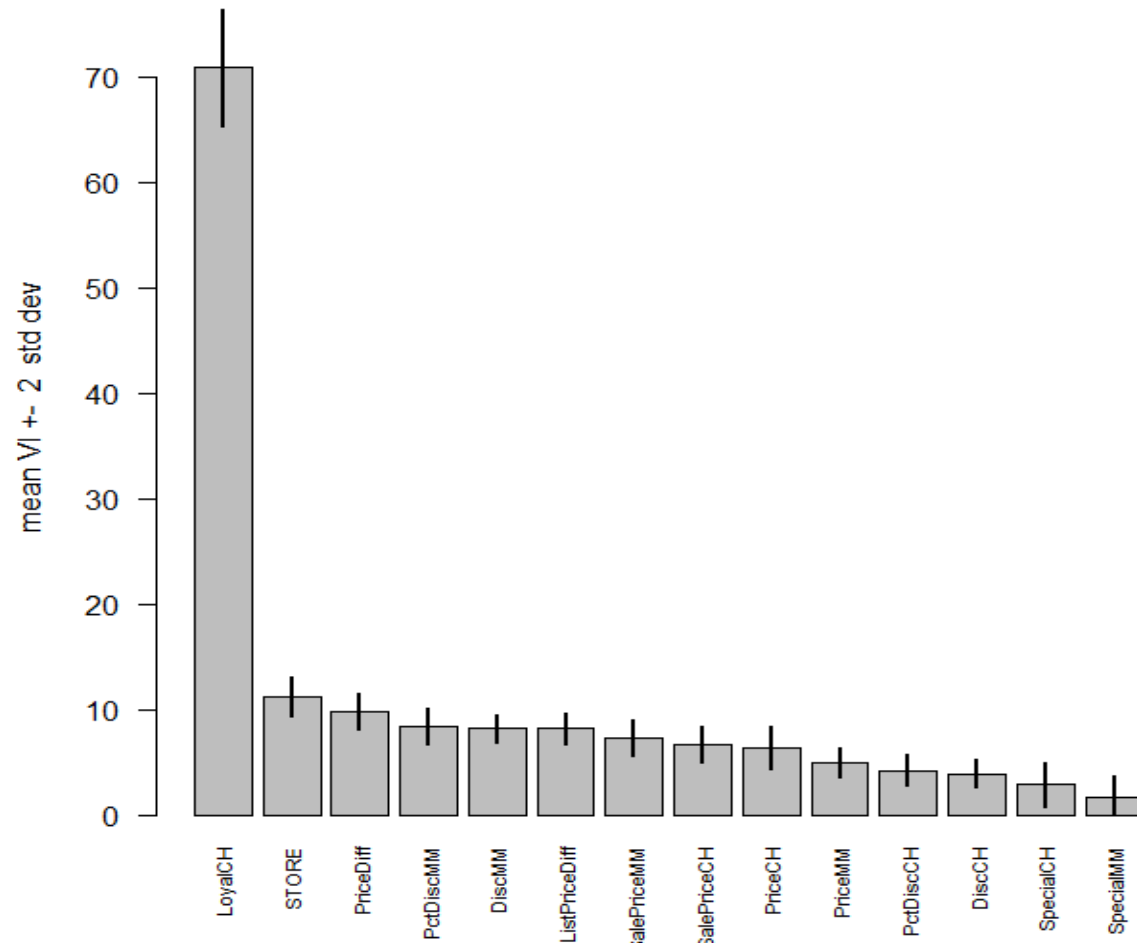
method	# of variables in the model	model complexity	RMSEP _{cv}	Pseudo R ²
CART	13	4 nœuds terminaux	347.25	38.4%
CART	4	5 nœuds terminaux	315.85	49.1%
Rd F	13	_*	293.07**	56.1%
Rd F	4	-	269.72**	62.9%
lm	13	-	323.79	46.5%
Stepwise lm (AIC)	5 à 9	-	313.35	49.9%
PLS Reg	13	4 comp.PLS	320.33	47.6%
PLS-VIP (>1)	6	1 comp.PLS	334.61	42.8%

Ex. Forêt Aléatoire -sélection de variables

OJ
y: #Purchase

Step ①

Moyenne et intervalle de confiance (à ± 2 écart-types) du MDA (Mean Decrease in Accuracy) pour 50 répétitions (50 forêts) ntree=500 mtry=3

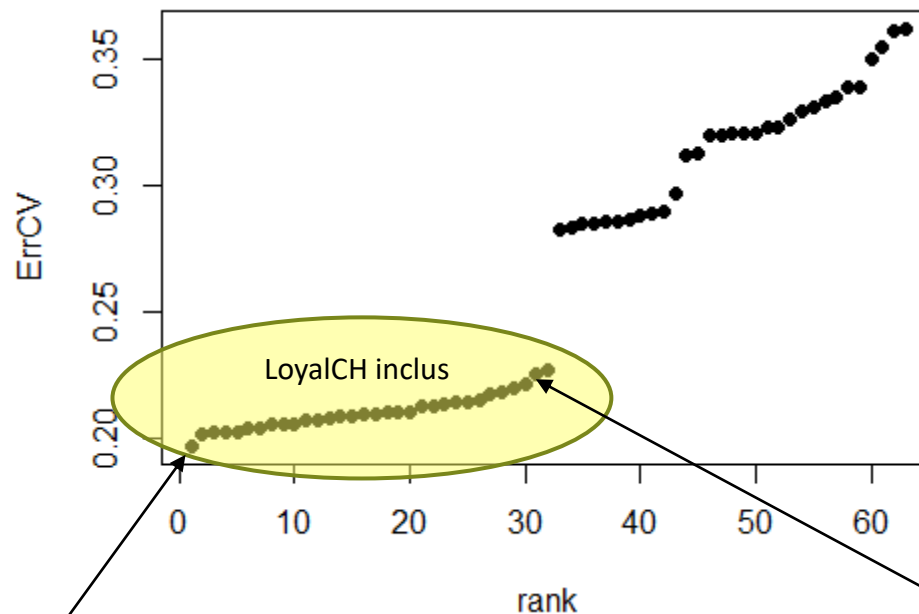


Ex. Forêt Aléatoire - sélection de variables

OJ
y: #Purchase

Step ②

pré-sélection de 6 variables \Rightarrow 63 combinaisons

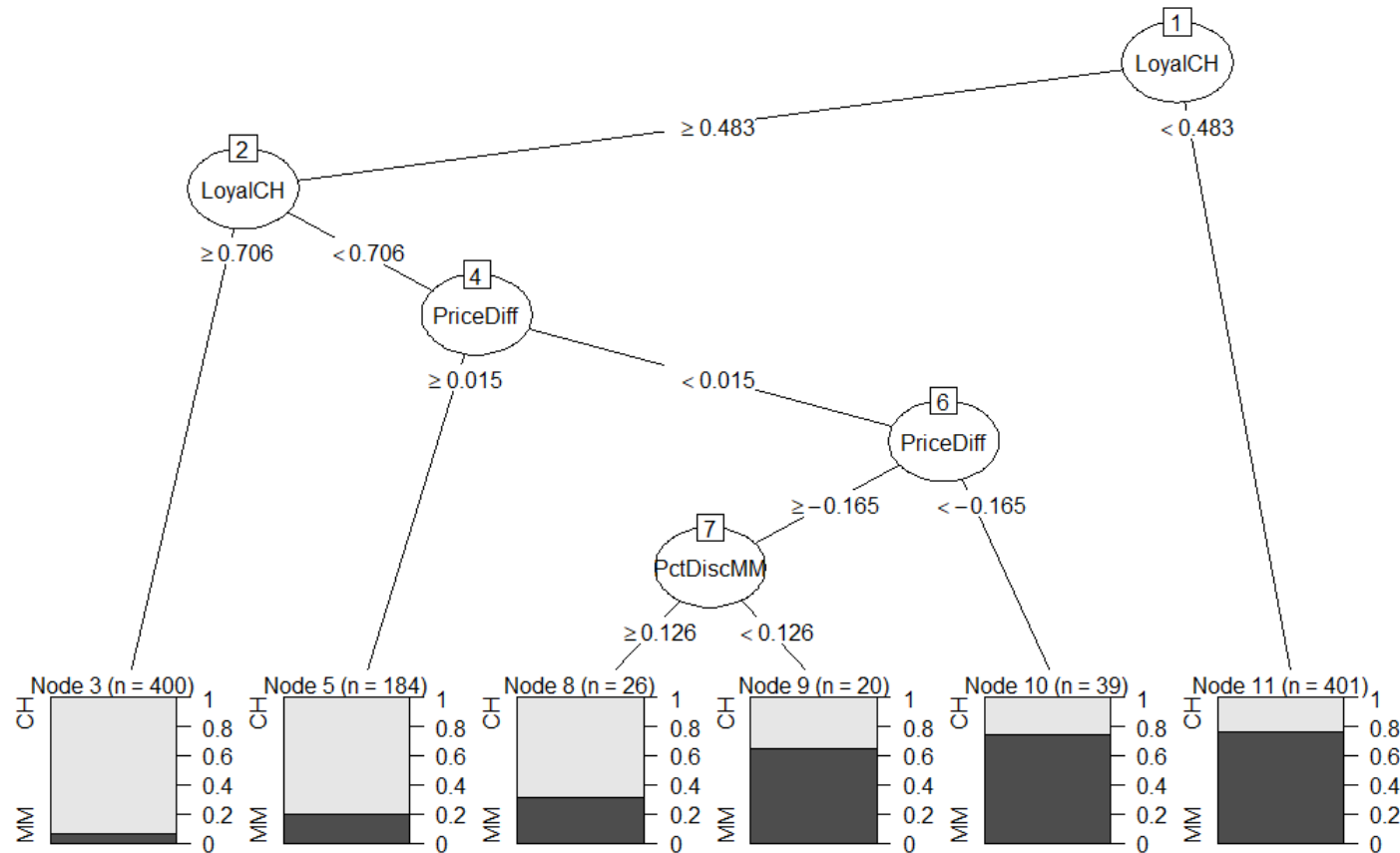


"LoyalCH" "PriceDiff" "PctDiscMM" (ErrrCV=19.63%)

"LoyalCH« seule((ErrrCV=22.52%)

Ex. arbre de régression avec les 3 variables retenues




OJ
y: #Purchase



Arbres CART / Forêts Aléatoires

Caractéristiques principales

- Variable de réponse et prédicteurs de tous types.
- Apte à tenir compte de relations complexes, non-linéaires, ainsi que des interactions.
- Approches construites sur les données elles-mêmes.

	CART	RdF
interprétation	 <ul style="list-style-type: none">• Modèle présenté graphiquement• Une série de règles de décision	
Sélection de variables		
prédiction		 <ul style="list-style-type: none">• Prédiction basées sur des ensembles• Qualité de prédiction souvent assez similaire à celle d'autres approches.

Boosting

Technique ensembliste qui consiste à **agrég**er des modèles (classifieurs faibles) élaborés **séquentiellement** sur un échantillon d'apprentissage pour lequel le **poids des individus sont modifiés au fur et à mesure** .

Les modèles sont ensuite pondérés selon leurs performances et agrégés.

Boosting

original algorithm : **AbaBoost**, Freund & Schapire (1996)

- Développé pour des problèmes de discrimination binaire (-1/+1)
- Séquence de classifieurs “faibles” dont chaque membre est un “expert” vis-à-vis des erreurs de son prédécesseur,
- Les observations du jeu d'apprentissage sont re-pondérées à chaque étape de sorte à donner plus de poids aux observations jusque-là mal attribuées.

Boosting

Pseudo-algorithm de Adaboost.M1

1. Initialisation des poids des observations : $w_i^1 = 1/n$ ($i=1, \dots, n$)

2. Pour $b = 1, \dots, B$, répétitions:

(a) ajustement d'un classifieur : modèle M_b

arbre peu profond

(b) calcul du taux d'erreurs pondérée : err_b

(c) calcul de $\alpha_b = \log((1 - \text{err}_b)/\text{err}_b)$

(d) mise à jour des poids des observations :

$$w_i^b = w_i^{b-1} \exp(\alpha_b I(y_i \neq \hat{y}_i)), \text{ puis normalisation}$$

Lien avec un modèle de régression logistique

3. Modèle final $f(x) = \text{sign}(\sum_{b=1}^B \alpha_b M_b(x))$

Gradient Boosting

Il a été montré (Breiman, 1999) que le processus AdaBoost peut être vu comme un **algorithme de gradient descendant** (avec une fonction de coût exponentielle).

Friedman, Hastie and Tibshirani (2000) développent un cadre statistique, très général, dans lequel les techniques de « boosting » s'interprètent comme des méthodes d'estimation de fonctions. Ils parlent de

"stagewise additive modeling"

Gradient Boosting

Pseudo-algorithm du gradient boosting pour des fonctions de perte avec erreurs quadratiques (L2-boosting)

Efron, Hastie (2016).

Choisir le nombre d'étapes, **B** et un facteur de « shrinkage » τ .

Typiquement, le « base learner » est un arbre CART avec **d** « splits »

1. Initialisations: $\hat{G}_0 \equiv 0$ et vecteur des résidus $\mathbf{r} = \mathbf{y}$
2. Pour $b = 1, \dots, B$, répétitions:
 - (a) ajustement d'un arbre de décision de profondeur d aux données (\mathbf{X}, \mathbf{r}) : \tilde{g}_b
 - (b) mise à jour du modèle : $\hat{G}_b = \hat{G}_{b-1} + \hat{g}_b$ avec $\hat{g}_b = \tau \tilde{g}_b$
 - (c) mise à jour des résidus : $r_i = r_i - \hat{g}_b(x_i)$, $i=1, \dots, n$
3. Séquence de modèles $\hat{G}_b, b = 1, \dots, B$

Gradient Boosting

Taille des arbres

Si $d=1$ (2 nœuds terminaux, « stumps »), on ne peut pas introduire d'éventuelles interactions entre les prédicteurs.

Si $d=2$ (3 nœuds terminaux), le modèle peut prendre en compte des interactions d'ordre deux entre les prédicteurs.

Certains auteurs suggèrent de choisir d entre 3 et 9 .

Shrinkage

Empiriquement, lorsque le paramètre de lissage/ « shrinkage » est faible ($\tau < 0.10$), on constate de meilleures performances prédictives, mais au prix d'une convergence plus lente (nombre d'itérations, B , plus élevé).

Stochastic gradient boosting (Friedman, 1999)

A chaque étape, seule une fraction f de l'échantillon (empiriquement, $0.5 < f < 0.8$) (sélection sans remise) est utilisée pour la construction des « base learners ».

Gradient Boosting

Hitters
y: annual salary

R Package "gbm"

Valdepth=1

valshrink=0.01

valT=1000

```
fit.gbm<- gbm(Salary~., data=DATA,  
              distribution = "gaussian",  
              interaction.depth=valdepth,  
              n.trees = valT,  
              shrinkage = valshrink,  
              bag.fraction = 1,  
              cv.folds=5)
```

Currently available options are "gaussian", "laplace", "tdist", "bernoulli", "huberized", "adaboost", "poisson", "coxph", "quantile", or "pairwise"

Integer specifying the maximum depth of each tree. A value of 1 implies an additive model,... Default is 1.

the fraction of the training set observations randomly selected to propose the next tree in the expansion. This introduces randomness into the model fit. ... Default is 0.5.

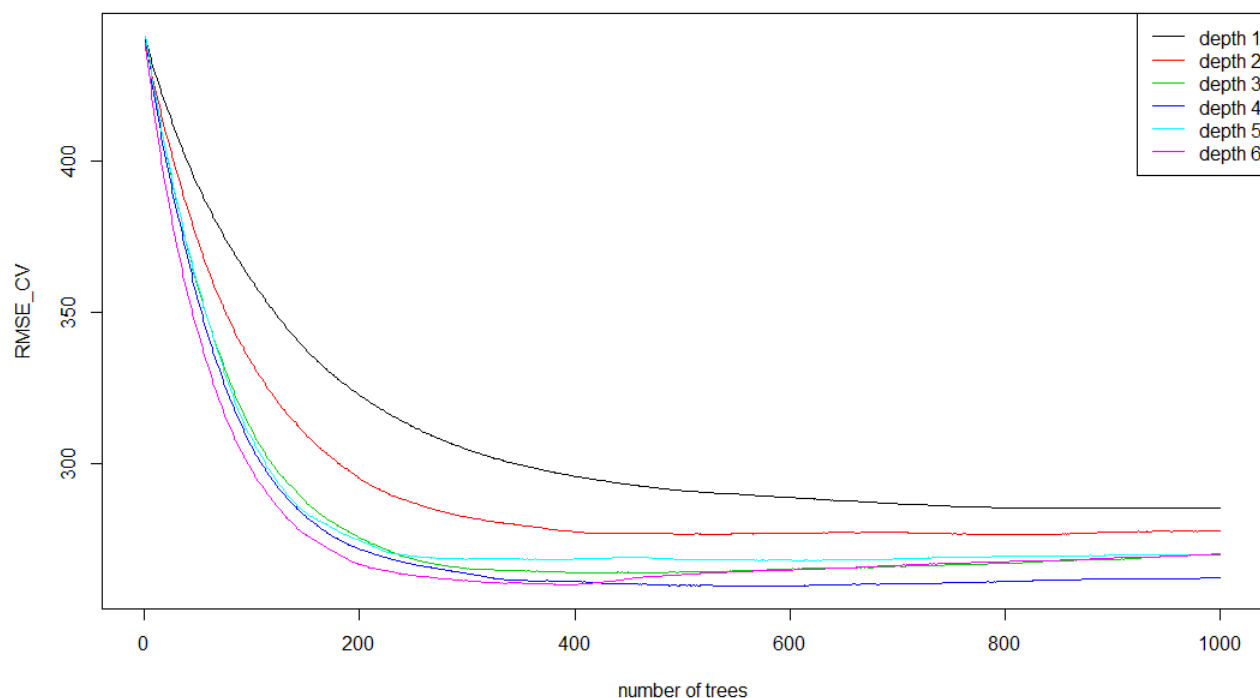
Number of cross-validation folds to perform. If cv.folds>1 then gbm, in addition to the usual fit, will perform a cross-validation, calculate an estimate of generalization error returned in cv.error

Gradient Boosting

Hitters
y: annual salary

RMSE _CV en fonction du nb d'arbres (B) et de leur complexité (d)

$\tau = 0.01$



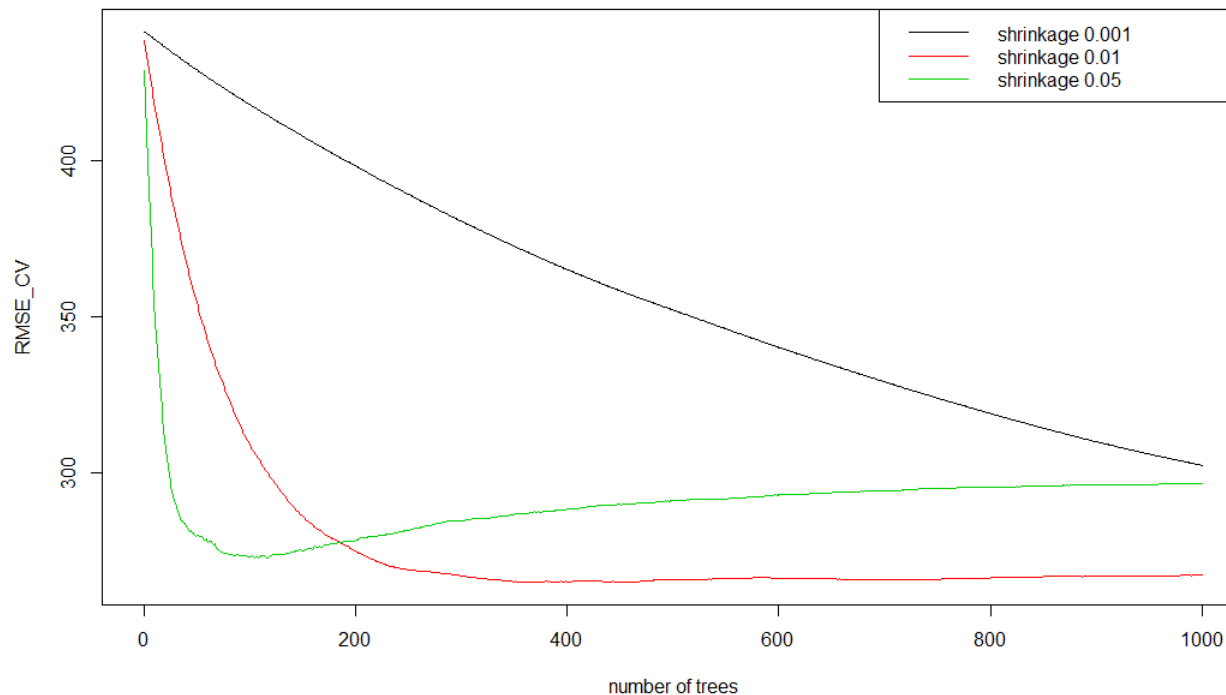
Ici $d=4$ semble être un choix intéressant (avec un risque de sur-ajustement moins important que si $d=6$)

Gradient Boosting

Hitters
y: annual salary

RMSE_CV en fonction du nb d'arbres (B) et du paramètre de shrinkage

$d = 4$



Gradient Boosting

Hitters
y: annual salary

```
gbm(formula = Salary ~ ., distribution = "gaussian", data = DATA,
     n.trees = 1000, interaction.depth = 4, shrinkage = 0.01,
     bag.fraction = 1, cv.folds = 5)
```

A gradient boosted model with gaussian loss function.

1000 iterations were performed.

The best number of iterations (cross-validation) was 302.

There were 13 predictors of which 13 had non-zero influence.

Method	# of variables in the model	model parameters	RMSEP _{cv} *	Pseudo R ²
Rd F	13	mtry=4, nodesize=5 T=300	293.07	56.1%
Rd F	4	mtry= 1, nodesize=5 T=300	269.72	62.9%
L2 gradient boosting	13	d=4, τ =0.01 Bbest=301	275.65	61.2%

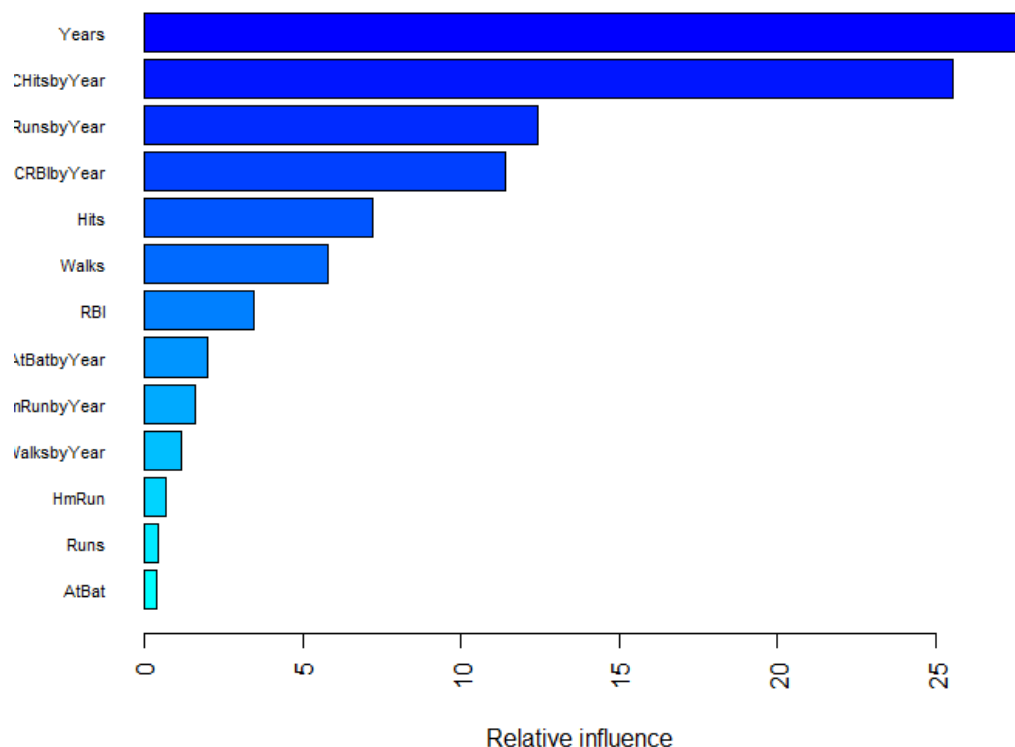
* moy sur plusieurs essais

Gradient Boosting

Hitters
y: annual salary

Influence relative de chaque prédicteur

(moyenne sur l'ensemble des B arbres, de l'amélioration du critère lorsque ce prédicteur est utilisé dans un nœud non terminal d'un arbre i.e. *Decrease in node impurity*)



Gradient Boosting

- Lorsque la variable de réponse Y est une variable de catégories à K modalités, l'algorithme reste globalement le même mais il faut définir une fonction de coût adaptée au classement, et en dériver le gradient.
- On travaille sur les indicatrices (y_k) de chaque classe.
- Si f^k est le modèle additif issu des arbres successifs pour la classe k ($k=1,\dots,K$), la probabilité conditionnelle d'appartenance à la classe k :

$$\pi^k(x_i) = \frac{e^{f^k(x_i)}}{\sum_k e^{f^k(x_i)}}$$

- La règle d'affectation est classique

$$\hat{Y}_i = \arg \max_k \pi^k(x_i)$$

- Pour la classe k , le gradient correspond à l'écart entre son indicatrice et la probabilité d'appartenance à cette classe.
- Remarque : même si on est dans le cadre du classement, le mécanisme interne repose toujours sur un arbre de régression.

Méthodes d'ensembles

L'idée d'assembler des modèles,
sur la base d'approches de type "**bagging**" ou "**boosting**",
s'adresse aux deux composantes de l'erreur de prédiction : le **biais** et la **variance**

Bagging & Random Forests	boosting
<p>Bagging (bootstrap des observations) objectif principal : réduire la variance...mais il faut que les arbres soient individuellement performants (peu de biais).</p> <p>Introduire une perturbation aléatoire en jouant sur la sélection des variables de segmentation à chaque noeud, rend les modèles plus indépendants.</p>	<p>L'apprentissage itératif, à chaque étape, agit essentiellement sur le biais.</p> <p>La combinaison finale des modèles permet de réduire la variance.</p> <p>Ne nécessite pas de grands arbres (weak learners)</p>
<i>Stratégies « stochastiques »</i>	<i>Stratégies "adaptatives"</i>

Un inconvénient soulevé avec ces approches de type machine learning est le manque de lisibilité du méta-modèle ... défaut en partie réduit avec l'évaluation des "Variables Importances"

Quelques références:

- Breiman, L., Friedman, J. H., Olshen R.A. & Stone, C. J. (1984). Classification and Regression Trees. Chapman and Hall, New York
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5-32.
- Freund, Y., Schapire, R. (1996). Experiments with new boosting algorithm. *Machine Learning: proceedings of the 13th International conference*, Morgan Kauffman, San Francisco, p. 148-156.
- Hastie, T., Tibshirani, R., Efron, B., (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Series in Statistics, Springer, New York.
- Friedman, J., Hastie, T., Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting (with discussion). *Ann. Statist.* 28 337–407. MR1790002
- Genuer R., Poggi J.M., Tuleau-Malot C. (2015). VSURF: An R Package for Variable Selection Using Random Forests. *The R Journal*, 7/2.
- Hothorn T., Hornik K., Zeileis A. (2006). Unbiased Recursive Partitioning: A conditional Inference Framework. *Journal of Computational and Graphical Statistics*, 15(3), 651-674.
- Kass G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Appl Stat*, 29, 119–127.
- Loh WY, Chen C, Hordle W, Unwin A (2009). Improving the precision of classification trees. *Ann Appl Stat*, 3, 1710–1737.
- Quinlan J.R. (1993). C4-5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Mateo, California;
- Strobl C., Boulesteix A-L., Zeileis A., Hothorn T. (2007). Biases in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8:25.

Useful R packages

rpart	Recursive Partitioning and Regression Trees
randomForest	Breiman and Cutler's Random Forests for Classification and Regression
VSURF	Variable Selection Using Random Forests
party	A Laboratory for Recursive Partytioning
partykit	A Toolkit for Recursive Partitioning
gbm	Generalized Boosted Regression Models
mboost	Model-Based Boosting