

# Data Science for Actuaries (ACT6100)

Arthur Charpentier

Supervisé # 4 (Arbres)

automne 2020

 <https://github.com/freakonometrics/ACT6100/>



# Procédure cadre

Pour réaliser des prédictions à partir d'un arbre il faut

1. diviser l'espace des variables explicatives en  $J$  régions,  $R_1, \dots, R_J$ , telles que
  - ▶  $R_i \cap R_j = \emptyset$ ,  $i \neq j$  et
  - ▶  $R_1 \cup R_2 \cup \dots \cup R_J = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_J$ .
2. Pour chaque observation qui tombe dans la région  $R_j$ , une prédiction identique sera faite en utilisant, par exemple, la **moyenne** des observations de cette classe:

$$\hat{Y}_{R_j} = \frac{1}{|R_j|} \sum_{i: \mathbf{X}_i \in R_j} Y_i$$

ou la majorité pour un classifieur

# Comment diviser l'espace des variables explicatives?

- ▶ La procédure cadre ne donne aucune indication sur la forme des régions: toutes les possibilités peuvent être tentées
- ▶ On limite la forme des régions à des (hyper-)rectangles (des rectangles dans  $\mathbb{R}^p$ ) afin de simplifier la construction de l'arbre et de permettre une interprétation des résultats (sous la forme d'un graphique).
- ▶ On procède de manière itérative (arbre de décision)
- ▶ Chacun des noeuds se construit en optimisant une fonction objectif.

# Fonction objectif

Pour une problématique de **régression**, on travaillera souvent avec la somme des erreurs au carré (SSE) ou l'erreur quadratique moyenne (MSE)

$$\text{MSE} = \sum_{j=1}^J \sum_{i:\mathbf{X}_i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

où  $\hat{y}_{R_j}$  est la valeur prédite dans la région  $R_j$ . On cherche alors une partition  $R_1, \dots, R_J$  qui **minimise** le SSE.

Pour une problématique de **classification**, notons que

$$\text{MSE}_j = \sum_{i:\mathbf{X}_i \in R_j} (y_i - \hat{y}_{R_j})^2 = n_{0,j}(0 - \hat{y}_{R_j})^2 + n_{1,j}(1 - \hat{y}_{R_j})^2$$

# Fonction objectif

$$\max\{\text{MSE}_j\} = n_{0,j} \left( \frac{n_{0,j}}{n_{0,j} + n_{1,j}} \right)^2 + n_{1,j} \left( \frac{n_{0,j}}{n_{1,j} + n_{1,j}} \right)^2 = \frac{n_{0,j}n_{1,j}}{n_{0,j} + n_{1,j}}$$

de telle sorte que

$$\text{MSE} = \sum_{j=1}^J \frac{n_{0,j}n_{1,j}}{n_{0,j} + n_{1,j}} = \sum_{j=1}^J n_j \cdot p_{1,j}(1 - p_{1,j})$$

On parle d'impureté de Gini

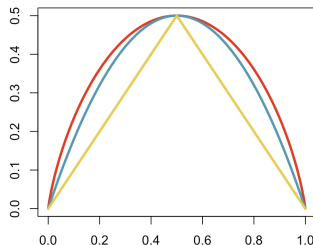
# Fonction objectif

Formellement, l'impureté est une fonction  $\psi(p_{1,j})$  avec  $\psi$  positive, symétrique ( $\psi(p) = \psi(1 - p)$ ), minimale en 0 (et 1).

**Example:** Missclassification  $\psi(p) = 1 - \max\{p, 1 - p\}$

**Example:** Gini,  $\psi(p) = p(1 - p)$

**Example:** (cross) entropy,  $\psi(p) = -p \log p - (1 - p) \log(1 - p)$



# Fonction objectif

- ▶ À nouveau, si on considère toutes les partitions possibles de l'espace des variables explicatives, on devra considérer un trop grand nombre de possibilités.
- ▶ On va plutôt utiliser un algorithme glouton descendant (*top-down greedy approach*) par division binaire récursive (*recursive binary splitting*).
- ▶ Algorithme **descendant**: on commence avec toutes les observations dans une même classe (racine de l'arbre) et on divise l'espace en régions de plus en plus petites.
- ▶ Algorithme **glouton**: l'optimisation se fait à chaque étape sans regard vers le passé ou vers l'avenir.



# Division binaire récursive

**Première étape:** on sélectionne la variable explicative  $X_j$  et le point  $c$  tels que la division en deux régions

$$R_1(j, c) = \{X_1, \dots, X_J | X_j < c\}$$

$$R_2(j, c) = \{X_1, \dots, X_J | X_j \geq c\}$$

conduise à la plus grande réduction possible de la fonction objectif. Si la somme des erreurs au carré a été choisie, on cherche alors à minimiser

$$\sum_{i: \mathbf{X}_i \in R_1(j, c)} \left( Y_i - \hat{Y}_{R_1(j, c)} \right)^2 + \sum_{i: \mathbf{X}_i \in R_2(j, c)} \left( Y_i - \hat{Y}_{R_2(j, c)} \right)^2.$$

(ou n'importe quelle fonction d'impureté)

# Division binaire réursive

**Étapes suivantes:** à l'étape  $k$ , on répète la procédure décrite à la première étape pour chacune des régions créées à l'étape  $k - 1$ , c'est-à-dire que pour chacune des régions  $r$ , on doit identifier la variable explicative  $X_j$  et le point  $c$  qui vont minimiser la fonction objectif après la division donnée par

$$\{\mathbf{X} \in r | X_j < c\} \text{ et } \{\mathbf{X} \in r | X_j \geq c\}.$$

La procédure est arrêtée lorsqu'un certain critère est atteint. En l'absence de critère d'arrêt, l'arbre obtenu aura  $n$  feuilles ( $n$  n'étant pas nécessairement le nombre d'observations dans la base de données)  $\rightarrow$  surapprentissage.

# Construction Hiérarchique d'Arbres de Classification

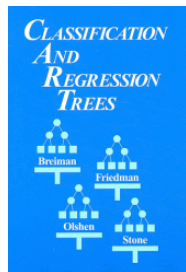
Classification = regrouper les individus en un nombre limité de classes

Ces classes sont construites au fur et à mesure

→ regrouper des individus similaires, séparer des individus ayant des caractéristiques proches.

Histoire : date des années 1960,  
puis Breiman et al. (1984).

outils devenu populaire en apprentissage automatique (machine learning)



# Construction Hiérarchique d'Arbres de Classification

classification descendante :

on sélectionne par les variables explicatives la plus liée

à la variable à expliquer  $Y$ ,

→ donne une première division de l'échantillon

on réitère dans chaque classe

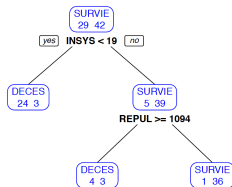
→ chaque classe doit être la plus homogène possible, en  $Y$ .

Différence par rapport à la régression logistique

utilisation séquentielle des variables explicatives

présentation des sorties sous forme d'arbre de décision

i.e. une séquence de noeuds



# Subdivisonner l'espace: une variable explicative

$Y \in \{0, 1\}$  et  $X \in \mathbb{R}$  : on découpe suivant un seuil  $s$ ,

$$\begin{cases} \tilde{X} = L \text{ si } X \leq s \\ \tilde{X} = R \text{ si } X > s \end{cases}$$

	$\tilde{X} = L$ $X \leq s$	$\tilde{X} = R$ $X > s$	
$Y = 0$	$n_{L,0}$	$n_{R,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{L,1}$	$n_{R,1}$	$n_{\cdot,1}$
	$n_{L,\cdot}$	$n_{R,\cdot}$	$n$

**Gini**  $\text{gini}(Y|\tilde{X}) = \sum_{x \in \{L,R\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left(1 - \frac{n_{x,y}}{n_{x,\cdot}}\right)$

# Subdivisonner l'espace: une variable explicative

$Y \in \{0, 1\}$  et  $X \in \mathbb{R}$  : on découpe suivant un seuil  $s$ ,

$$\begin{cases} \tilde{X} = L \text{ si } X \leq s \\ \tilde{X} = R \text{ si } X > s \end{cases}$$

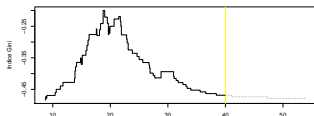
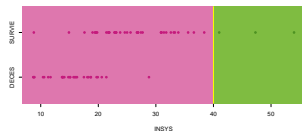
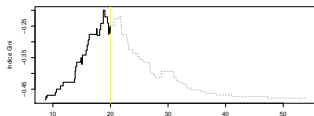
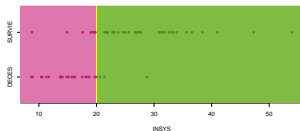
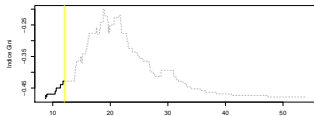
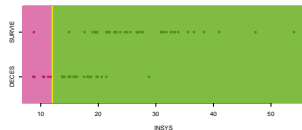
	$\tilde{X} = L$ $X \leq s$	$\tilde{X} = R$ $X > s$	
$Y = 0$	$n_{L,0}$	$n_{R,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{L,1}$	$n_{R,1}$	$n_{\cdot,1}$
	$n_{L,\cdot}$	$n_{R,\cdot}$	$n$

Entropie  $\text{entropie}(Y|X) = - \sum_{x \in \{L,R\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \log \left( \frac{n_{x,y}}{n_{x,\cdot}} \right)$

# Subdivisonner l'espace: une variable explicative

Découpage et indice de Gini

$$\sum_{x \in \{A,B\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left(1 - \frac{n_{x,y}}{n_{x,\cdot}}\right)$$

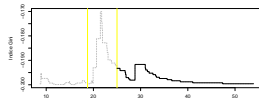
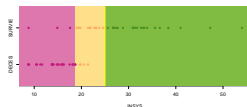


# Subdivisonner l'espace: une variable explicative

On fixe  $s$ , on cherche un second découpage,

$$A = (-\infty, s_2] \quad B = (s_2, s] \quad C = (s, \infty)$$

	$\tilde{X} = A$ $X \leq s_2$	$\tilde{X} = B$ $X \in (s_2, s]$	$\tilde{X} = C$ $X > s$	
$Y = 0$	$n_{A,0}$	$n_{B,0}$	$n_{C,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{A,1}$	$n_{B,1}$	$n_{C,1}$	$n_{\cdot,1}$
	$n_{A,\cdot}$	$n_{B,\cdot}$	$n_{C,\cdot}$	$n$



Découpage et indice de Gini

$$\sum_{x \in \{A,B,C\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left( 1 - \frac{n_{x,y}}{n_{x,\cdot}} \right)$$

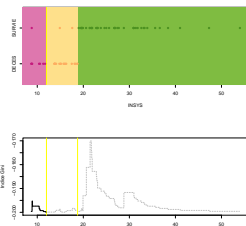


# Subdivisonner l'espace: une variable explicative

On fixe  $s$ , on cherche un second découpage,

$$A = (-\infty, s] \quad B = (s, s_2] \quad C = (s_2, \infty)$$

	$\tilde{X} = A$ $X \leq s$	$\tilde{X} = B$ $X \in (s, s_2]$	$\tilde{X} = C$ $X > s_2$	
$Y = 0$	$n_{A,0}$	$n_{B,0}$	$n_{C,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{A,1}$	$n_{B,1}$	$n_{C,1}$	$n_{\cdot,1}$
	$n_{A,\cdot}$	$n_{B,\cdot}$	$n_{C,\cdot}$	$n$



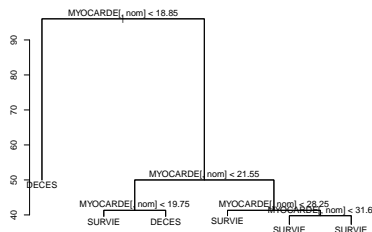
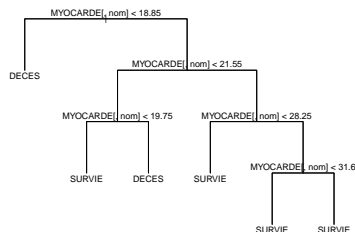
Découpage et indice de Gini

$$\sum_{x \in \{A,B,C\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left( 1 - \frac{n_{x,y}}{n_{x,\cdot}} \right)$$

# Élagage de l'arbre

Étape 1 Construction de l'arbre par un processus récursif de divisions binaires

Étape 2 Élagage de l'arbre (*pruning*), en supprimant les branches trop vides, ou peu représentatives → besoin d'un critère d'élagage (gain en entropie)



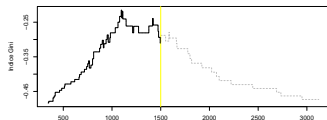
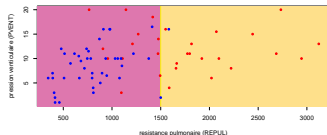
# Cas de plusieurs variables quantitatives

$Y \in \{0, 1\}$  et  $X_1, X_2 \in \mathbb{R}$  : on découpe  $X_1$  suivant un seuil  $s$ ,

$$\begin{cases} \tilde{X} = A \text{ si } X_1 \leq s \\ \tilde{X} = B \text{ si } X_1 > s \end{cases}$$

	$\tilde{X} = A$ $X_1 \leq s$	$\tilde{X} = B$ $X_1 > s$	
$Y = 0$	$n_{A,0}$	$n_{B,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{A,1}$	$n_{B,1}$	$n_{\cdot,1}$
	$n_{A,\cdot}$	$n_{B,\cdot}$	$n$

$$\sum_{x \in \{A,B\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left( 1 - \frac{n_{x,y}}{n_{x,\cdot}} \right)$$



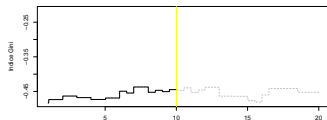
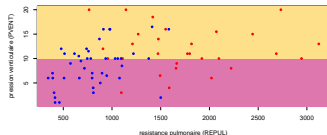
# Cas de plusieurs variables quantitatives

$Y \in \{0, 1\}$  et  $X_1, X_2 \in \mathbb{R}$  : on découpe  $X_2$  suivant un seuil  $s$ ,

$$\begin{cases} \tilde{X} = A \text{ si } X_2 \leq s \\ \tilde{X} = B \text{ si } X_2 > s \end{cases}$$

	$\tilde{X} = A$ $X_2 \leq s$	$\tilde{X} = B$ $X_2 > s$	
$Y = 0$	$n_{A,0}$	$n_{B,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{A,1}$	$n_{B,1}$	$n_{\cdot,1}$
	$n_{A,\cdot}$	$n_{B,\cdot}$	$n$

$$\sum_{x \in \{A,B\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left( 1 - \frac{n_{x,y}}{n_{x,\cdot}} \right)$$



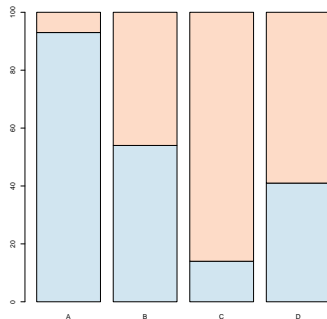
# Cas de variables qualitatives

$X$  prend des valeurs  $\{a, b, c, d\}$ .

Au lieu de faire un arbre par *découpages* successifs, on peut faire un arbre par *regroupement* successifs.

$\{a, b, c, d\}$

$\left\{ \begin{array}{lll} \{(a, b), c, d\} & \{(a, d), b, c\} & \{(a, d), b, c\} \\ \{(b, c), a, d\} & \{(b, d), a, c\} & \{(c, d), a, b\} \\ \{(b, c, a), d\} & \{(b, c, d), a\} & \{(b, c), (a, d)\} \end{array} \right.$



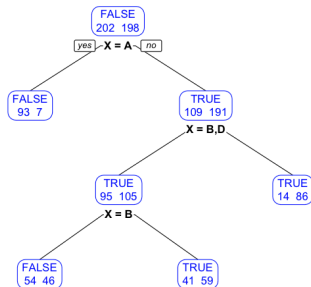
# Cas de variables qualitatives

$X$  prend des valeurs  $\{a, b, c, d\}$ .

Au lieu de faire un arbre par *découpages* successifs, on peut faire un arbre par *regroupement* successifs.

$\{a, b, c, d\}$

$\left\{ \begin{array}{lll} \{(a, b), c, d\} & \{(a, d), b, c\} & \{(a, d), b, c\} \\ \{(b, c), a, d\} & \{(b, d), a, c\} & \{(c, d), a, b\} \\ \{(b, c, a), d\} & \{(b, c, d), a\} & \{(b, c), (a, d)\} \end{array} \right.$



# Cas de variables qualitatives

$X$  prend des valeurs  $\{a, b, c, d\}$ .

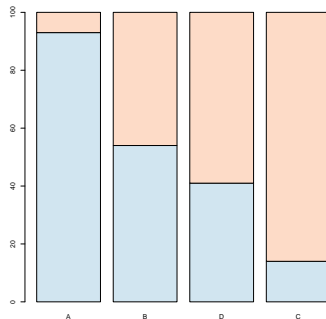
Mais classiquement, on va ordonner les modalités

$$\{a, b, d, c\}$$

puis trouver le découpage optimale

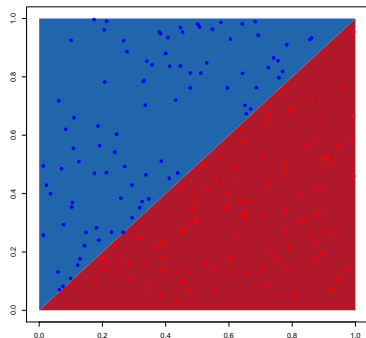
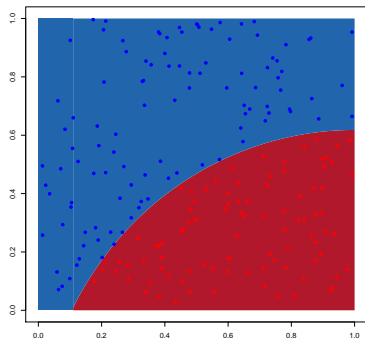
$$\{(a), (b, d, c)\} \quad \{(a, b), (d, c)\} \quad \{(a, b, d), (c)\}$$

$$\{(a), (b, d), (c)\} \quad \{(a), (b), (d, c)\}$$



# Méthode CART et extensions

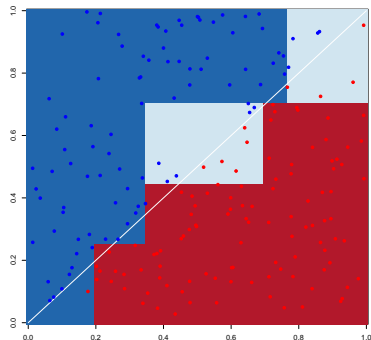
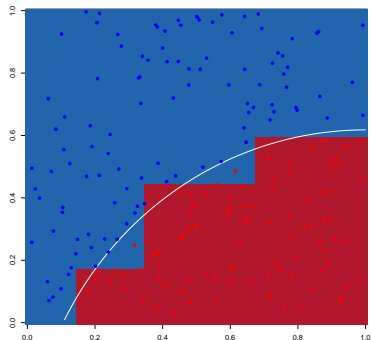
On se contente de faire des coupes suivant  $X_1$  ou  $X_2$ .





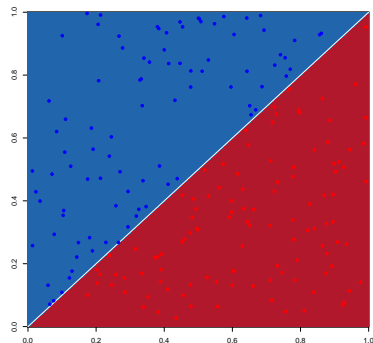
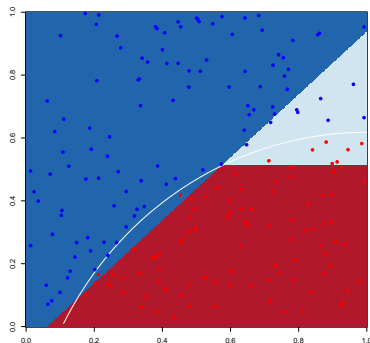
# Méthode CART et extensions

mais ne marche pas bien en présence de non linéarités, ou de rotations



# Méthode CART et extensions

on peut aussi tenter des arbres sur  $X_1 + X_2$



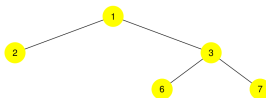
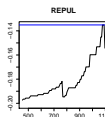
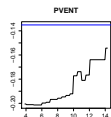
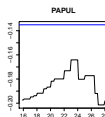
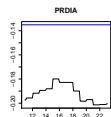
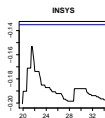
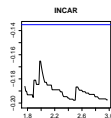
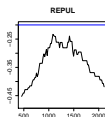
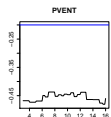
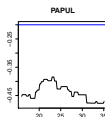
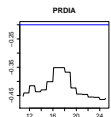
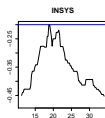
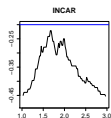
# Mise en oeuvre

$$N_L: \{x_{i,j} \leq s\} \quad N_R: \{x_{i,j} > s\}$$

$$\text{solve } \max_{j \in \{1, \dots, k\}, s} \{\mathcal{I}(N_L, N_R)\}$$

← first split

second split →

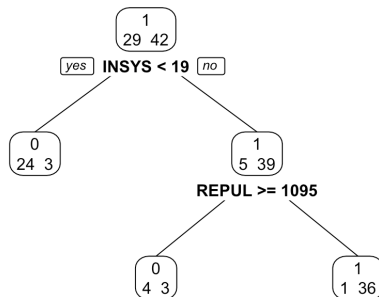
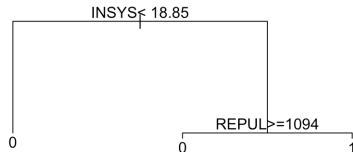


# Arbres avec R

```
1 > myocarde = read.table("http://freakonometrics.free.fr/saporta.csv", header=TRUE, sep=";")
2 > levels(myocarde$PRONO) = c("0", "1")
```

```
1 > library(rpart)
2 > cart = rpart(PRONO~.,
3   data=myocarde)
4 > plot(cart)
5 > text(cart)
```

```
1 > library(rpart.plot)
2 > prp(cart, type=2, extra=1)
```

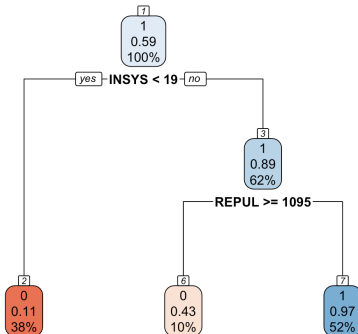
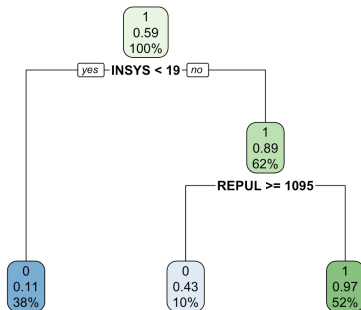


# Arbres avec R

```
1 > print(cart, digits = 2)
2 1) root 71 29 1 (0.408 0.592)
3 2) INSYS < 19 27 3 0 (0.889 0.111) *
4 3) INSYS >= 19 44 5 1 (0.114 0.886)
5 6) REPUL >= 1094.5 7 3 0 (0.571 0.429) *
6 7) REPUL < 1094.5 37 1 1 (0.027 0.973) *
```

```
1 > rpart.plot(cart)
```

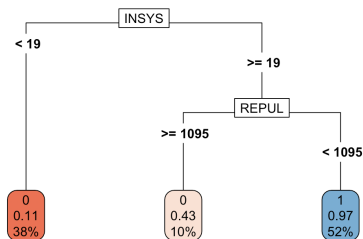
```
1 > rpart.plot(cart, box.
  palette="RdBu", nn=TRUE)
```



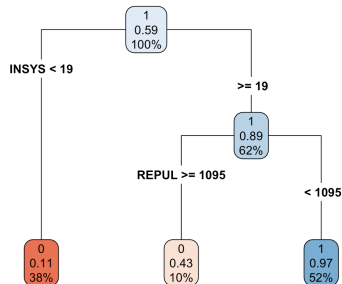
# Arbres avec R

```
1 > rpart.rules(cart, extra = 4, cover = TRUE)
2 PRONO    0    1                                cover
3    0 [.89 .11] when INSYS < 19                    38%
4    0 [.57 .43] when INSYS >= 19 & REPUL >= 1095    10%
5    1 [.03 .97] when INSYS >= 19 & REPUL < 1095    52%
```

```
1 > rpart.plot(cart, box.
  palette="RdBu", type
  =5)
```



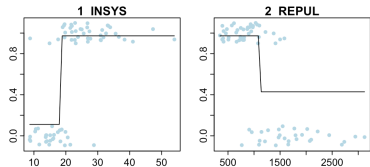
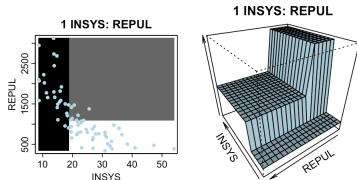
```
1 > rpart.plot(cart, box.
  palette="RdBu", type=4)
```



# Arbres avec R

```
1 > library(plotmo)
2 > plotmo(cart, type = "
  prob", nresponse = "1")
```

```
1 > plotmo(cart, type = "prob",
  nresponse = "1",
  type2 = "image", ngrid2 = 200)
```

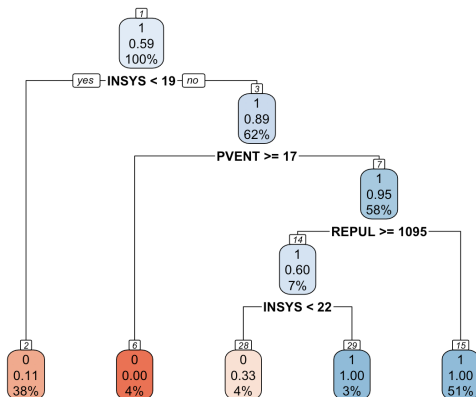
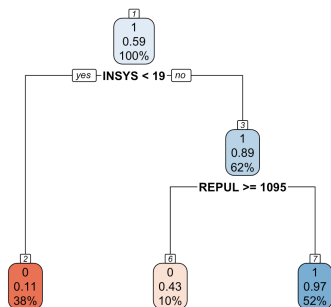


```
1 > path2root = function(node){
2   if(node == 1) node
3   else c(node, path.to.root(node %% 2))}
```

# Arbres avec R

```
1 > cart = rpart(PRONO ~., data=myocarde)
2 > rpart.plot(cart, box.palette="RdBu", nn=TRUE)
```

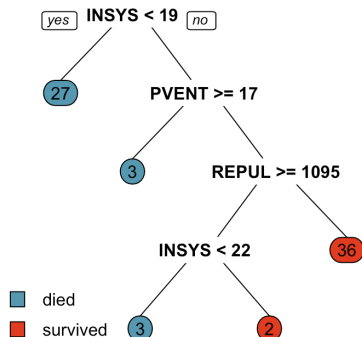
```
1 > cart = rpart(PRONO ~., data=
  myocarde, minsplit = 5,
  cp = 0.01)
2 rpart.plot(cart, box.palette="
  RdBu", nn=TRUE)
```



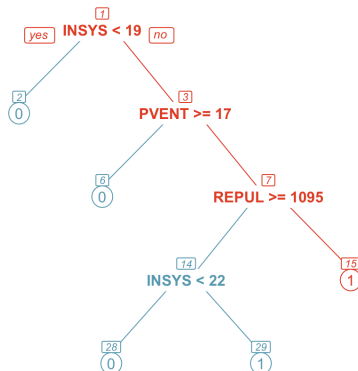


# Arbres avec R

```
1 > boxcols = c("red", "blue")
2 > prp(cart, faclen = 0,
  node.fun = only_count,
  box.col = boxcols)
```



```
1 > node = 15
2 > nodes = as.numeric(row.names(cart$frame))
3 > cols = ifelse(nodes %in%
4 path2root(node), "red", "blue")
5 prp(cart, nn = TRUE, col =
  cols, branch.col = cols,
  split.col = cols)
```



# Critère d'arrêt

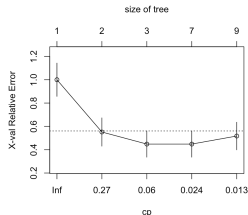
- ▶ On utilise souvent comme critère d'arrêt le fait d'avoir un nombre minimal d'observations dans chacune des régions.
- ▶ Utiliser comme critère d'arrêt le fait d'avoir une diminution de la fonction objectif qui dépasse un certain seuil n'est généralement pas une bonne idée.
- ▶ Même si un critère d'arrêt approprié est sélectionné, il y a fort à parier que l'arbre conduise à du surajustement, c'est-à-dire qu'il performera (trop) bien sur une base de données d'entraînement mais sera mauvais sur une base de données d'évaluation.

# Critère d'arrêt

Formellement, pour savoir si on coupe une feuille  $\{N\}$  en deux branches  $\{N_L, N_R\}$ , on mesure la variation d'impureté

$$\Delta\mathcal{I}(N_L, N_R) = \mathcal{I}(N) - \mathcal{I}(N_L, N_R) = \mathcal{I}(N) - \left( \frac{n_L}{n} \mathcal{I}(N_L) + \frac{n_R}{n} \mathcal{I}(N_R) \right)$$

On coupe si  $\Delta\mathcal{I}(N_L, N_R)/\mathcal{I}(N)$  dépasse cp (complexity parameter, par défaut 1%).



```
1 > cart = rpart(PRONO ~ ., data =  
  myocarde, minsplit=3)  
2 > plotcp(cart)
```

```
1 > prune(cart, cp=0.06)  
2 node), split, n, loss, yval, (yprob) *= terminal node  
3  
4 1) root 71 29 SURVIE (0.40845070 0.59154930)  
5 2) INSYS< 18.85 27 3 DECES (0.888889 0.111111) *  
6 3) INSYS>=18.85 44 5 SURVIE (0.113636 0.886363)  
7 6) PVENT>=17.25 3 0 DECES (1.000000 0.000000) *  
8 7) PVENT< 17.25 41 2 SURVIE (0.048780 0.951219) *
```

# Élagage de l'arbre (pruning)

- ▶ Une meilleure stratégie consiste en (1) construire un très gros arbre de départ  $\mathcal{T}_0$  et (2) couper certaines branches de l'arbre.
- ▶ On cherchera à identifier le sous-arbre  $\mathcal{T} \subset \mathcal{T}_0$  qui minimise

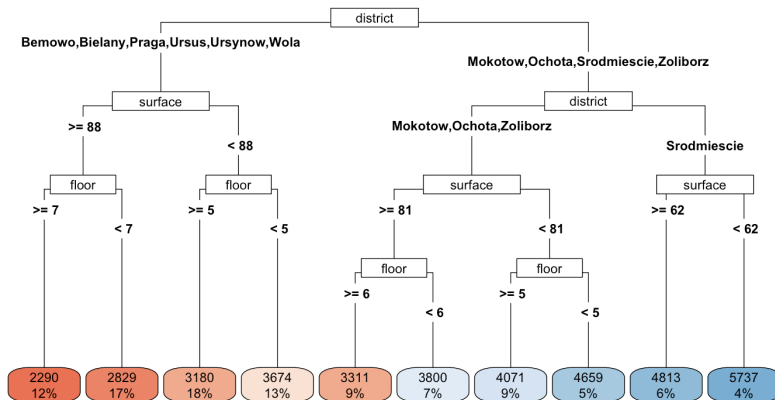
$$\sum_{m=1}^{|\mathcal{T}|} \sum_{i: \mathbf{X}_i \in R_m} \left( Y_i - \hat{Y}_{R_m} \right)^2 + \alpha |\mathcal{T}|,$$

où  $\alpha$  est un (hyper-)paramètre de complexité et  $|\mathcal{T}|$  représente le nombre de feuilles du sous-arbre  $\mathcal{T}$ .

- ▶ Similaire à la régression Ridge/Lasso: compromis entre le biais et la variance, entre la précision et la parcimonie.
- ▶ Le paramètre de complexité peut être estimé par validation croisée.

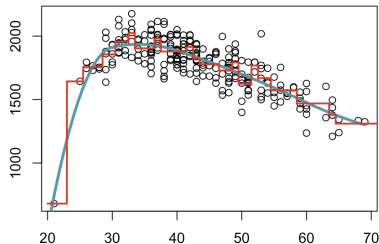
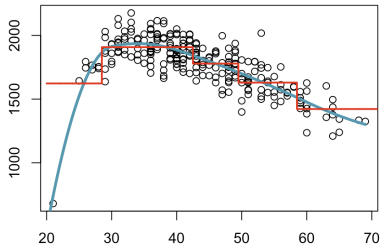
# Arbres de régression, $y \in \mathbb{R}$

```
1 > library(DALEX)
2 > arbre=rpart(m2.price~., data=apartments)
3 > rpart.plot(arbre, box.palette="RdBu", type=5)
```



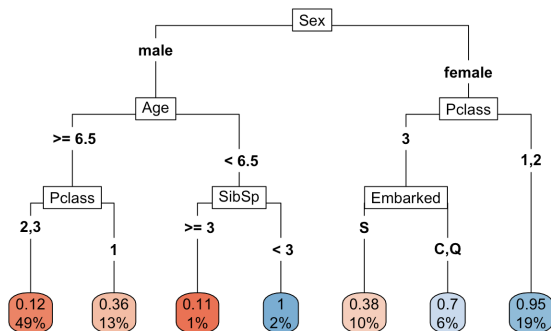
# Arbres de régression, $y \in \mathbb{R}$

```
1 > arbre=rpart(y~x, data=base)
2 > predict(arbre,newdata = data.frame(x=x))
```



# Arbres de classification, $y \in \{0, 1\}$

```
1 > loc_fichier = "http://freakonometrics.free.fr/  
  titanic.RData"  
2 > download.file(loc_fichier, "titanic.RData")  
3 > load("titanic.RData")  
4 > cart = rpart(Survived~., data=base[,1:7])  
5 > rpart.plot(cart, box.palette="RdBu", type=5)
```



# Arbres de classification, $y \in \{A, B, C, D\}$

```
1 > multi.class.model <- rpart(Reliability~., data = cu.  
  summary)  
2 > rpart.plot(multi.class.model)
```

