

# STT3030 - Cours #5

Arthur Charpentier

Automne 2024

# Test et Décision

La vérité est  $y \in \{0, 1\}$  (ou  $\{-, +\}$ ) et on a une prédiction  $\hat{y} \in \{0, 1\}$  (ou  $\{-, +\}$ ), via  $\hat{y} = \mathbf{1}(\hat{p} > s)$ , où  $\hat{p}$  est le score prédict,  $\hat{p} \in [0, 1]$ .

		vérité		vérité	
		-	+	-	+
décision	-	true negative	false negative	-	bonne décision
	+	false positive	true positive	+	type 1 error      bonne décision

On a un **arbitrage** (ou **tradeoff**) entre les types d'erreurs, cf **base rate fallacy**.

Cf théorie des tests, où on teste  $H_0$ , de la forme  $y = 1$  – qui peut être valide (ou pas) – et on doit prendre une décision : rejeter  $H_0$  ou accepter  $H_0$ .

## Test et Décision

$$\text{Prevalence } \frac{200}{10,000} = 2\%$$

$$\text{Specificity } \frac{9,751}{9,800} = 99.5\%$$

$$\text{Sensitivity } \frac{100}{200} = 50\%$$

$$\text{Positive Predictive Value } \frac{100}{149} \sim 67\%$$

$$\text{Specificity } \frac{9,310}{9,800} = 95\%$$

$$\text{Positive Predictive Value } \frac{100}{590} \sim 17\%$$

	- well	+	disease	
-	9,751	100		9,851
+	49	100		149
	9,800	200		10,000

	well	disease		
-	9,310	100	9,410	
+	490	100	590	
	9,800	200		10,000

Cf Wainer & Savage (2008) (until proven guilty: False positives and the war on terror)

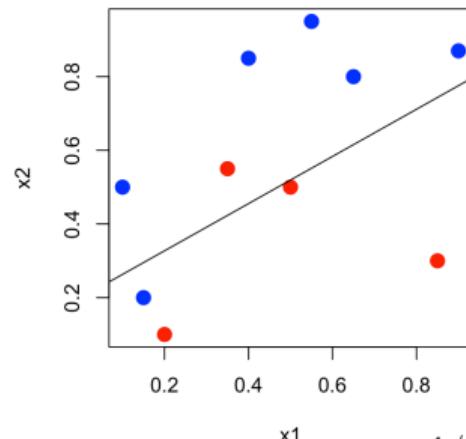
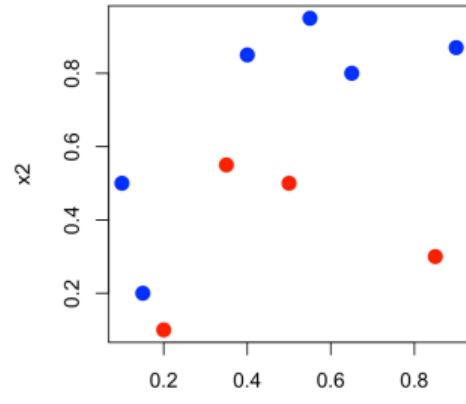
# Courbe ROC

```
1 > x1 = c(.4,.55,.65,.9,.1,.35,.5,.15,.2,.85)
2 > x2 = c(.85,.95,.8,.87,.5,.55,.5,.2,.1,.3)
3 > y = c(1,1,1,1,1,0,0,1,0,0)
4 > df = data.frame(x1 = x1, x2 = x2,
5                   y = as.factor(y))
6 > plot(x1,x2,col=1+y)
7 > reg = glm(y~x1+x2, data=df,
8              family=binomial(link = "logit"))
9 > b = coefficients(reg)
10 > abline(a=-b[1]/b[3],b=-b[2]/b[3])
```

$\mathbb{P}[Y=1|\mathbf{X}=\mathbf{x}] = \mathbb{P}[Y=0|\mathbf{X}=\mathbf{x}]$  si

$$e^{\mathbf{x}^\top \boldsymbol{\beta}} = 1 \text{ ou } \mathbf{x}^\top \boldsymbol{\beta} = 0$$

soit ici  $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$ .



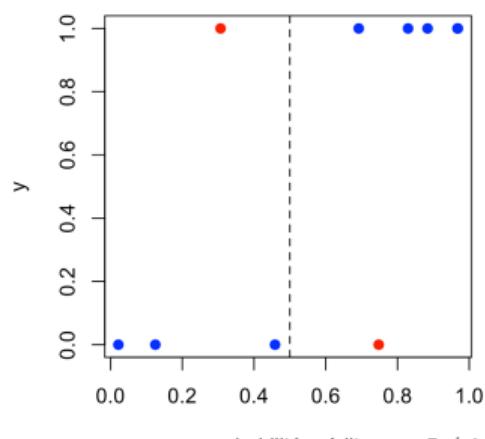
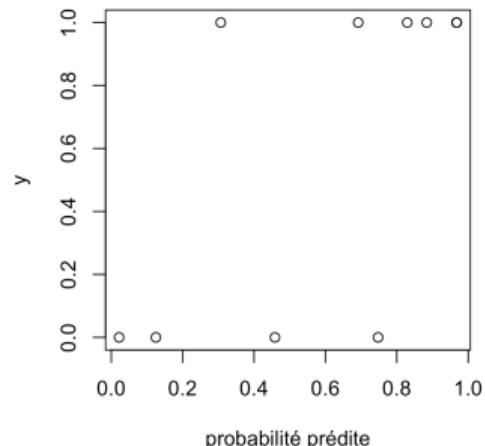
# Courbe ROC

Si on note  $S$  le score prédict (i.e.  $\hat{p}_i$ )

```
1 > Y = df$y
2 > S = predict(reg, type="response")
3 > plot(S,y)
4 > seuil = .5
5 > Yhat = (S>seuil)*1
6 > plot(S,y,col=1+(y==Yhat))
7 > abline(v=seuil,lty=2)
```

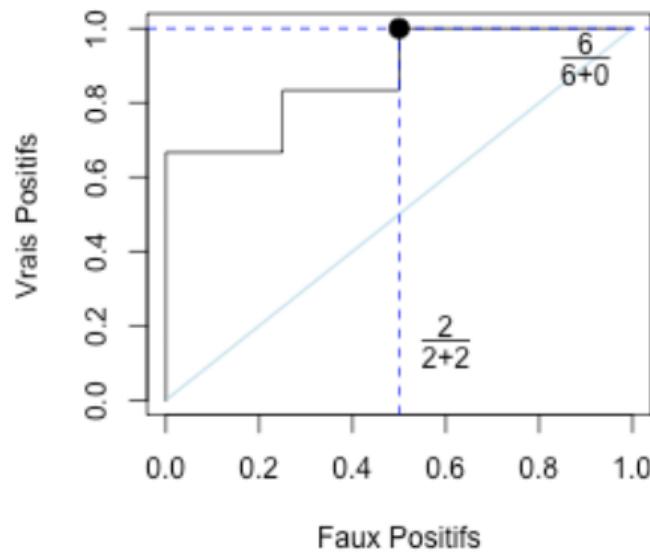
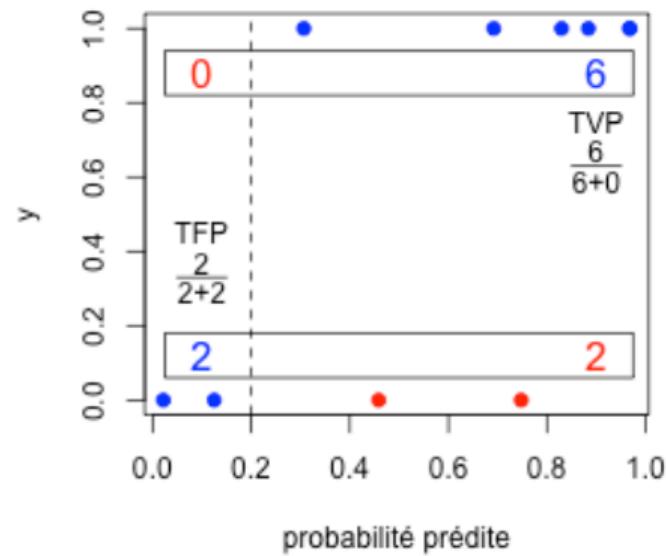
$$\hat{y}_i = \begin{cases} 1 & \text{si } \hat{p}_i > \text{seuil} \\ 0 & \text{si } \hat{p}_i \leq \text{seuil} \end{cases}$$

```
1 > table(Yhat,Y)
2
3   Y
4 Yhat 0 1
5   0 3 1
6   1 1 5
```



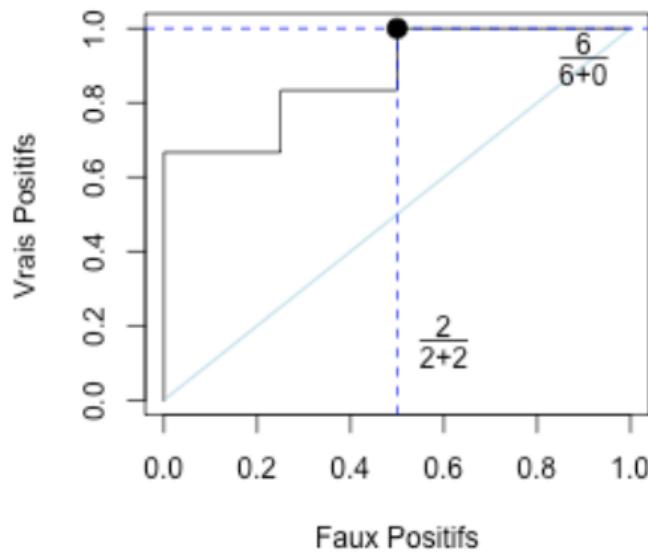
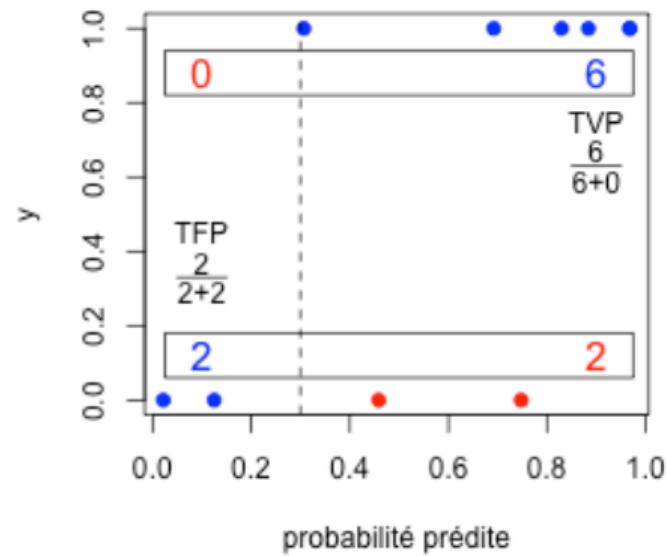
# Courbe ROC

$$FPR = \frac{\mathbb{P}[y = 0, \hat{y} = 1]}{\mathbb{P}[y = 0]} \text{ et } TPR = \frac{\mathbb{P}[y = 1, \hat{y} = 1]}{\mathbb{P}[y = 1]}$$



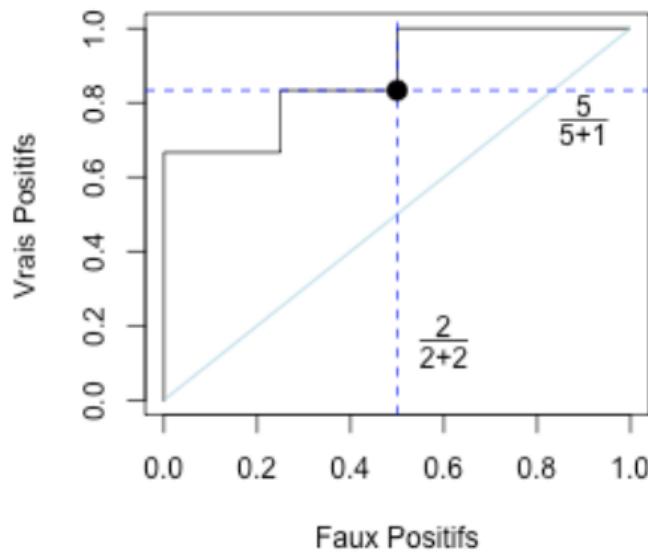
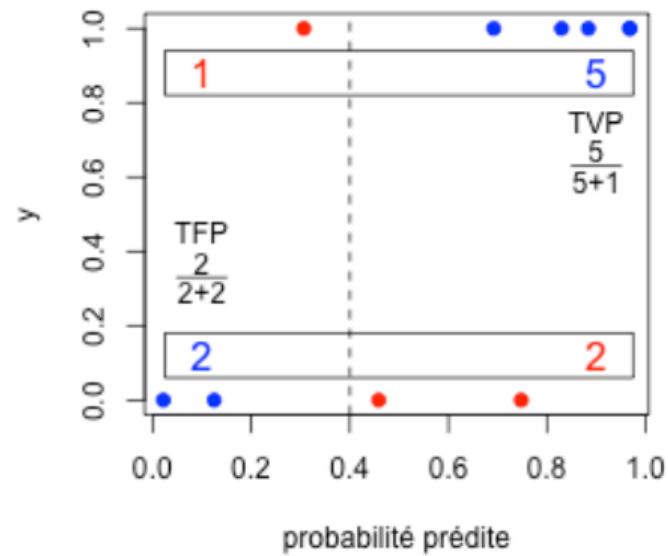
# Courbe ROC

$$FPR = \frac{\mathbb{P}[y = 0, \hat{y} = 1]}{\mathbb{P}[y = 0]} \text{ et } TPR = \frac{\mathbb{P}[y = 1, \hat{y} = 1]}{\mathbb{P}[y = 1]}$$



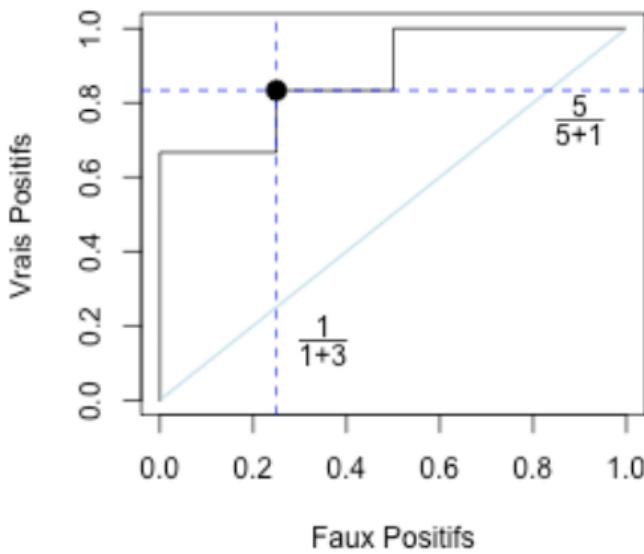
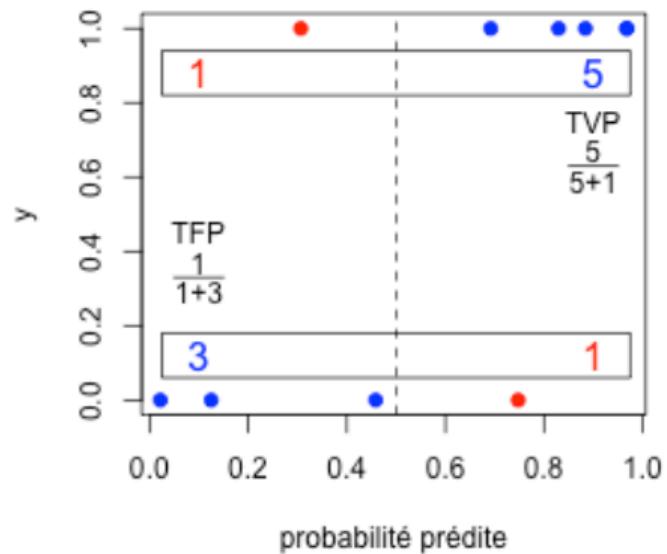
# Courbe ROC

$$FPR = \frac{\mathbb{P}[y = 0, \hat{y} = 1]}{\mathbb{P}[y = 0]} \text{ et } TPR = \frac{\mathbb{P}[y = 1, \hat{y} = 1]}{\mathbb{P}[y = 1]}$$



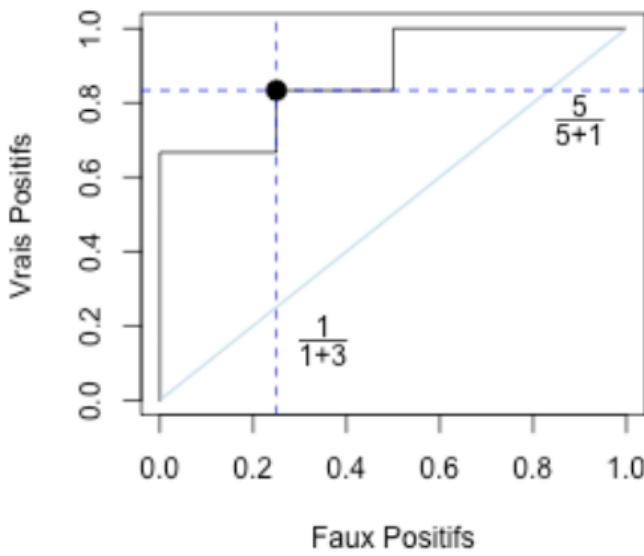
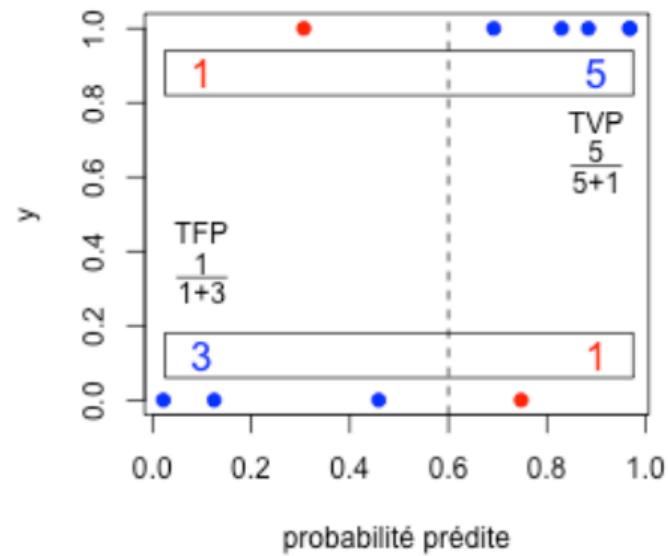
# Courbe ROC

$$FPR = \frac{\mathbb{P}[y = 0, \hat{y} = 1]}{\mathbb{P}[y = 0]} \text{ et } TPR = \frac{\mathbb{P}[y = 1, \hat{y} = 1]}{\mathbb{P}[y = 1]}$$



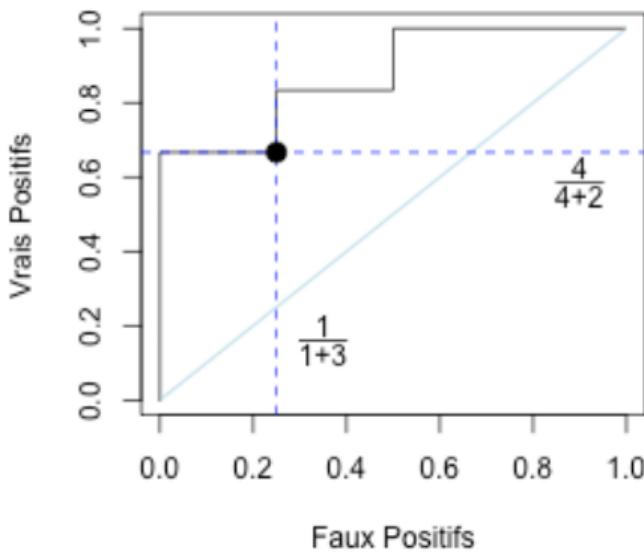
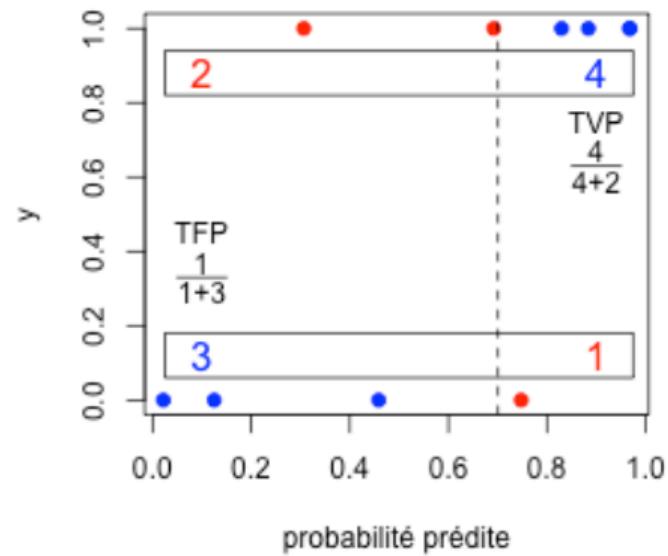
# Courbe ROC

$$FPR = \frac{\mathbb{P}[y = 0, \hat{y} = 1]}{\mathbb{P}[y = 0]} \text{ et } TPR = \frac{\mathbb{P}[y = 1, \hat{y} = 1]}{\mathbb{P}[y = 1]}$$



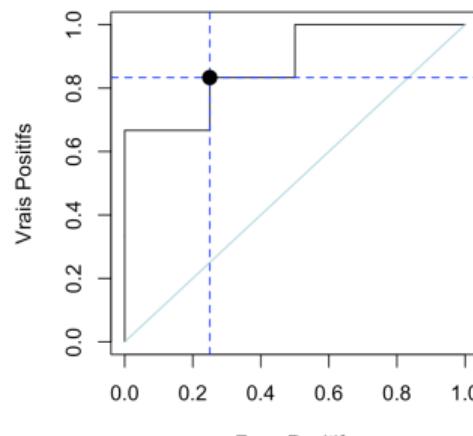
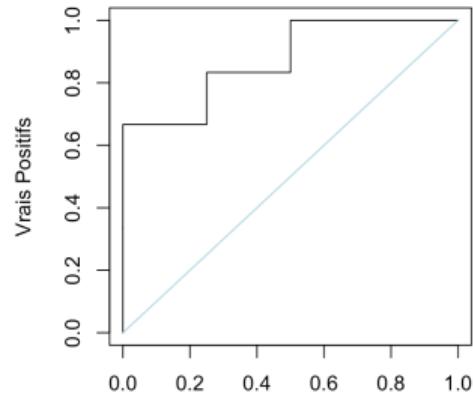
# Courbe ROC

$$FPR = \frac{\mathbb{P}[y = 0, \hat{y} = 1]}{\mathbb{P}[y = 0]} \text{ et } TPR = \frac{\mathbb{P}[y = 1, \hat{y} = 1]}{\mathbb{P}[y = 1]}$$



# Courbe ROC

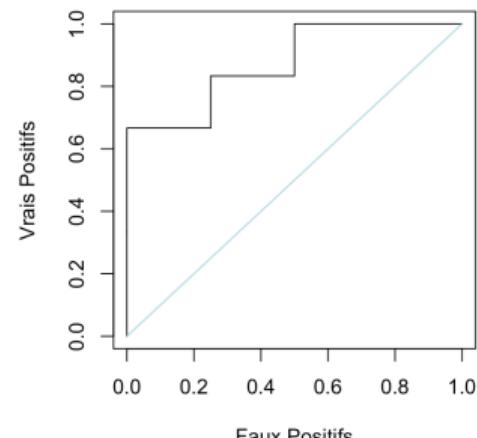
```
1 > roc.curve=function(s){  
2   Ps = (S>s)*1  
3   FP = sum((Ps==1)*(Y==0))/sum(Y==0)  
4   TP = sum((Ps==1)*(Y==1))/sum(Y==1)  
5   vect = c(FP,TP)  
6   names(vect) = c("FPR","TPR")  
7   return(vect) }  
8 > u = seq(0,1,length=251)  
9 > V = Vectorize(roc.curve)(u)  
10 > plot(t(V), type="s")  
11 > table(Yhat,Y)  
12  
13 Yhat 0 1  
14 0 3 1  
15 1 1 5  
16 > sum((Yhat)*(Y==0))/sum(Y==0)  
17 [1] 0.25  
18 > sum((Yhat==1)*(Y==1))/sum(Y==1)  
19 [1] 0.8333333
```



# Courbe ROC

De nombreux packages permettent de tracer des courbes ROC, dont ROCR (ou pROC, plotROC)

```
1 > library(ROCR)
2 > pred = prediction(S,Y)
3 > plot(performance(pred,"tpr","fpr"))
4 > auc.perf = performance(pred, measure = "auc")
5 > auc.perf@y.values[[1]]
6 [1] 0.875
```

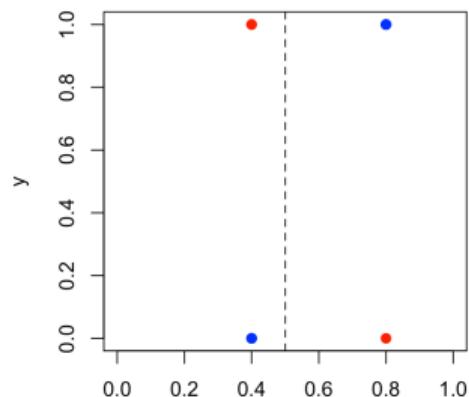
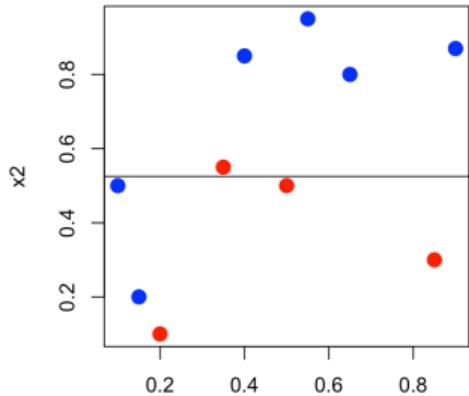


L'**AUC** – aire sous la courbe – donne une idée de la qualité de la classification.

# Courbe ROC

On peut aussi considérer un modèle avec 2 classes,  
Régression de  $y$  sur  $\mathbf{1}_{[.525,\infty)}(x_2)$

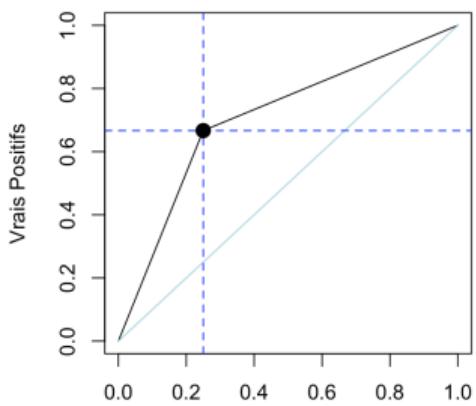
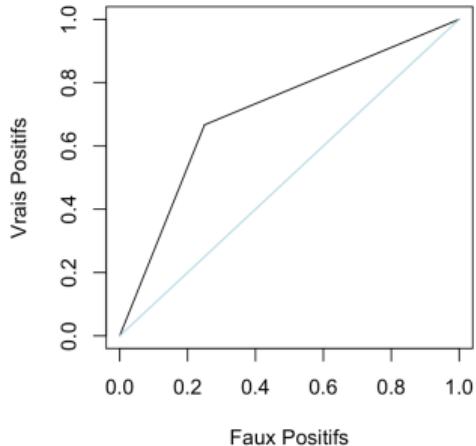
```
1 > reg = glm(y~I(x2>.525), data=df,
2                  family=binomial(link = "logit"))
3 > abline(h=.525)
4 > Y = df$y
5 > S = predict(reg,type="response")
6 > plot(S,y,xlim=0:1)
7 > seuil = .5
8 > Yhat = (S>seuil)*1
9 > table(Yhat,Y)
10
11      Y
12      Yhat 0 1
13        0 3 2
14        1 4 5
```



# Courbe ROC

Avec des classes, la courbe ROC est linéaire par morceaux

```
1 > pred = prediction(S,Y)
2 > plot(performance(pred,"tpr","fpr"))
3 > table(Yhat,Y)
4 
5 Y
6 Yhat 0 1
7   0 3 2
8   1 1 4
9 > auc.perf = performance(pred, measure = "auc")
10 > auc.perf@y.values[[1]]
11 [1] 0.7083333
```

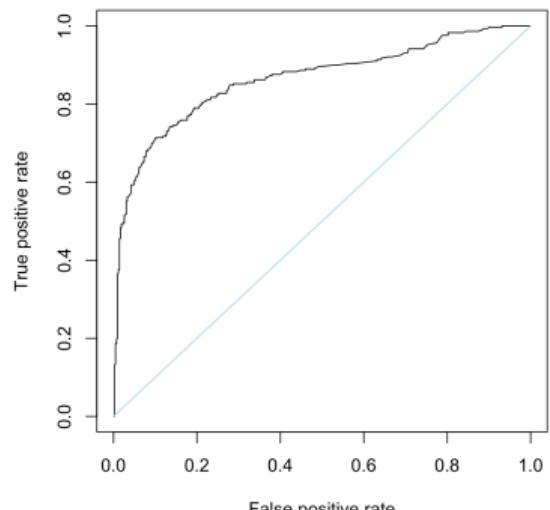


# Survie des Passagers du Titanic

$y$  : indicatrice de survie d'un passager du Titanic

```
1 > loc = "http://freakonometrics.free.fr/titanic.RData"
2 > download.file(loc, "titanic.RData")
3 > load("titanic.RData")
4 > base = base[!is.na(base$Age),1:7]
5 > reg = glm(Survived ~ Sex+poly(Age,3)+Pclass+SibSp,
6           family = "binomial", data = base)
```

```
1 > library(ROCR)
2 > Y = base$Survived
3 > S = predict(reg,type="response")
4 > pred = prediction(S,Y)
5 > plot(performance(pred,"tpr","fpr"))
6 > performance(pred, measure = "auc")
7   @y.values[[1]]
8 [1] 0.8627358
```



# Survie des Passagers du Titanic

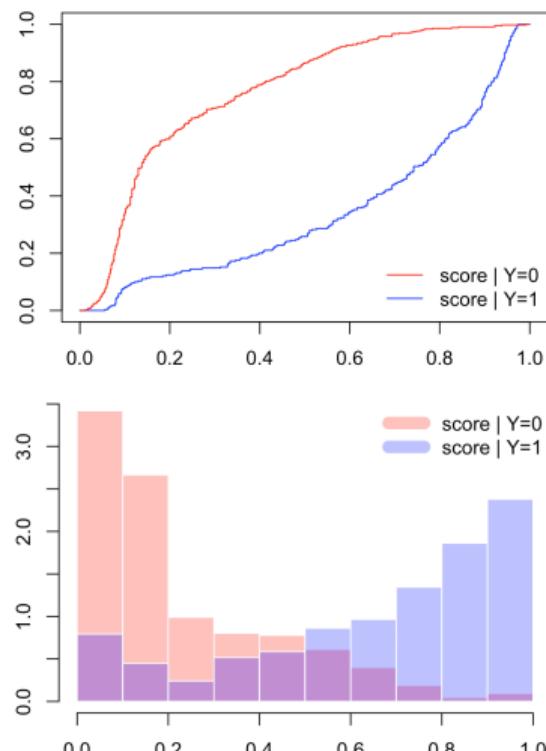
Parmi les autres métriques usuelles, Kolmogorov-Smirnov (**KS**)

Comparer les distributions de  $(S|Y=0)$  et  $(S|Y=1)$

$$d = \sup_{x \in [0,1]} \{|\hat{F}_1(x) - \hat{F}_0(x)|\}$$

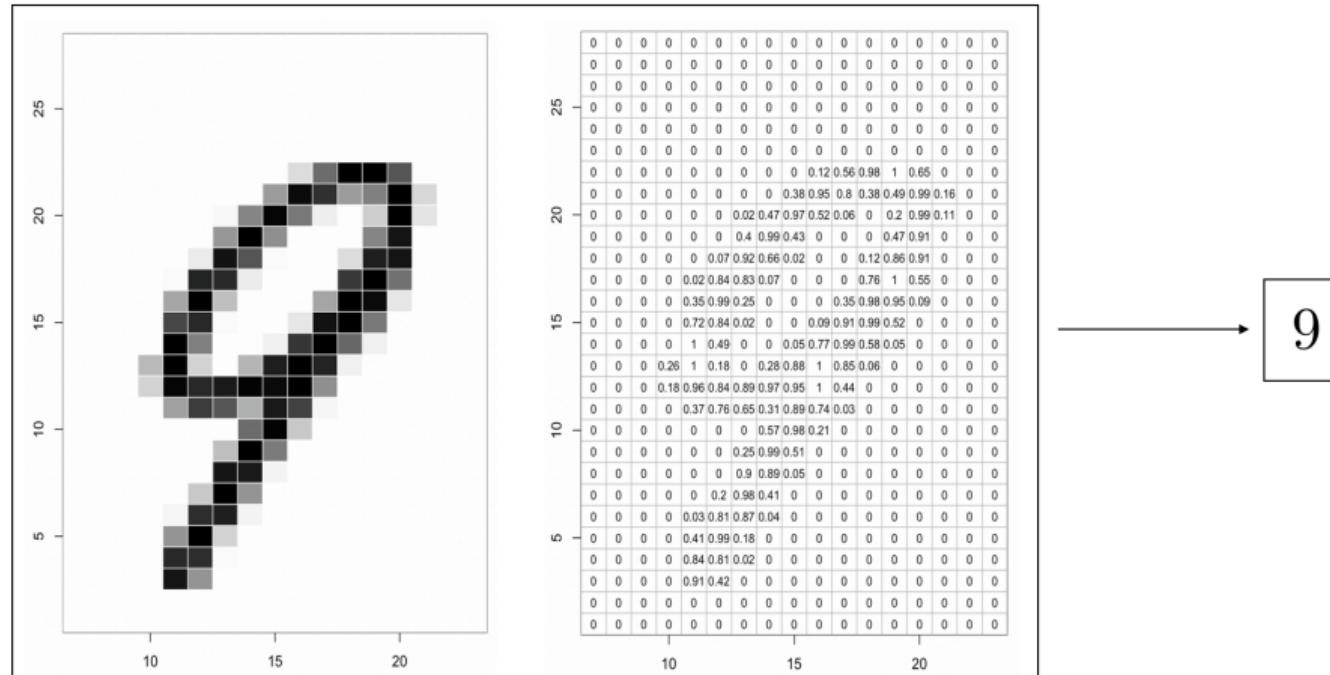
où  $\hat{F}_1(x) = \frac{1}{n_1} \sum_{i:y_i=1} \mathbf{1}(s_i \leq x)$  et  $\hat{F}_0(x) = \frac{1}{n_0} \sum_{i:y_i=0} \mathbf{1}(s_i \leq x)$

```
1 > ks.test(S[Y==0], S[Y==1])
2
3   Asymptotic two-sample Kolmogorov-Smirnov test
4
5 data: S[Y == 0] and S[Y == 1]
6 D = 0.61238, p-value < 2.2e-16
7 alternative hypothesis: two-sided
```



## Classification... sur des images

$(y_i, \mathbf{x}_i)$ , où  $y \in \{0, 1, 2, 3, \dots, 9\}$  et  $\mathbf{x}_i \in \mathcal{M}_{28,28} = [0, 1]^{28 \times 28}$

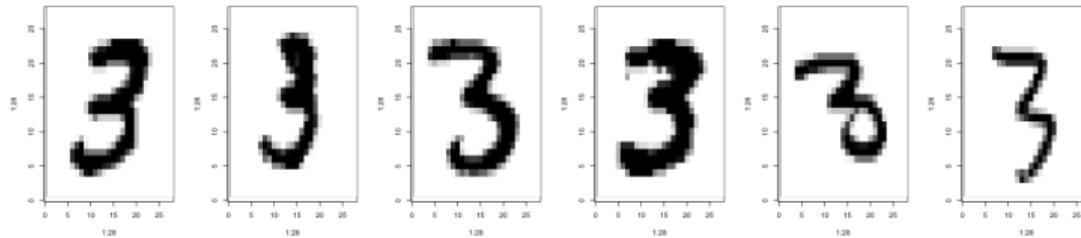


$$\boldsymbol{x}_i \in \mathcal{M}_{28,28}$$

$$y_i \in \{0, 1, \dots, 9\}$$

# Classification... sur des images

Here  $\{(y_i, \mathbf{x}_i)\}$  with  $y_i = \text{"3"}$  and  $\mathbf{x}_i \in [0, 1]^{28 \times 28}$



```
1 > library(keras)
2 > mnist = dataset_mnist()
3 > n = 1000
4 > V = mnist$train$x[1:n,,]
5 > MV = NULL
6 for(i in 1:n) MV=cbind(MV,as.vector(V[i,,]))
7 > MV = t(MV)
8 > df = data.frame(y=mnist$train$y[1:n],x=MV)
```

# Classification... sur des images

Peut-on reconnaître les '1' ?

```
1 > reg=glm((y==1)~.,data=df,family=binomial)
2 Warning messages:
3 1: glm.fit: algorithm did not converge
4 2: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Problème numérique !

On a  $k = 784$  variables explicatives... il faut réduire la dimension !

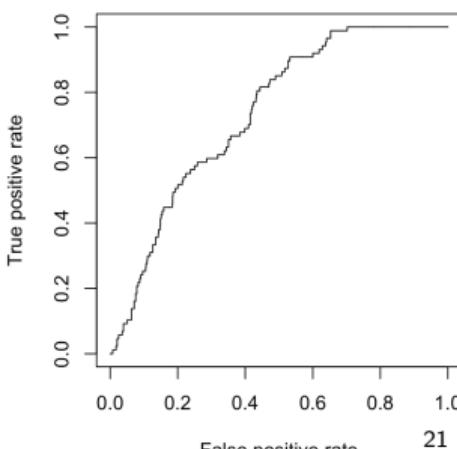
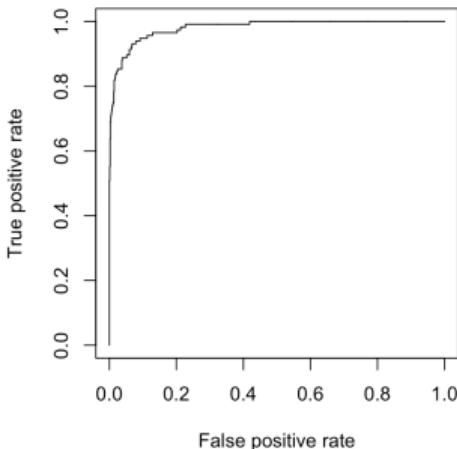
```
1 > library(factoextra)
2 > pca=prcomp(MV)
3 > res.ind = get_pca_ind(pca)
4 > PTS = res.ind$coord
5 > k=3
6 > dfpca = data.frame(y=mnist$train$y[1:n],x=PTS[,1:k])
7 > reg1 = glm((y==1)~.,data=dfpca,family=binomial)
8 > reg8 = glm((y==8)~.,data=dfpca,family=binomial)
```

# Classification... sur des images

On essaye de reconnaître les '1' et les '8'

```
1 > library(ROCR)
2 > Y1 = as.numeric((df$y == 1)*1)
3 > S1 = predict(reg1,type="response")
4 > pred1 = prediction(S1,Y1)
5 > plot(performance(pred1,"tpr","fpr"))
6 > Y8 = as.numeric((df$y == 8)*1)
7 > S8 = predict(reg8,type="response")
8 > pred8 = prediction(S8,Y8)
9 > plot(performance(pred8,"tpr","fpr"))
```

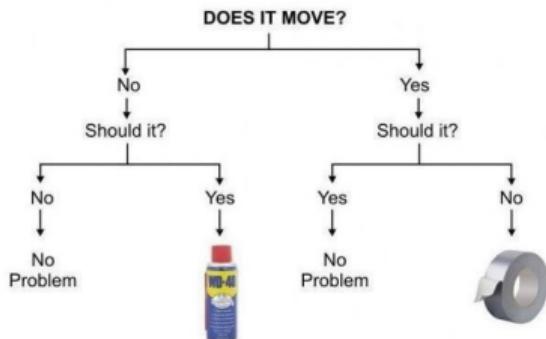
On utilise ici seulement  
les 3 premières composantes principales



## Terminologie

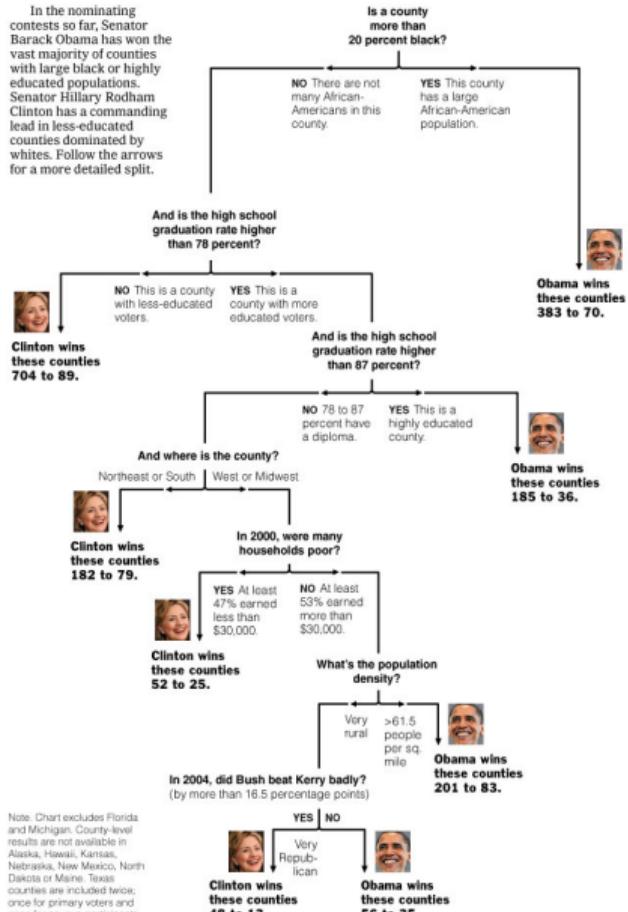
- ▶ Les régions finales obtenues sont nommées  **noeuds terminaux** ou, plus souvent, **feuilles** de l'arbre.
  - ▶ Les autres points où se font les divisions sont des  **noeuds internes** de l'arbre.
  - ▶ Les segments de l'arbre qui connectent les noeuds sont des  **branches**.

via New York Times (2008)



## Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.



Note: Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice; once for primary voters and once for general election.

# Prédicteurs

$\mathbf{X} = \{X_1, \dots, X_p\}$  forment une collection de prédicteurs.

On dit que  $\mathbf{X} \in \mathcal{X}_1 \times \dots \times \mathcal{X}_p = \mathcal{X}$ , où  $\mathcal{X}_j$  est le domaine de  $X_j$ .

Ce sont les variables que nous utilisons pour prédire.

$Y \in \mathcal{Y}$  est une réponse.

On veut prédire la réponse  $Y$  avec les prédicteurs  $X$ .

- ▶ On suppose l'existence d'une fonction  $f: \mathcal{X} \rightarrow \mathcal{Y}$
- ▶ Souvent de la forme la vrai fonction  $Y = f(\mathbf{x}) + \varepsilon$

Où  $\varepsilon$  est une supposé variabilité inexpliquée.

En apprentissage supervisé, on veut estimer  $f$ .

- ▶ Pour faire de la prédiction:  $\hat{Y} = \hat{f}(\mathbf{x})$
- ▶ Étant donnés des prédicteurs  $X$  quelle est notre meilleur estimation de la réponse.

## Données et modèle

Nous avons un ensemble de données  $D = \{(\mathbf{x}_i, y_i) | i \in (1, \dots, n)\}$  ou  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$  ou données **d'entraînement** (*training set*).

C'est grâce à ces données que nous allons entraîner notre modèle à “imiter”  $f$ .

C'est notre **échantillon**.

- ▶ Chapitre 8 du manuel de référence, James et al. (2013)
- ▶ Section 9.2 du deuxième manuel de référence, Friedman et al. (2001)
- ▶ Nous allons discuter de la version la plus simple des arbres de décision.
- ▶ Depuis, de multiples extensions et modifications furent proposées.

## Forme d'un arbre de décision

- ▶ Encore une fois, nous cherchons à estimer  $f$  une fonction de  $\mathcal{X}$  vers  $\mathcal{Y}$ .
- ▶ On prédit une réponse pour  $\mathbf{x} \in \mathcal{X}$  en fonction de la région  $R \subset \mathcal{X}$  dans laquel  $\mathbf{x}$  appartient.
- ▶ La réponse prédite pour une région est constant, disons  $c_r$  pour la région  $R_r$ .
- ▶ On doit donc subdiviser (partitionner)  $\mathcal{X}$  en régions disjointes,

$$\forall i, j, \quad i \neq j \text{ (i.e. } R_i \neq R_j) \Rightarrow R_i \cap R_j = \emptyset \text{) et } \bigcup_{i=1}^m R_i = \mathcal{X}.$$

## Forme d'un arbre de décision

Peut prendre la forme:

$$\hat{Y}_i = \begin{cases} c_1 & \text{si } \mathbf{x}_i \in R_1 \\ c_2 & \text{si } \mathbf{x}_i \in R_2 \\ \dots \\ c_m & \text{si } \mathbf{x}_i \in R_m \end{cases}$$

ou pour un point quelconque  $\mathbf{x}$ ,

$$\hat{Y} = \begin{cases} c_1 & \text{si } \mathbf{x} \in R_1 \\ c_2 & \text{si } \mathbf{x} \in R_2 \\ \dots \\ c_m & \text{si } \mathbf{x} \in R_m \end{cases}$$

ou peut être présenté comme un modèle additif:

$$\hat{Y} = \sum_{j=1}^m c_j \mathbf{1}_{\mathbf{x} \in R_j}$$

## Régions d'une arbre de décision

- ▶ C'est quoi une région ?
- ▶ Les régions  $R_r$  sont définies par un ensemble de conditions que l'on représente souvent par une intersection de plusieurs fonctions indicatrices.
- ▶ Supposons que  $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3 \times \mathcal{X}_4 \{0, 1\}^4$ .  
Par exemple, une région peut se définir comme:  $R_r = \mathbf{1}_{x_1=1} \cap \mathbf{1}_{x_2=0} \cap \mathbf{1}_{x_4=0}$
- ▶ Supposons que  $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 (0, 1)^2$   
Par exemple, une région peut se définir comme:  $R_r = \mathbf{1}_{x_1 \leq 0.2} \cap \mathbf{1}_{x_2 > 0.4}$
- ▶ Pour apprendre un arbre de décision il faut donc apprendre comment former ces régions.

# Régions d'une arbre de décision

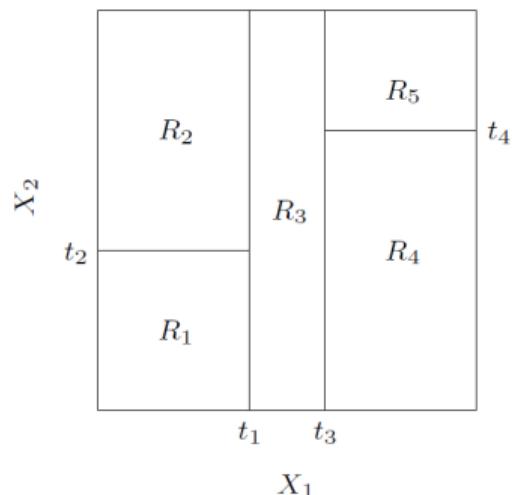


Figure 1: Partitionnement de  $X$  en régions. Extrait de [Friedman et al. \(2001\)](#).

## Mais où est l'arbre là dedans ?

- Le partitionnement de la diapositive précédente peut être associé à l'arbre suivant.

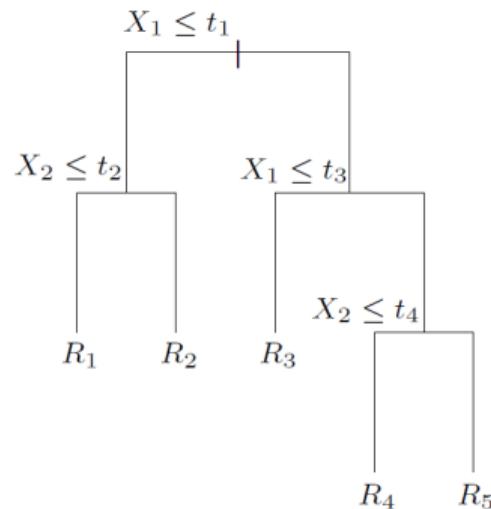


Figure 2: Arbre attaché au partitionnement précédent. Extrait de Friedman et al. (2001)

Remarque: deux arbres différents peuvent mener au même partitionnement

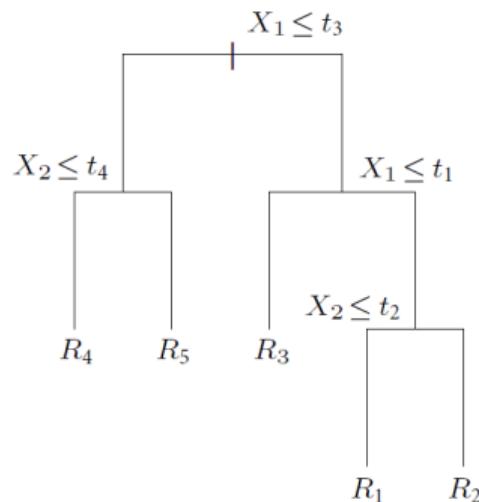


Figure 3: Arbre attaché au partitionnement précédent. Extrait de Friedman et al. (2001)

## Exemple chiffré du livre.

- ▶ Base de données *Hitters*
- ▶ On veut prédire le salaire du joueur (Salary) en fonction des années d'expériences du joueur (Years) et son nombre de frappes (Hits).

## Exemple chiffré du livre.

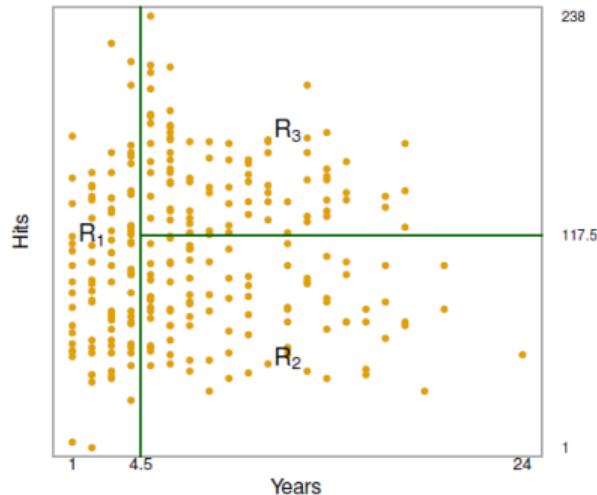


Figure 4: Partitionnement des données Hitters. Extrait de James et al. (2013).

## Exemple chiffré du livre.

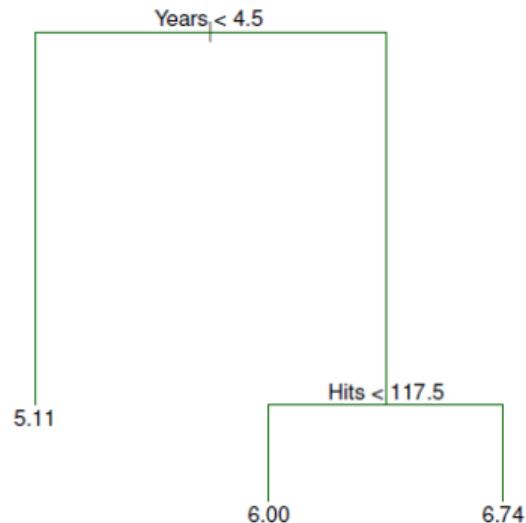


Figure 5: Arbre rattaché au partionnement des données Hitters. Extrait de James et al. (2013).

## Exemple chiffré du livre.

- ▶ On prédit que le salaire des joueurs ayant moins de 4.5 années d'expérience ( $R_1 = \{X|X_Y < 4.5\}$ ) est 5.11M.
- ▶ On prédit que le salaire des joueurs ayant plus de 4.5 années d'expérience et moins de 117.5 frappe ( $R_2 = \{X|X_Y > 4.5 \cap X_H < 117.5\}$ ) est 6.00M.
- ▶ On prédit que le salaire des joueurs ayant plus de 4.5 années d'expérience et plus de 117.5 frappe ( $R_3 = \{X|X_Y > 4.5 \cap X_H > 117.5\}$ ) est 6.74M.

## Mais où est l'arbre la dedans ?

- ▶ Les arbres apportent une visualisation.
- ▶ Les arbres sont faciles à interpréter.
- ▶ Les arbres sont intimement liés à la procédure d'apprentissage.
- ▶ *On apprend l'arbre, pas les régions.*

## Comment choisir le bon arbre

- ▶ Le meilleur partitionnement parmis tous les partitionnements ne peut être choisi. (Il en existe trop, dessin au tableau).
- ▶ L'estimation d'un arbre (et de son partitionnement joint) se fait par partitionnement binaire récursif.
- ▶ Simplement dit, on crée les branches de l'arbre une par une.
- ▶ On choisissant le meilleur partitionnement **à l'instant**.
- ▶ Pensez à meilleur sous-ensemble VS sélection par avant.

## Comment choisir le bon arbre

Pour former un arbre  $\hat{f}$  à l'aide d'un échantillon  $S$ , il nous faut **quatre** ingrédients.

1. Une collection de partitionnement binaire.
2. Une évaluation de la qualité du partitionnement.
3. Une règle d'arrêt.
4. Une règle étiquettage.

## Former une collection de partitionnement binaire

*Établir toutes les manières possible de diviser les données.*

Partitionnement binaire: on divise nos données en **deux groupes**.

Pour chaque prédicteurs  $X_j$  on regarde chacune des manière de diviser les données  $s_i \in S_{ent}$ .

- ▶ Si  $X_i$  est continue, les partitionnement sont de la forme  $X_i \leq t$  contre  $X_i > t$  et nous allons tester toute les division possible dans l'échantillon, on regarde donc les  $n - 1$  possible partitionnement.
- ▶ Si  $X_i$  est catégoriel avec  $c$  classes, nous allons regarder toute les  $2^{c-1} - 1$  possible partition de groupes.
- ▶ Exemples au tableau.

# Régions d'une arbre de décision

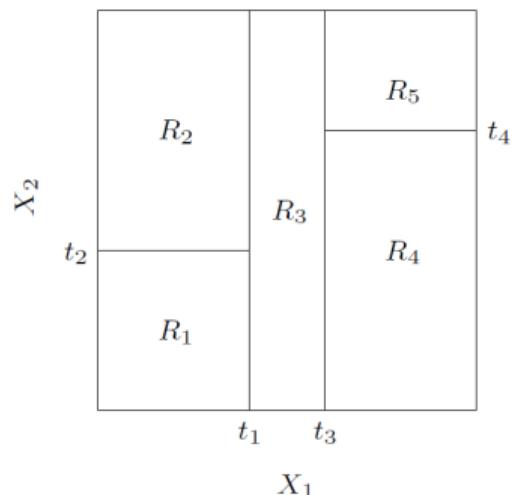


Figure 6: Partitionnement de  $X$  en régions. Extrait de [Friedman et al. \(2001\)](#).

## Régions impossibles

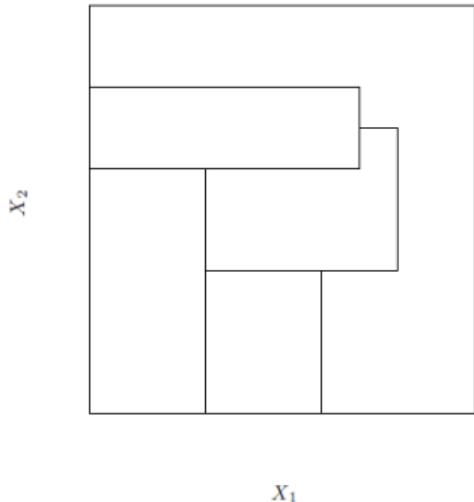


Figure 7: Un type de partitionnement qu'on ne peut pas faire avec les arbres de décision.  
Extrait de Friedman et al. (2001).

# Évaluer la qualité du partitionnement

## *Déterminer le meilleur partitionnement*

Intuitivement, on veut que les régions contiennent des sujets qui se ressemblent.

On veut faire de la prédiction ou de la classification, on essaie donc de regrouper des réponses similaires à l'aide des prédicteurs. On veut maximiser l'homogénéité.

Si les prédicteurs et la réponse sont liés, on peut utiliser les prédicteurs pour former des groupes de sujets dont la réponse se ressemble.

## Évaluer la qualité du partitionnement

Soit  $Q_j$ , une mesure de mixitude de la région  $j$  (impurity measure)

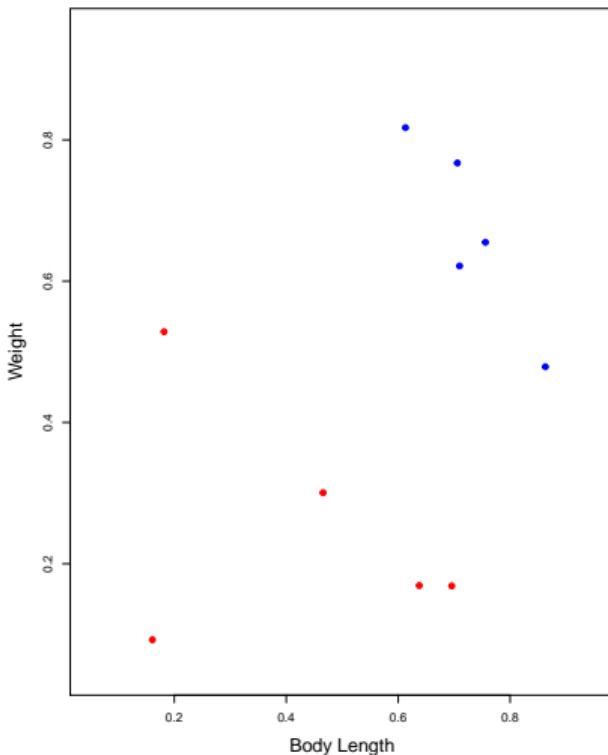
- ▶ Si  $y$  est continue, on évalue l'erreur quadratique:  $Q_j = \sum_{i \in R_j} (y_i - \frac{1}{n} \sum_{i \in R_j} y_i)^2$ .
- ▶ Si  $y$  est catégoriel on utilise d'autre mesure de mixitude tel que l'indice GINI:  
$$Q_j = \sum_{k=1}^c \hat{p}_{jc}(1 - \hat{p}_{jc})$$
 où  $\hat{p}_{jc}$  est la proportion de la classe  $c$  dans la région  $j$   
$$(\hat{p}_{jc} = \frac{1}{n_j} \sum_{i \in R_j} \mathbf{1}_{y_i=c})$$
.
- ▶ Exemples au tableau.

## Évaluer la qualité du partitionnement

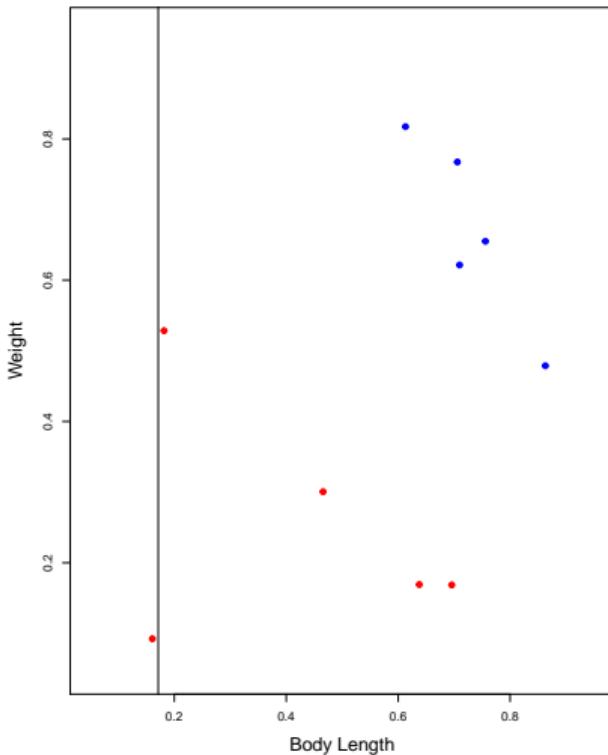
Finalement, pour chaque partitionnement total, on évalue l'hétérogénéité moyenne des 2 régions résultantes. Si l'on cherche à diviser  $R_p$  en deux nouvelle région  $R_t$  et  $R_r$  on regarde:

$$n_r Q_r + n_t Q_t \quad (1)$$

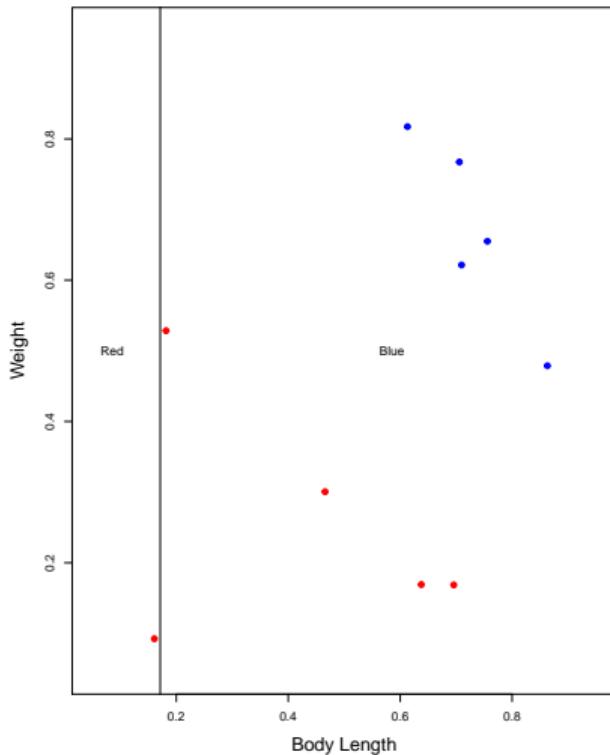
## Exemple: prédition continue et réponse binaire



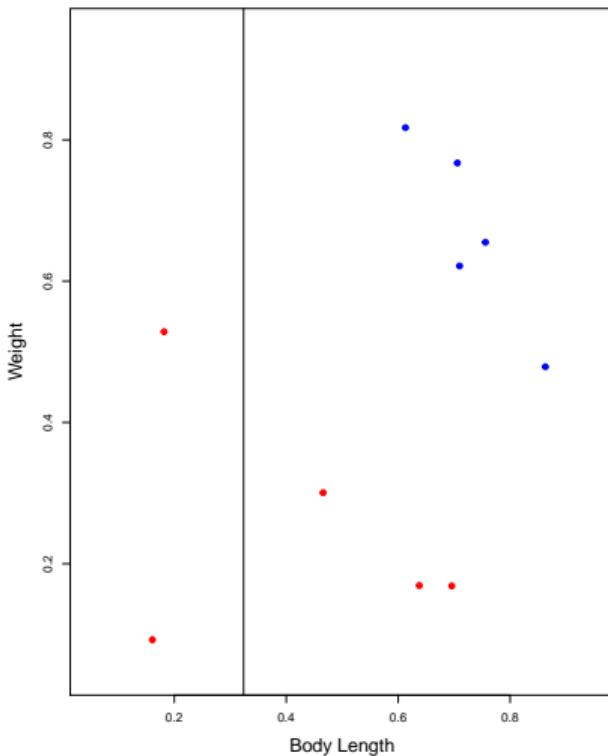
## Exemple: prédition continue et réponse binaire



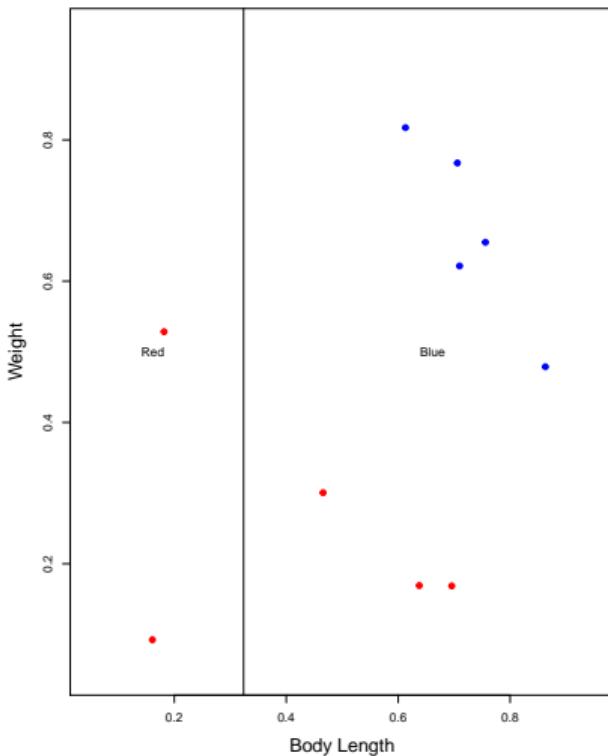
## Exemple: prédition continue et réponse binaire



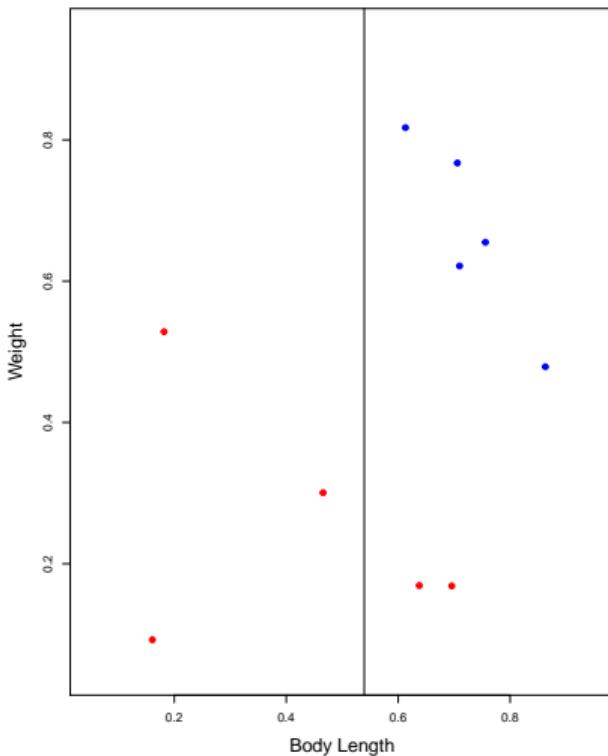
## Exemple: prédition continue et réponse binaire



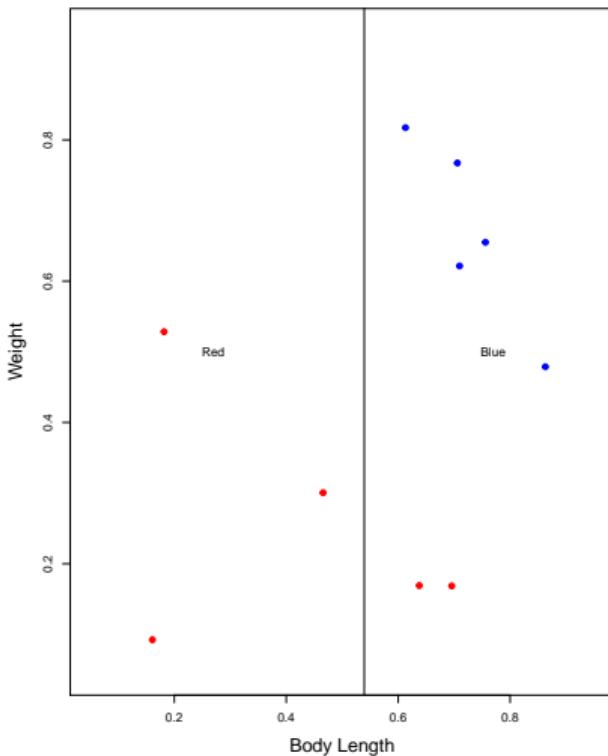
## Exemple: prédition continue et réponse binaire



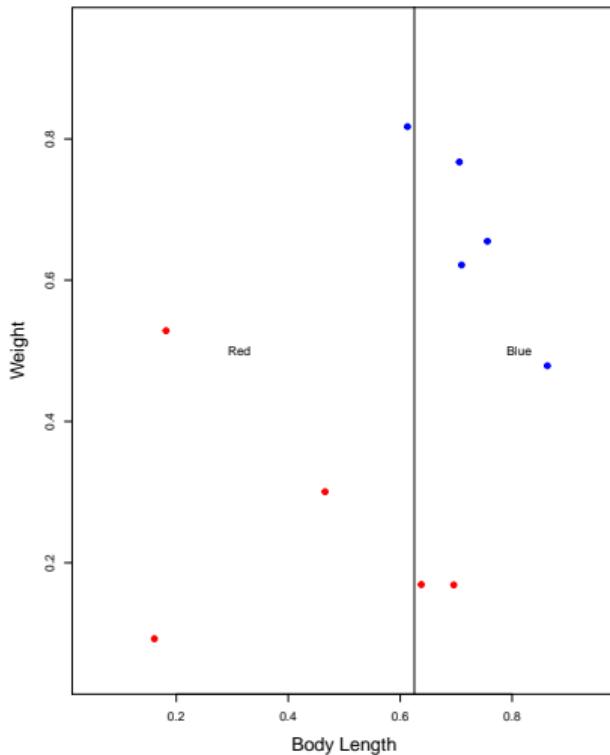
## Exemple: prédition continue et réponse binaire



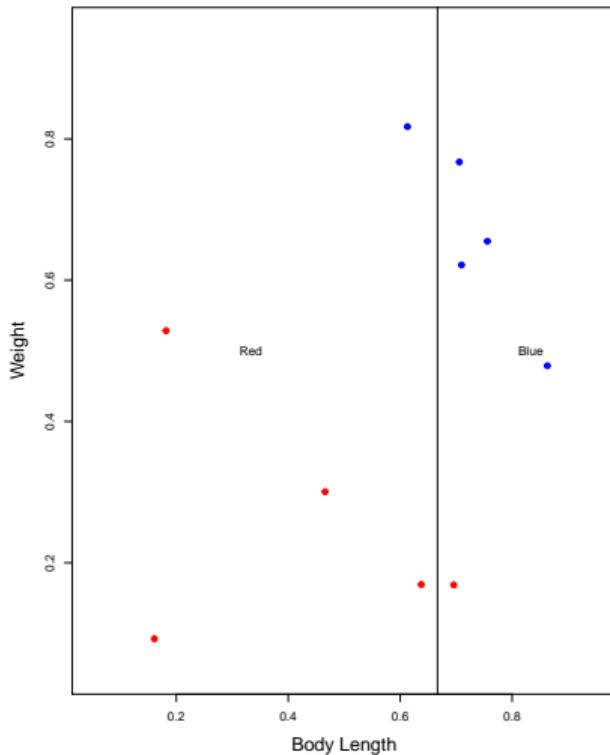
## Exemple: prédition continue et réponse binaire



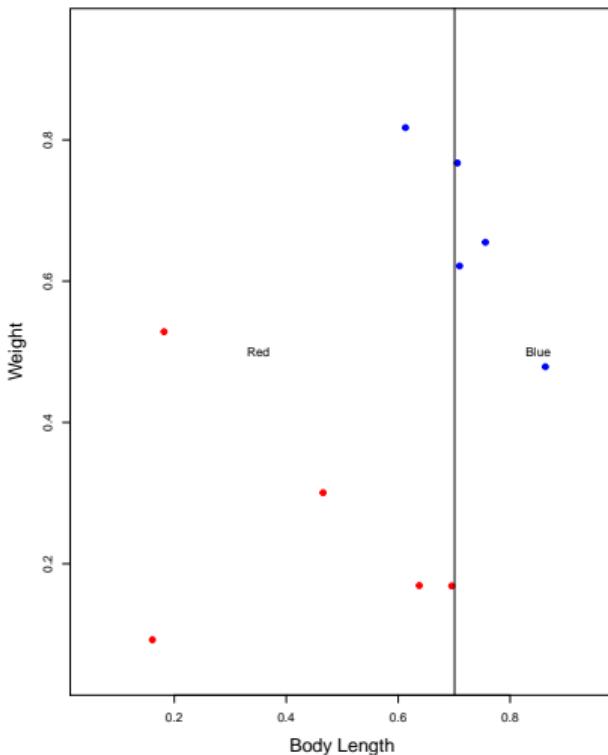
## Exemple: prédition continue et réponse binaire



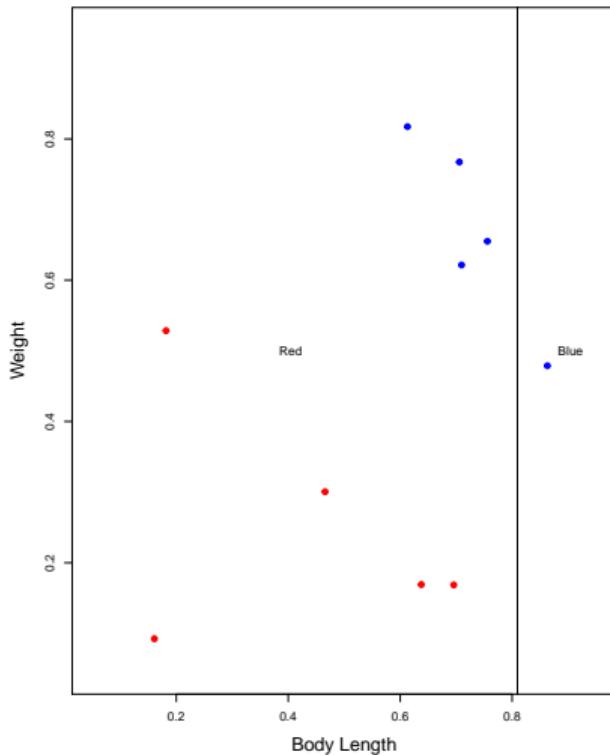
## Exemple: prédition continue et réponse binaire



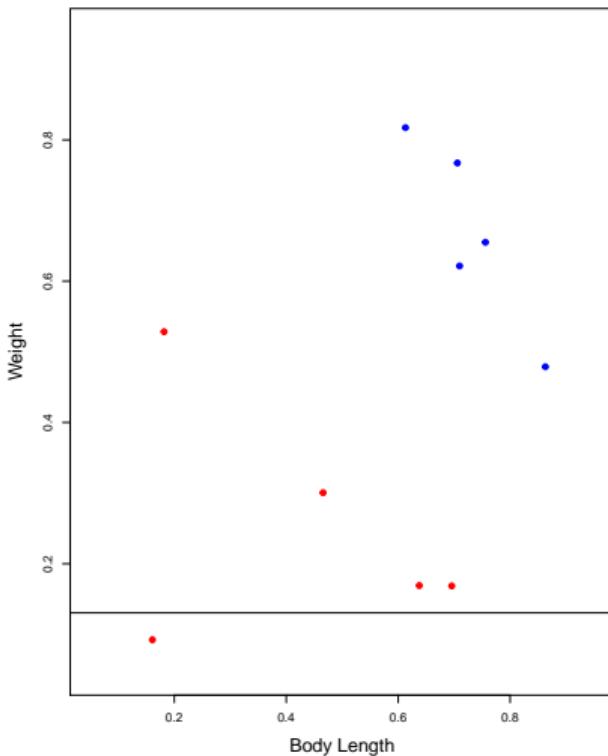
## Exemple: prédition continue et réponse binaire



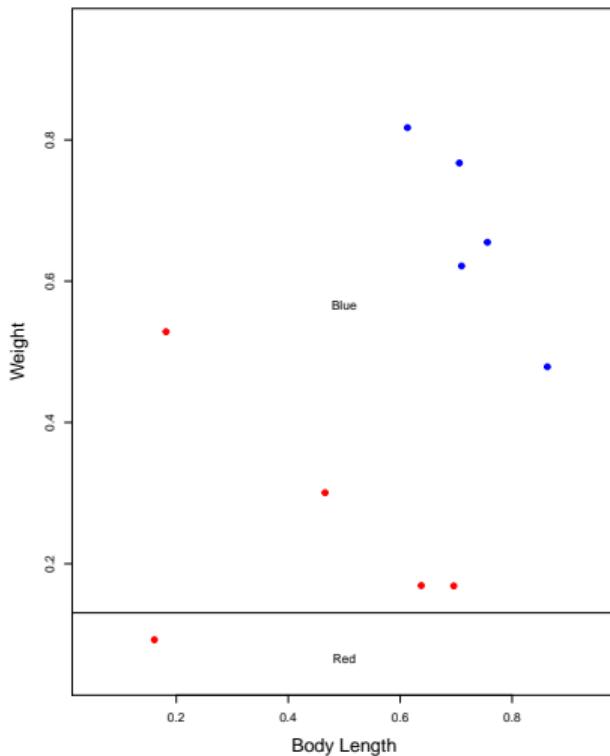
## Exemple: prédition continue et réponse binaire



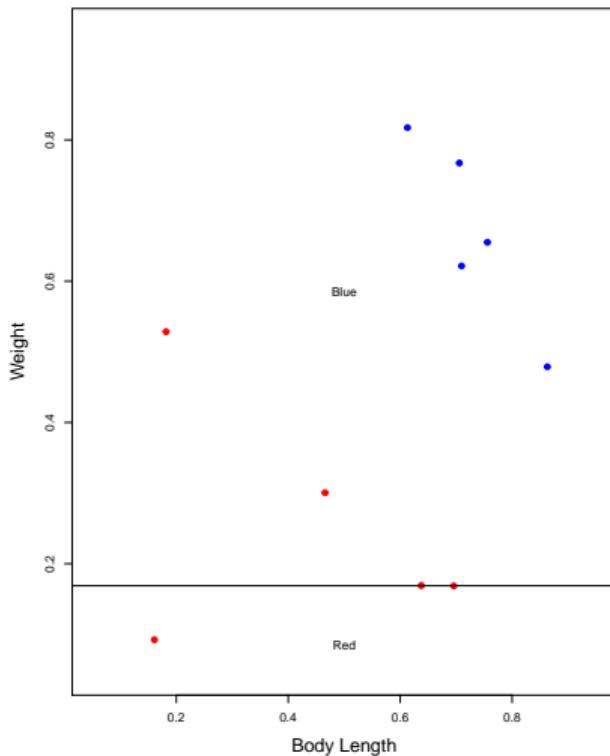
## Exemple: prédition continue et réponse binaire



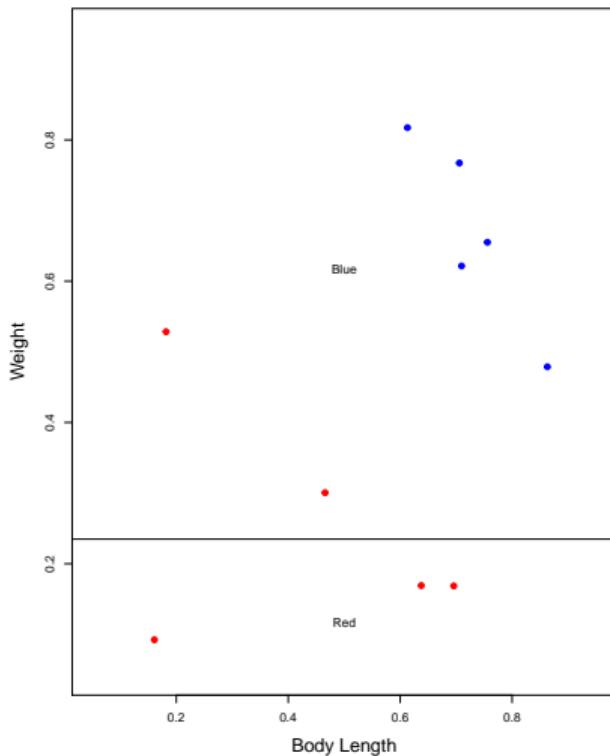
## Exemple: prédition continue et réponse binaire



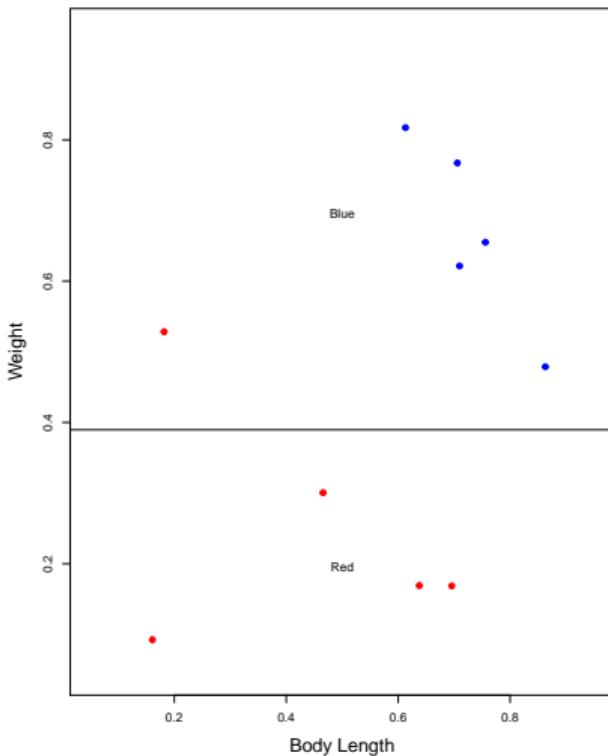
## Exemple: prédition continue et réponse binaire



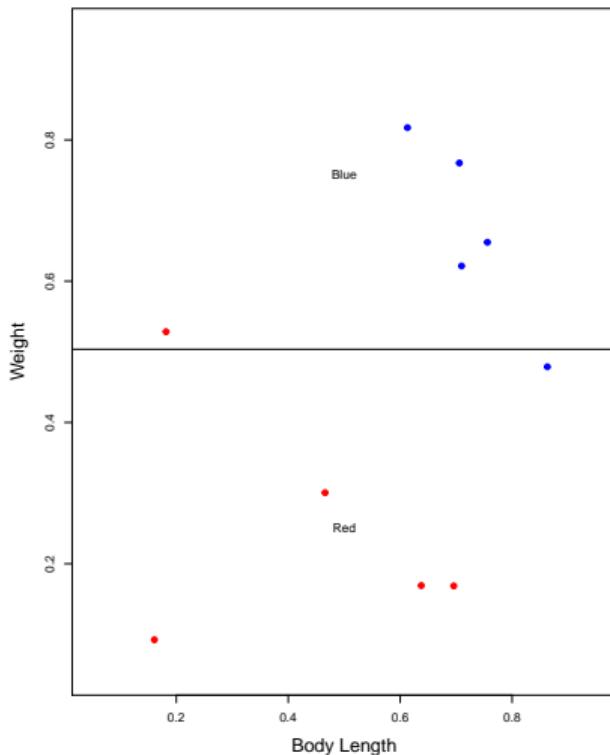
## Exemple: prédition continue et réponse binaire



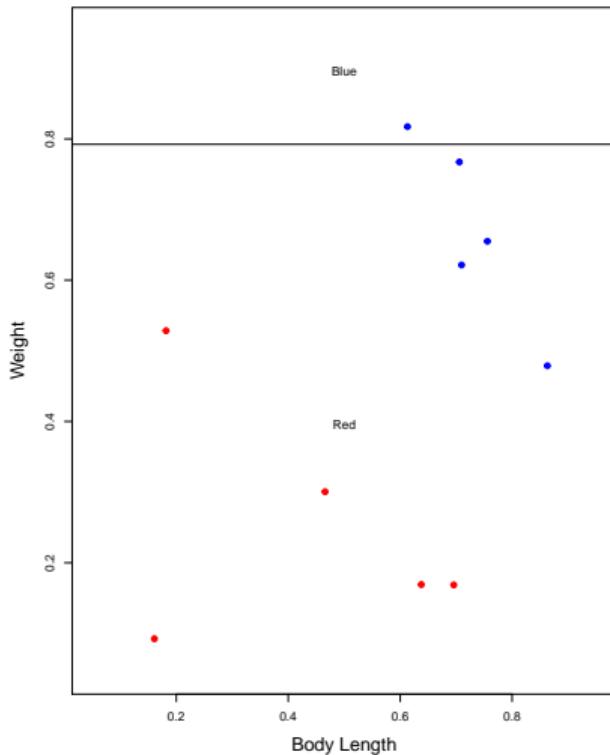
## Exemple: prédition continue et réponse binaire



## Exemple: prédition continue et réponse binaire



## Exemple: prédition continue et réponse binaire



## Deux éléments sur quatres

Jusqu'à maintenant, on sait comment former l'ensemble des partitions binaires à évaluer (premier ingrédient).

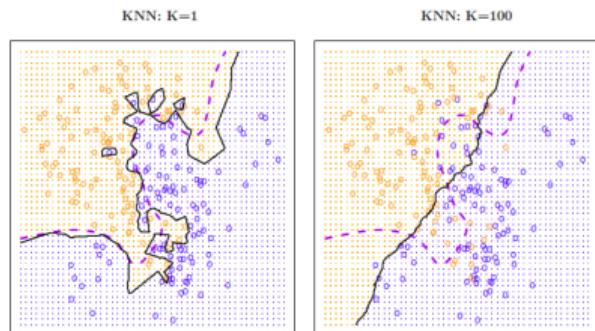
On sait aussi comment choisir la meilleure partition binaire, celle qui crée les régions avec les réponses les plus homogènes.

# Une règle d'arrêt

*Déterminer quand arrêter de partitionner*

Il nous faut une règle d'arrêt, sinon le processus continu jusqu'à ce que chaque région contiennent exactement une observation de l'échantillon  $S_{ent}$  (pour les variables continues).

- ▶ Cela cause du surapprentissage: pensez à la méthode des K plus proche voisin avec  $k = 1$ .



**FIGURE 2.16.** A comparison of the KNN decision boundaries (solid black curves) obtained using  $K = 1$  and  $K = 100$  on the data from Figure 2.13. With  $K = 1$ , the decision boundary is overly flexible, while with  $K = 100$  it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.

# Une règle d'arrêt

Les règles d'arrêt typique sont:

- ▶ (1) Lorsque les régions sont homogènes (si  $y$  est catégoriel)

Ou

- ▶ (2) Jusqu'à ce qu'il y ai moins de  $\beta$  observations ( $\beta$  ici est ce que l'on appelle un hyper-paramètre.)

## Partitionnement binaire récursif

Jusqu'à maintenant, nous savons comment créer nos options de partitionnement, la choisir en minimisant l'hétérogénéité et finalement quand arrêter ce processus récursif de partitionnement binaire.

Ne reste qu'à déterminer les valeurs  $c_j$  associées à chacune des régions.

Rappel:  $\hat{Y}_i = \sum_{j=1}^m c_j \mathbf{1}_{x_i \in R_j}$

## La réponse dans un arbre de décision

*Comment prédire la réponse dans chaque région*

- ▶ Si la réponse  $y$  est continue alors,  $c_j = \frac{1}{n_j} \sum_{i \in R_j} y_i$  est déterminé, la moyenne des observations  $y_i$  de l'échantillon dans la région  $R_j$ .
- ▶ Si la réponse  $y$  est catégoriel,  $c_j = \operatorname{argmax}_c \hat{p}_{jc}$  soit la classe majoritaire dans la région  $R_j$ .

# Prédiction

- ▶ Pour la prédiction, on prend une nouvelle observation disons  $x_0$
- ▶ Puis on détermine dans quel région l'observation appartient (soit par l'arbre ou bien par les indicatrices)
- ▶ Finalement on prédit  $\hat{y}_0 = c_r$  si  $x_0 \in R_r$

## 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 prédictive 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

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

On parle d'**impureté de Gini**

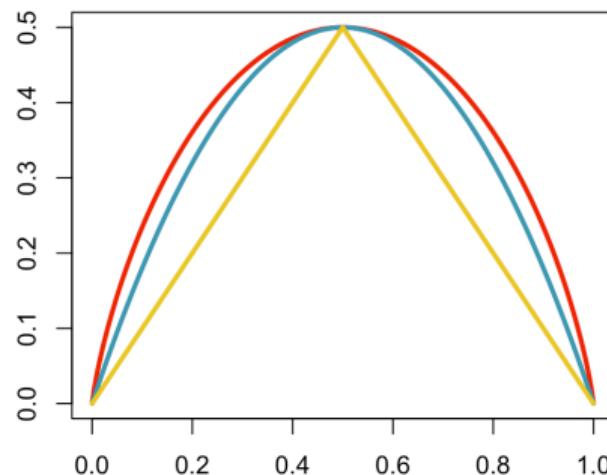
## Fonction objectif

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,  $\psi(p) = -p \ln p - (1 - p) \ln(1 - p)$



## 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

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

(ou n'importe quelle fonction d'impureté)

## Division binaire récursive

**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.

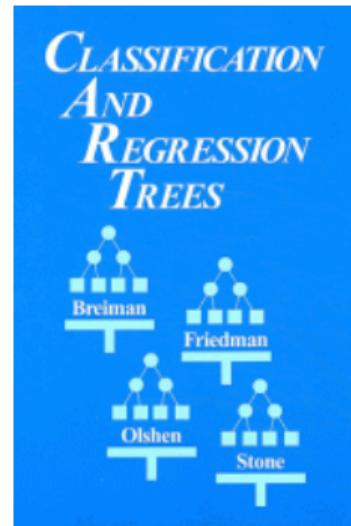
En l'absence de critère d'arrêt, l'arbre obtenu aura  $n$  feuilles ( $n$  n'étant pas nécessairement le nombre d'observations dans la base de données) → surapprentissage.

# 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  $Y$ ,

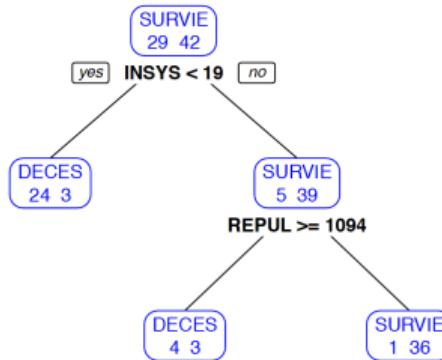
→ donne une première division de l'échantillon

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



## Subdivisionner l'espace: une variable explicative

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

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

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

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

## Subdivisionner l'espace: une variable explicative

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

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

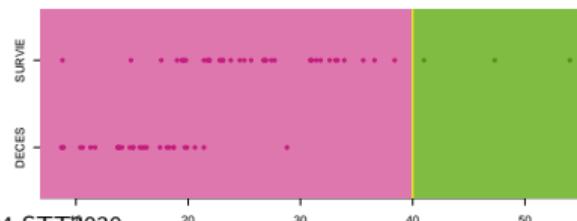
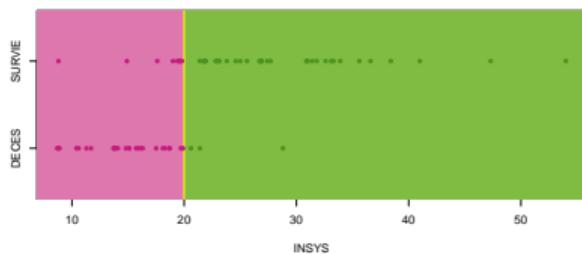
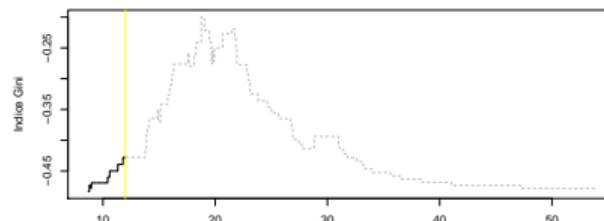
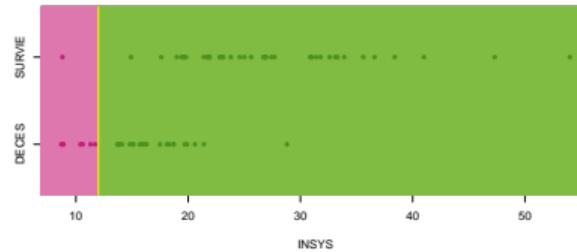
		$\tilde{X} = L$	$\tilde{X} = R$	
		$X \leq s$	$X > s$	
$Y = 0$	$n_{L,0}$	$n_{R,0}$	$n_{\cdot,0}$	
	$n_{L,1}$	$n_{R,1}$	$n_{\cdot,1}$	
		$n_{L,\cdot}$	$n_{R,\cdot}$	$n$

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

# Subdivisionner l'espace: une variable explicative

## Découpage et indice de Gini

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



# Subdivisionner l'espace: une variable explicative

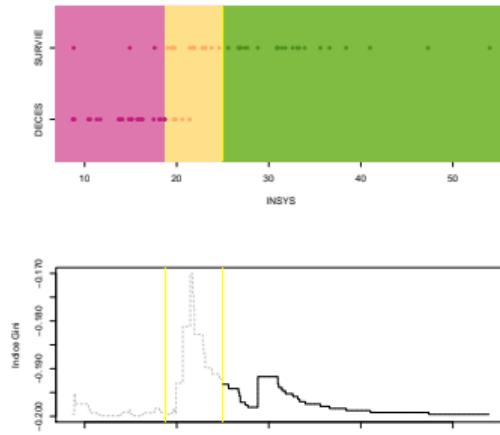
On fixe  $s$ , on cherche un second découpage,

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

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

Découpage et indice de Gini

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



# Subdivisionner l'espace: une variable explicative

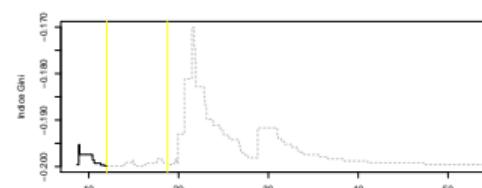
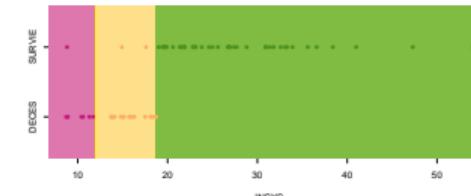
On fixe  $s$ , on cherche un second découpage,

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

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

Découpage et indice de Gini

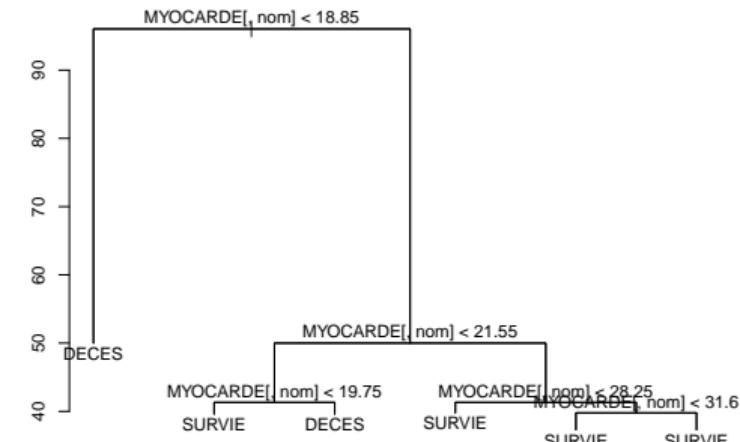
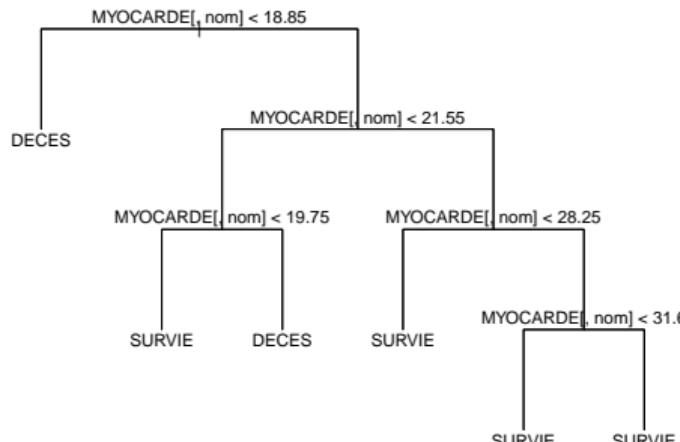
$$\sum_{x \in \{A, B, C\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0, 1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left( 1 - \frac{n_{x,y}}{n_{x,\cdot}} \right)$$



# 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)



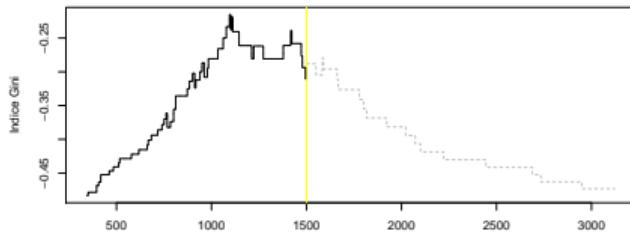
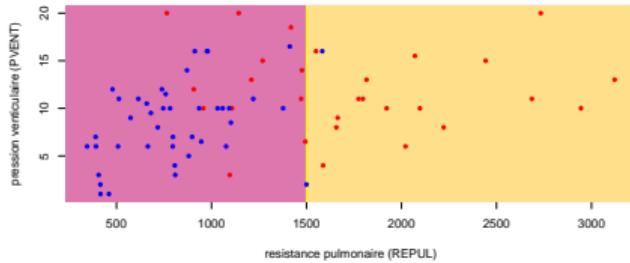
## Cas de plusieurs variables quantitatives

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

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

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

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



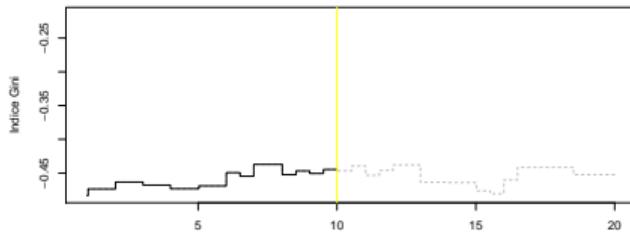
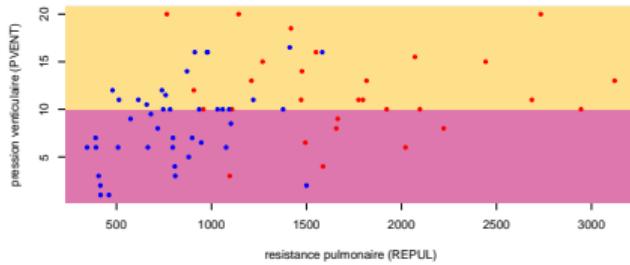
## Cas de plusieurs variables quantitatives

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

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

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

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



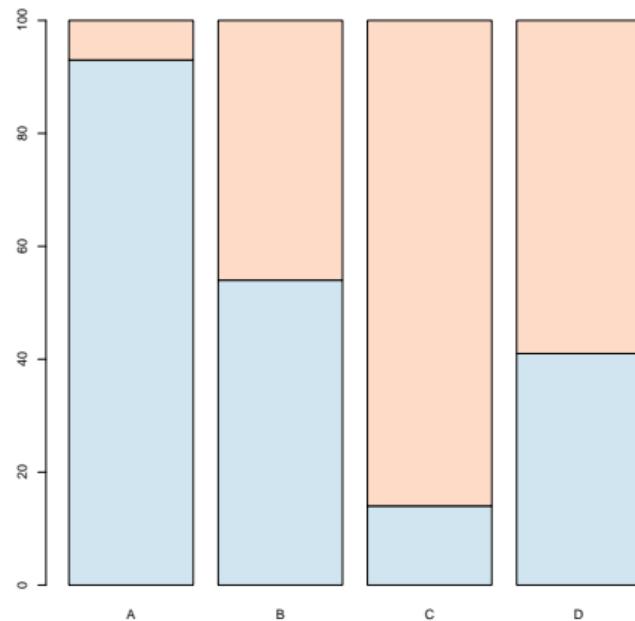
# Cas de variables qualitatives

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

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

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

$$\left\{ \begin{array}{l} \{(a, b), c, d\} \quad \{(a, d), b, c\} \quad \{(a, d), b, c\} \\ \{(b, c), a, d\} \quad \{\textcolor{red}{(b, d), a, c}\} \quad \{(c, d), a, b\} \\ \{(b, c, a), d\} \quad \{\textcolor{red}{(b, c, d), a}\} \quad \{(b, c), (a, d)\} \end{array} \right.$$

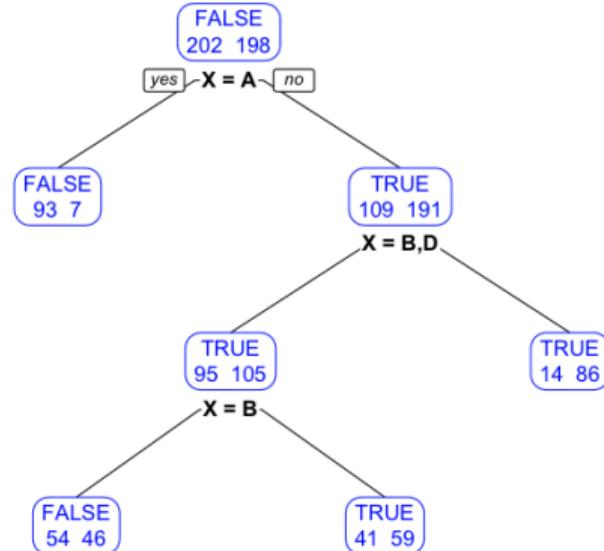


# Cas de variables qualitatives

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

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

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



# Cas de variables qualitatives

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

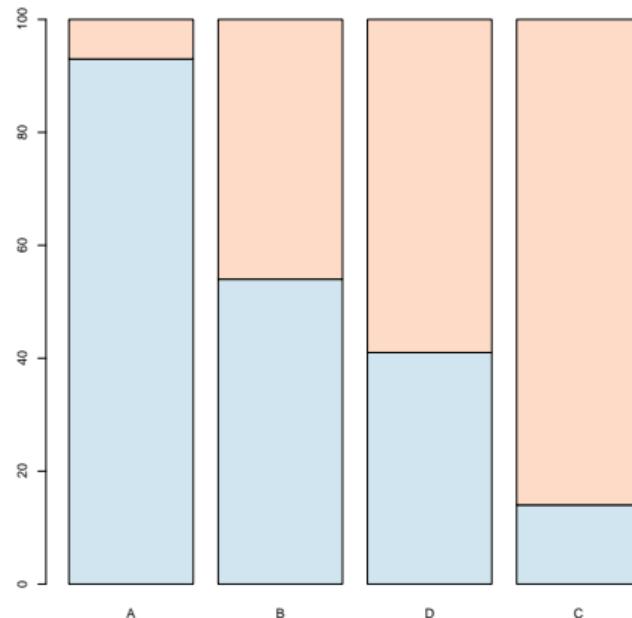
Mais classiquement, on va ordonner les modalités

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

puis trouver le découpage optimale

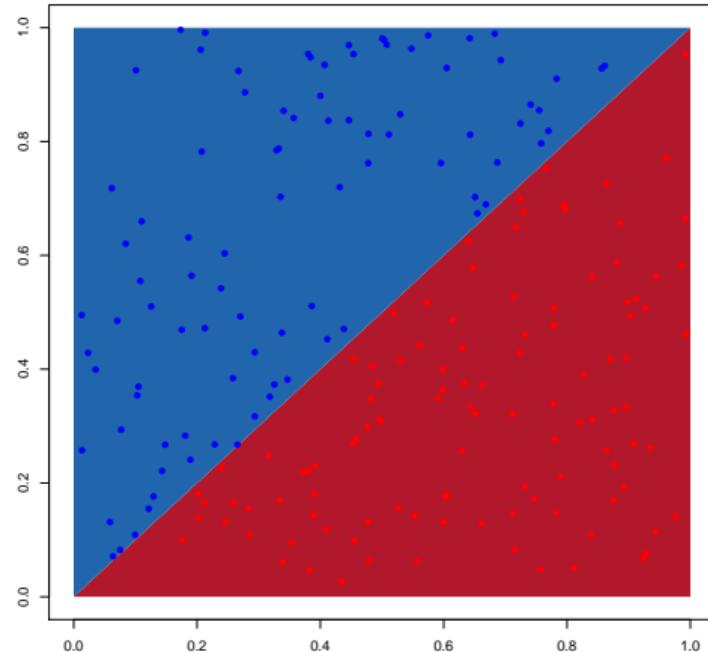
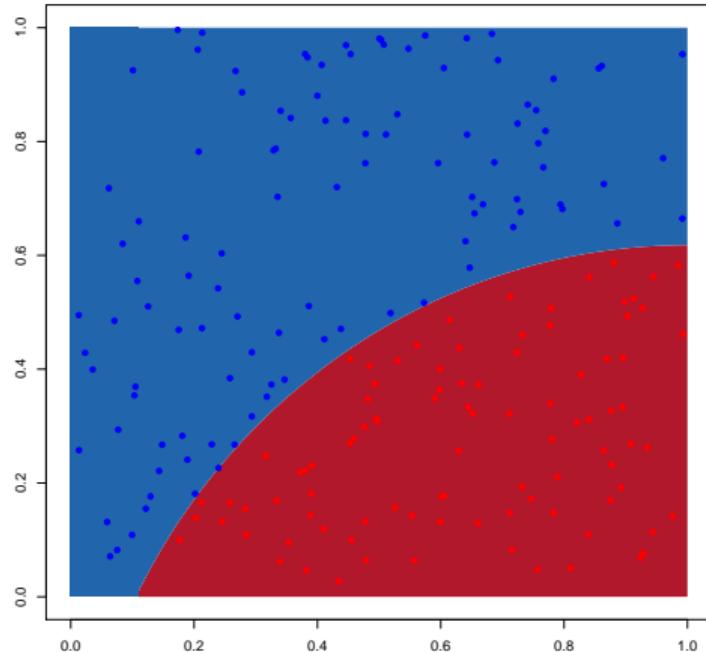
$$\{(a), (b, d, c)\} \quad \{(a, b), (d, c)\} \quad \{(a, b, d), (c)\}$$

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



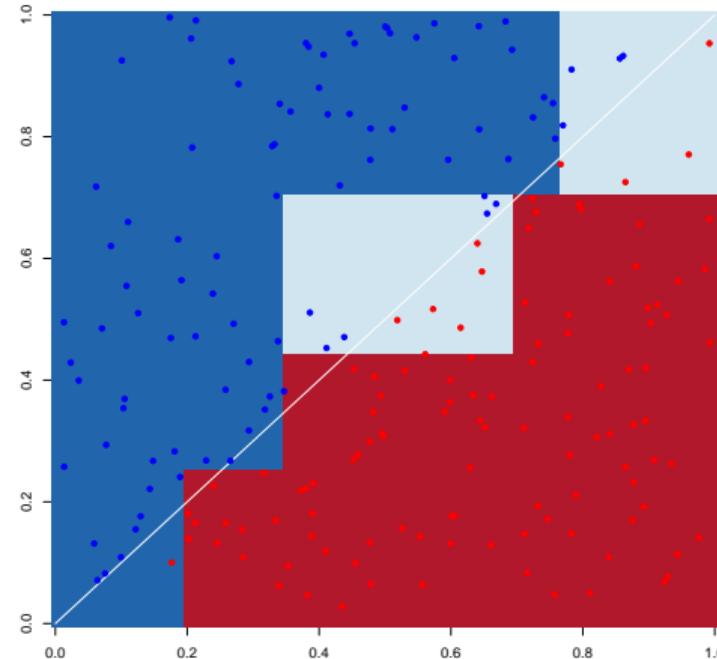
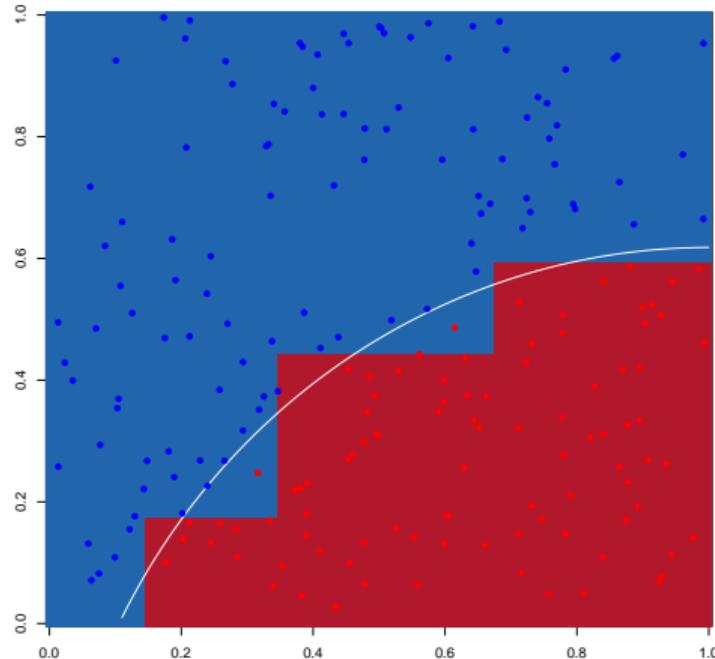
# Méthode CART et extensions

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



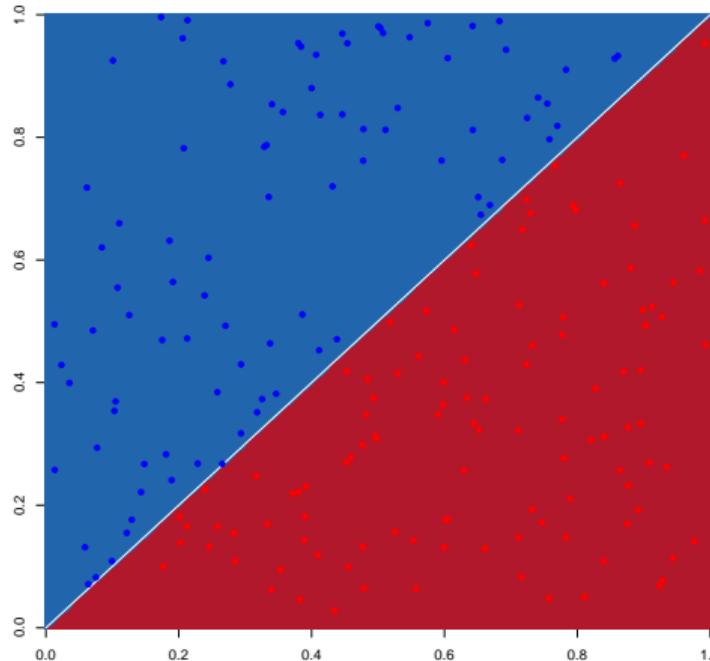
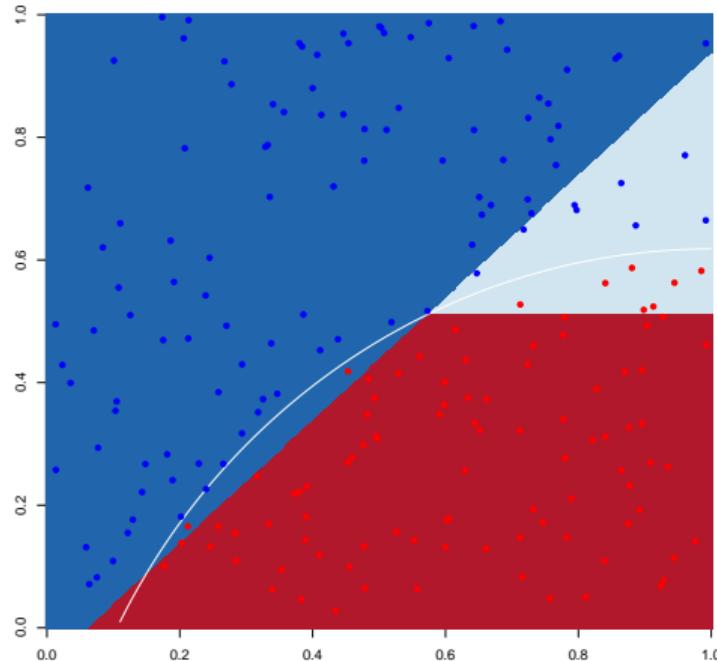
# Méthode CART et extensions

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



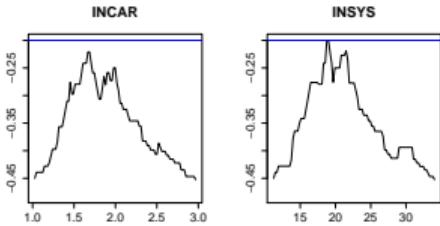
# Méthode CART et extensions

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

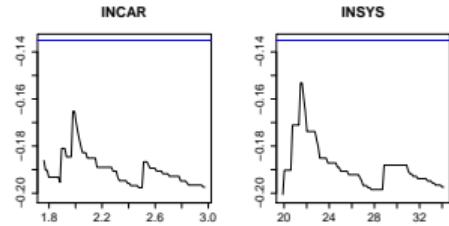


# Mise en oeuvre

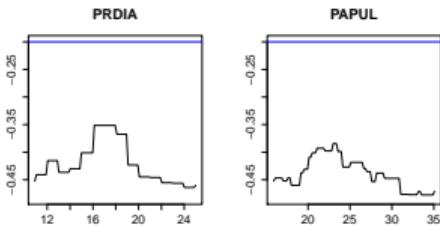
$$N_L: \{x_{i,j} \leq s\} \quad N_R: \{x_{i,j} > s\}$$



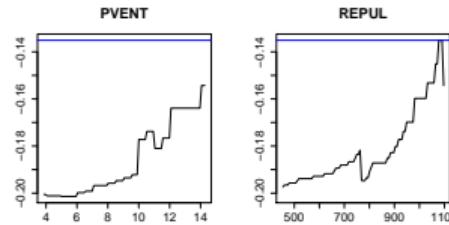
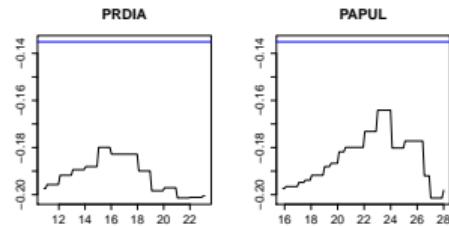
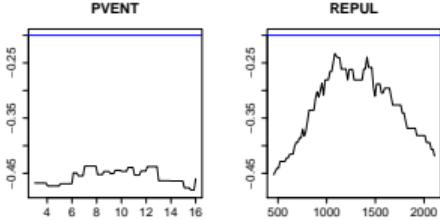
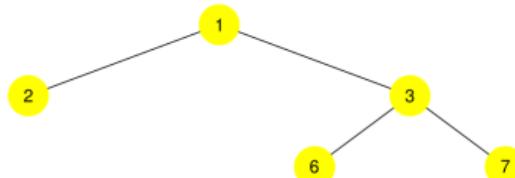
résoudre  $\max_{j \in \{1, \dots, k\}, s} \{\mathcal{I}(N_L, N_R)\}$



← premier split



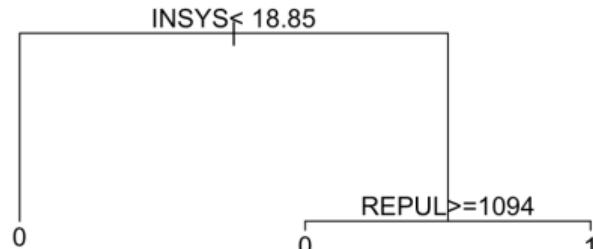
second split →



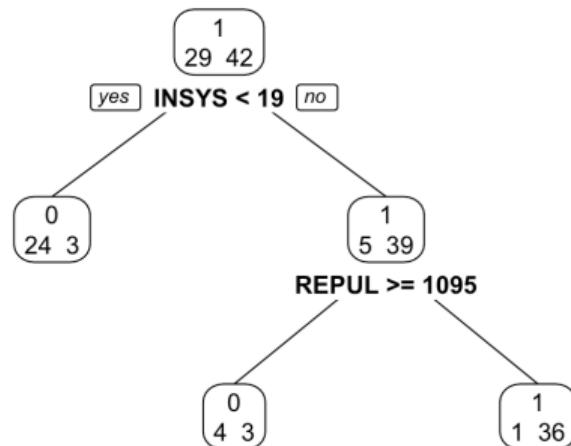
# Arbres avec R

```
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)
```



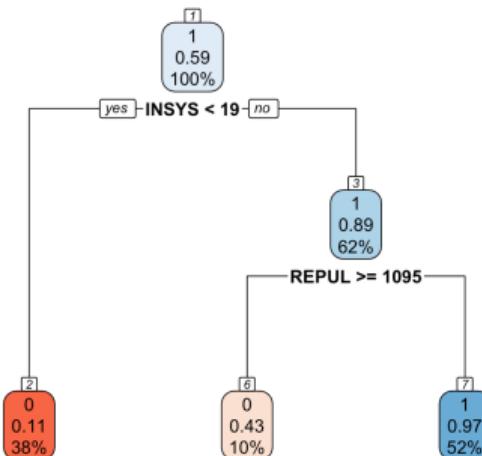
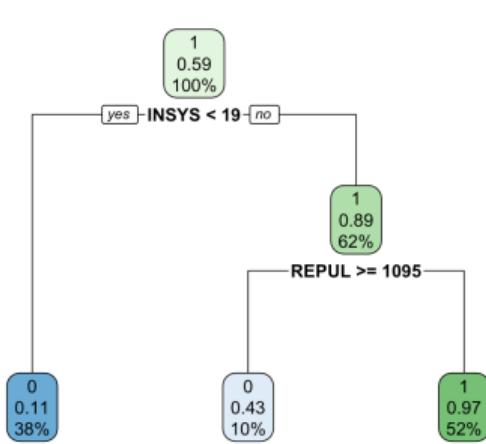
```
1 > library(rpart.plot)
2 > prp(cart, type=2, extra=1)
```



# 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) *
```

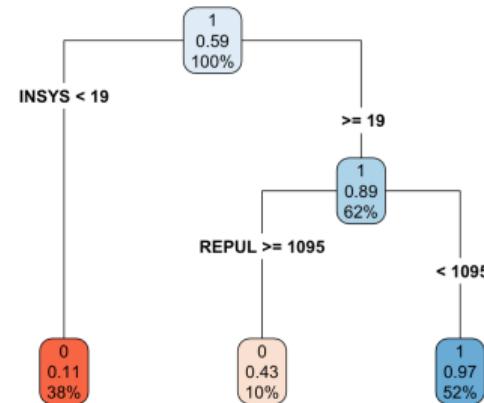
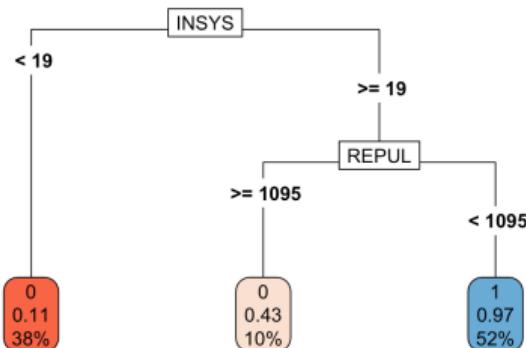
```
1 > rpart.plot(cart)           1 > rpart.plot(cart, box.palette="RdBu", nn=TRUE)
```



# Arbres avec R

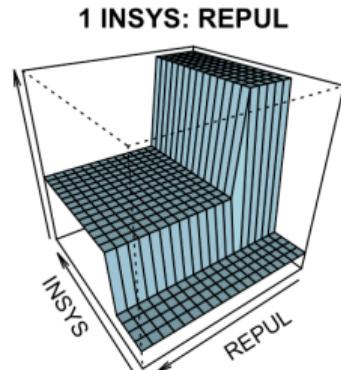
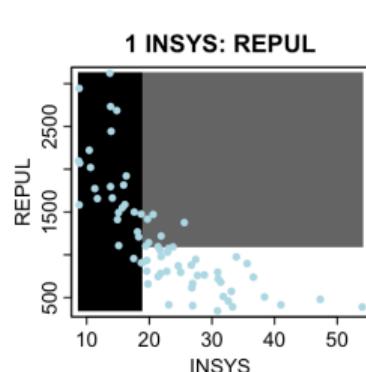
```
1 > rpart.rules(cart, extra = 4, cover = TRUE)
2 PRONO 0    1                                cover
3   0 [.89 .11] when INSYS < 19             38%
4   0 [.57 .43] when INSYS >= 19 & REPUL >= 1095 10%
5   1 [.03 .97] when INSYS >= 19 & REPUL < 1095 52%
```

```
1 > rpart.plot(cart, box.palette="RdBu" 1 > rpart.plot(cart, box.palette="RdBu",
              ", type=5)                           type=4)
```

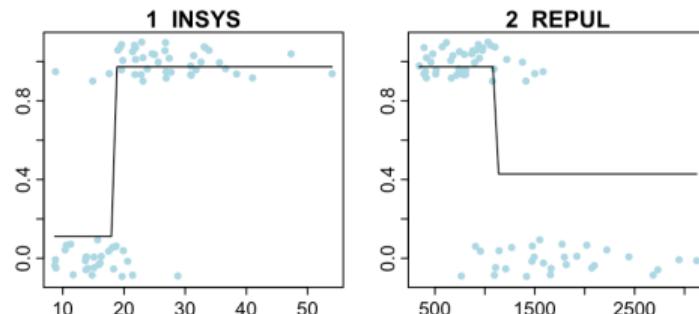


# Arbres avec R

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



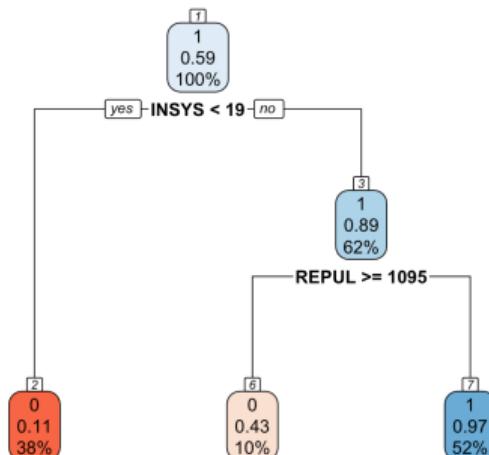
```
1 > plotmo(cart, type = "prob",
  nresponse = "1", type2 = "image",
  ngrid2 = 200)
```



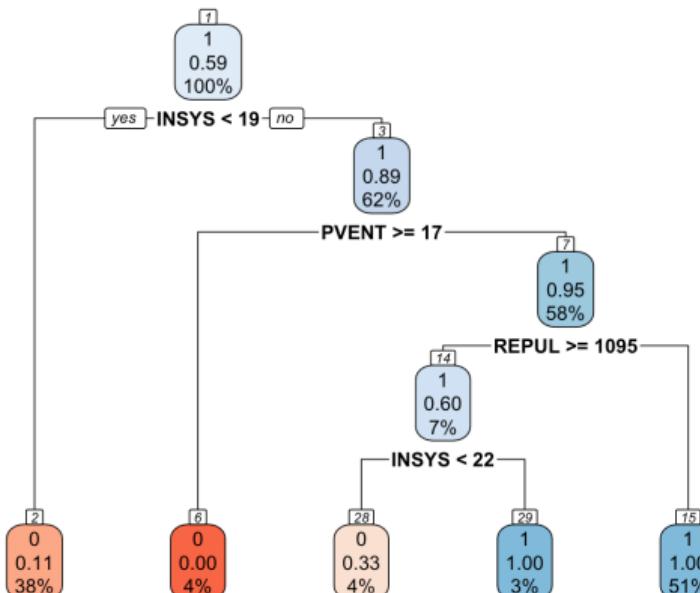
```
1 > path2root = function(node){
2   if(node == 1) node
3   else c(node, path.to.root(node %% 2))}
```

# Arbres avec R

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

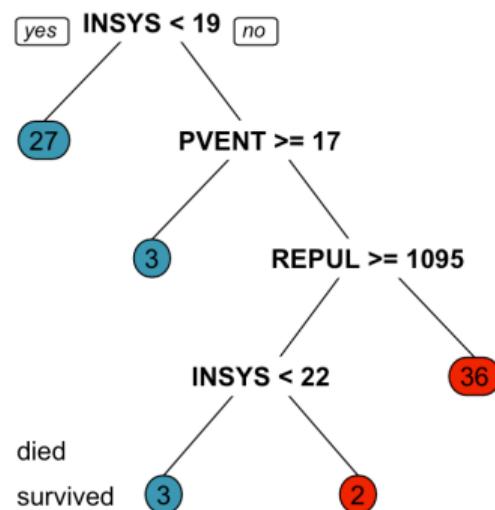


```
1 > cart = rpart(PRONO~., data=  
      myocarde, minsplit = 5,  
      cp = 0.01)  
2 rpart.plot(cart, box.palette="RdBu"  
      ", nn=TRUE)
```

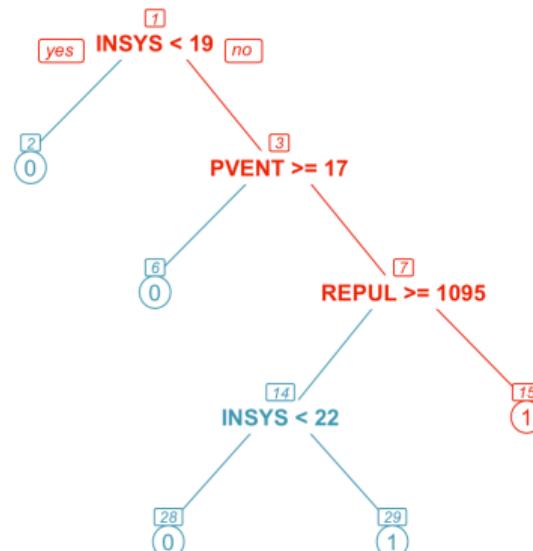


# Arbres avec R

```
1 > boxcols = c("red","blue")
   [cart$frame$yval]
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.

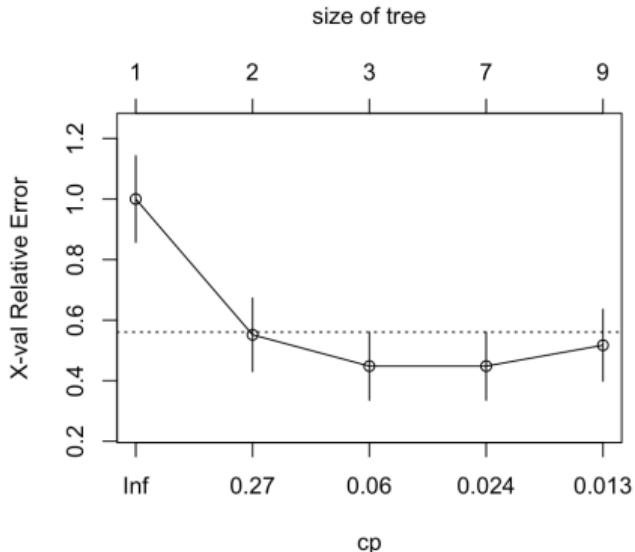
Formellement, pour savoir si on coupe une feuille  $\{N\}$  en deux branches  $\{N_L, N_R\}$ , on mesure la variation d'impureté

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

# Critère d'arrêt

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 ,  
2                      minsplit=3)  
3 > 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

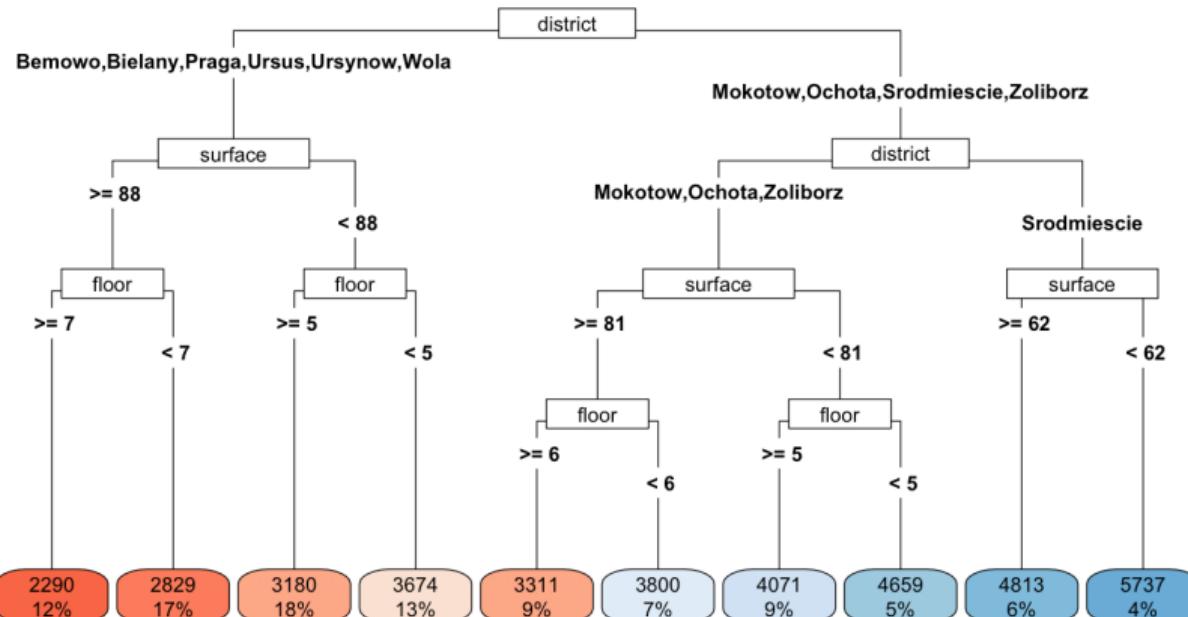
$$\sum_{m=1}^{|\mathcal{T}|} \sum_{i: \mathbf{X}_i \in R_m} (Y_i - \hat{Y}_{R_m})^2 + \alpha |\mathcal{T}|,$$

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.

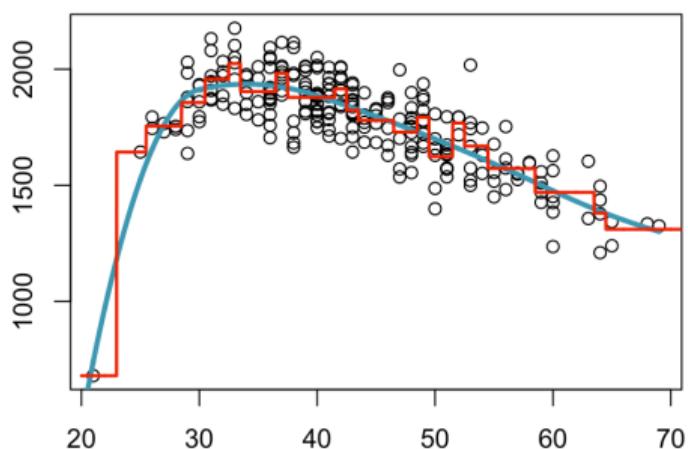
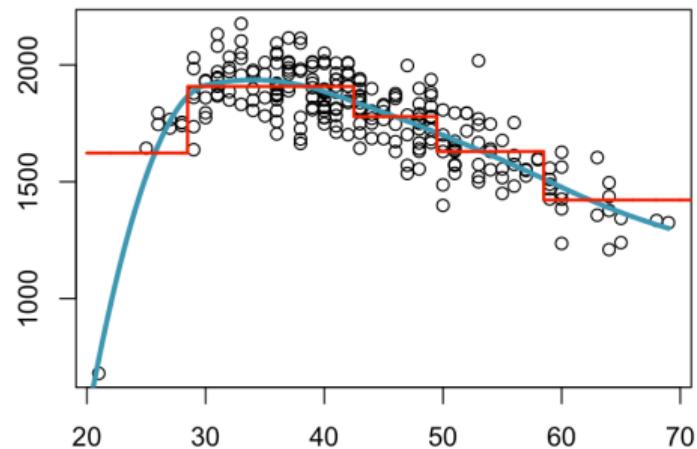
# Arbres de régression, $y \in \mathbb{R}$

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



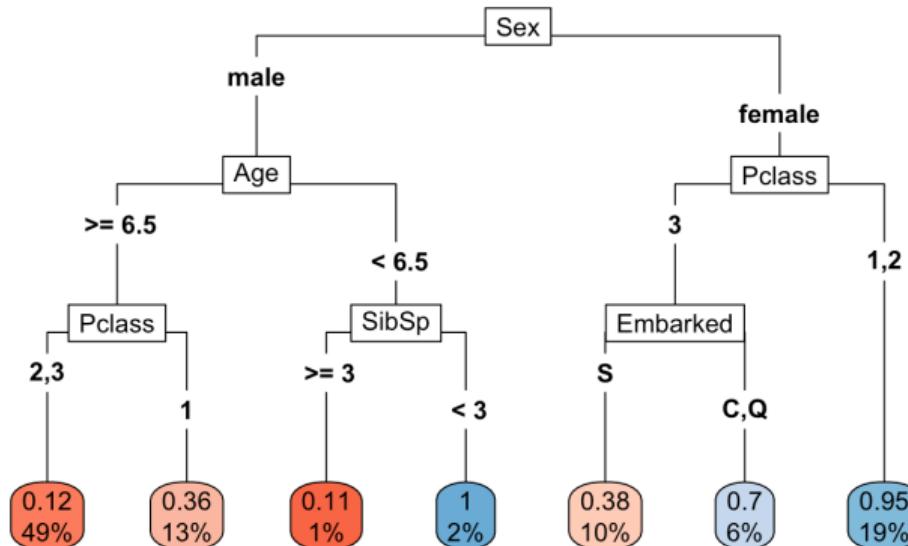
# Arbres de régression, $y \in \mathbb{R}$

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



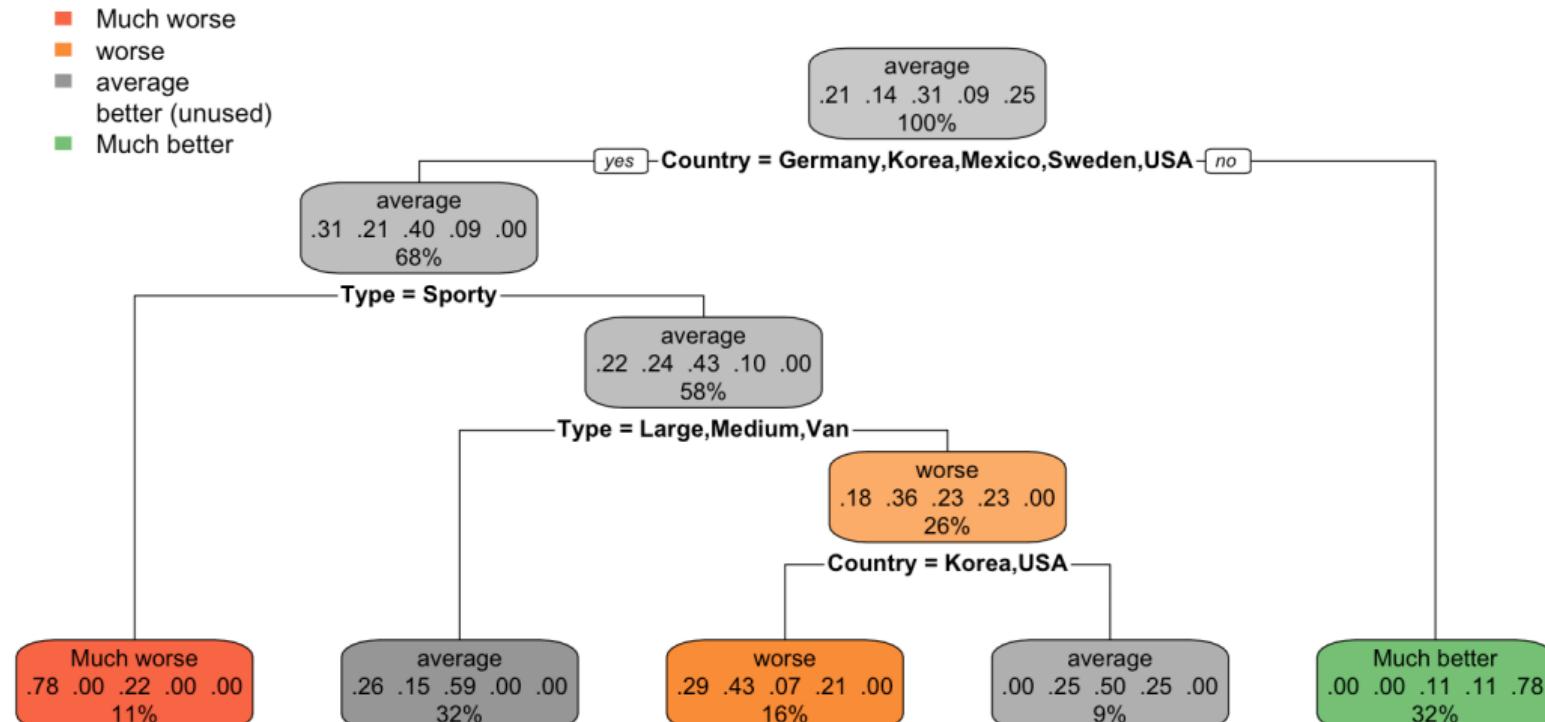
# Arbres de classification, $y \in \{0, 1\}$

```
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)
```



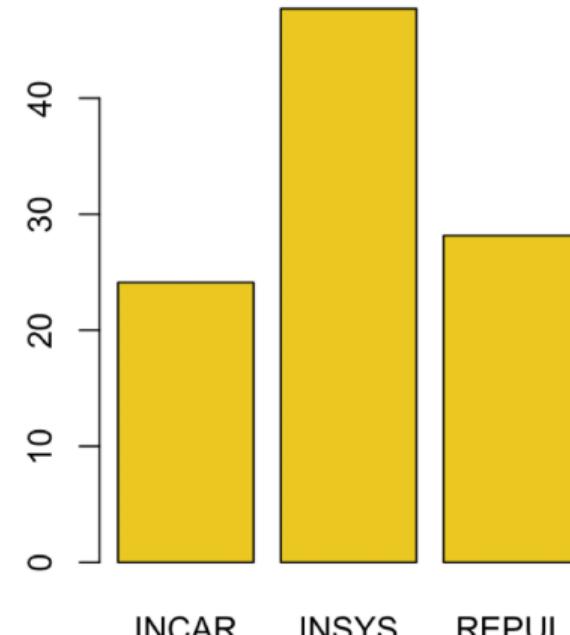
# Instabilité des arbres

Considérons ici plusieurs bases d'apprentissage

```
1 > variable=rep(NA,10000)
2 > for(i in 1:10000){
3 + arbre = rpart(PRONO~, data=
+   myocarde[sample(1:71,size=47),])
4 + variable[i] = as.character(
+   arbre$frame[1,"var"])
5 + }
6 > table(variable)/100
7 INCAR INSYS REPUL
8 24.12 47.72 28.16
```

La variable utilisée au premier noeud est

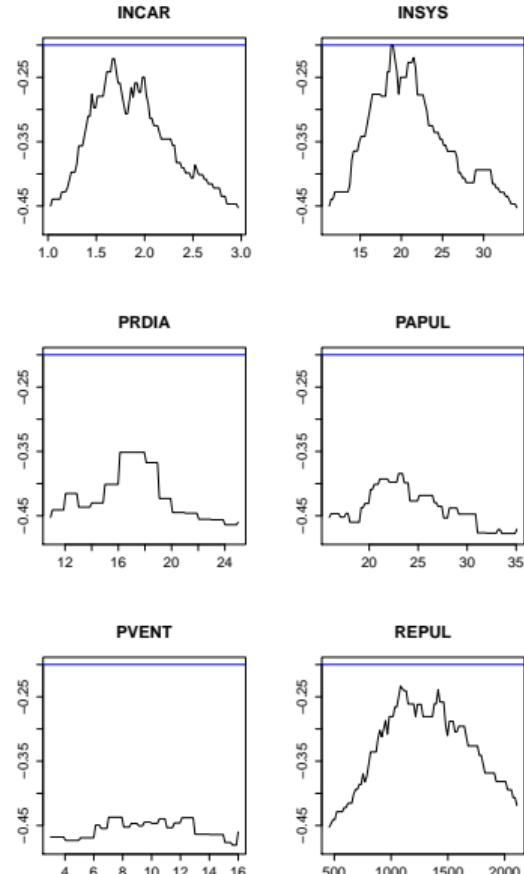
- ▶ INSYS (47.7%)
- ▶ REPUL (28.2%)
- ▶ INCAR (24.1%)



# Importance des variables

On avait déjà intuité l'importance de ces trois variables sur le graphique

```
1 > cart = rpart(PRONO~., myocarde)
2 > split = summary(cart)$splits
3 > split
4       count ncat      improve
5 INCAR     71    -1  0.58621312
6 REPUL     71     1  0.55440034
7 PRDIA     71    -1  0.54257020
8 PAPUL     71     1  0.27284114
9 PVENT     71     1  0.20466714
```



et on peut calculer les mêmes quantités à chaque noeud

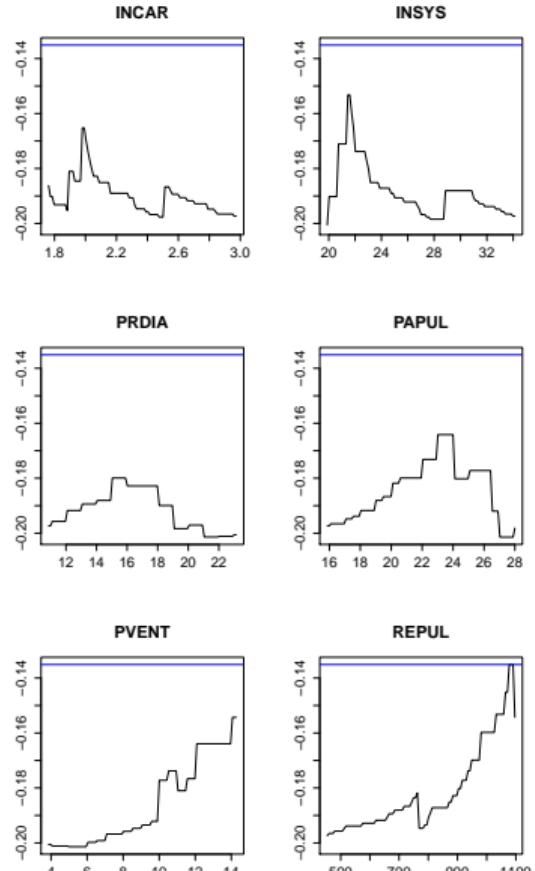
# Importance des variables

pour le second noeud

```
1 > split
2       count ncat    improve
3 REPUL     27      1 0.18181818
4 PVENT     27     -1 0.10803571
5 PRDIA     27      1 0.10803571
6 PAPUL     27      1 0.10803571
7 INCAR     27      1 0.04705882
```

etc

```
1 > cart$variable.importance
2   INSYS  REPUL  INCAR  PAPUL  PRDIA
3   FRCAR  PVENT
4 20.113 19.132 15.895  5.959  5.214
5   3.725  0.498
```



# Les arbres de décision,

... c'est bon!

- ▶ C'est une technique qui vaut la peine d'être connue.
- ▶ Aucune hypothèse nécessaire.
- ▶ Visualization intuitive.
- ▶ **Incorporation naturel d'interaction de haut degré.**

... c'est mauvais!

- ▶ Procédure prompt au surapprentissage.
- ▶ Procédure instable.

## Contrer le surapprentissage par la régularisation

Une technique standard est d'*émonder* les arbres en *coupant* les branches les moins profitables.

Soit un arbre de décisions, on pénalise un trop grand nombre de région  $m$ .

- ▶ Arbre de régression:  $\sum_{r=1}^m \sum_{x_i \in R_r} (y_i - \hat{y}_{R_r})^2 + \lambda m$
- ▶ autrement dit “erreur  $+\lambda m$ ”.

## Contrer le surapprentissage par la régularisation

Avec  $\lambda$  très grand, on émonde tout, on retourne à une seule région, et on retrouve  $\hat{y} = \bar{y}$ . (Comme dans le lasso/ridge) Ici, pas de surraprentissage.

Si  $\lambda$  est très petit on retrouve la solution d'arbre de décision standard. Sans règle d'arrêt, on pourra former des régions avec une seule observation tel que l'erreur de prédiction (sur  $S_{ent}$ ) est de 0.

## Contrer le surapprentissage par la régularisation

Bien qu'il peut paraître difficile d'optimiser la fonction objective régularisé, il y existe un petit truc qui nous permet de facilement re-crée l'équivalent du *chemin lasso*.

C'est-à-dire que nous obtiendrons une séquence d'arbre imbriqué, qui part d'un arbre très grand avec beaucoup de branches jusqu'à une arbre dit tronc, avec une seule région.

En partant d'un arbre très grand (disons  $\beta$  très petit). Nous allons progressivement émonder les branches en choisissant toujours celle qui avait réduit le moins la diversité dans les branches.

## Contrer le surapprentissage par la régularisation

Une autre perspective est que l'on regroupe les deux régions qui augmentent le moins l'erreur total lorsque recombinées.

Ainsi, on peut créer une séquence d'arbres imbriqués.

On pourrait démontrer que cela est équivalent à optimiser :  $\sum_{r=1}^m \sum_{x_i \in R_r} (y_i - \hat{y}_{R_r})^2 + \lambda m$  sur une séquence de valeur de  $\lambda$ .

## Contrer le surapprentissage par la régularisation

On doit choisir entre ces deux extrêmes, on le fera à l'aide de la validation.

Une approche intéressante:

1. Entrainer une arbre de décision avec beaucoup de branches ( $\beta$  très petit)
2. On génère la séquence d'arbre intermédiaire en émmondant les branches en commençant par celles qui diminue le moins l'hétérogénéité.
3. On évalue la performance de tous les arbres sur  $S_{val}$ .
4. On choisit l'arbre avec la meilleure performance sur  $S_{val}$ .

# Contrer le surapprentissage par la régularisation

---

## Algorithm 8.1 Building a Regression Tree

---

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
  2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of  $\alpha$ .
  3. Use K-fold cross-validation to choose  $\alpha$ . That is, divide the training observations into  $K$  folds. For each  $k = 1, \dots, K$ :
    - (a) Repeat Steps 1 and 2 on all but the  $k$ th fold of the training data.
    - (b) Evaluate the mean squared prediction error on the data in the left-out  $k$ th fold, as a function of  $\alpha$ .Average the results for each value of  $\alpha$ , and pick  $\alpha$  to minimize the average error.
  4. Return the subtree from Step 2 that corresponds to the chosen value of  $\alpha$ .
- 

Figure 8: Algorithme 8.1. Extrait de James et al. (2013).

## Contrer le surapprentissage par la régularisation

- ▶ Bien que l'émmondage fonctionne bien, il est très peu utilisé en pratique.
- ▶ C'est un processus un peu lent.
- ▶ Mais surtout, les arbres de décision sont peu utilisés en dehors de la forêt aléatoire et l'émmondage n'est pas utilisé lors de la formation de forêt.
- ▶ On verra les forêts au prochain cours!

## Comparativement aux modèles linéaires

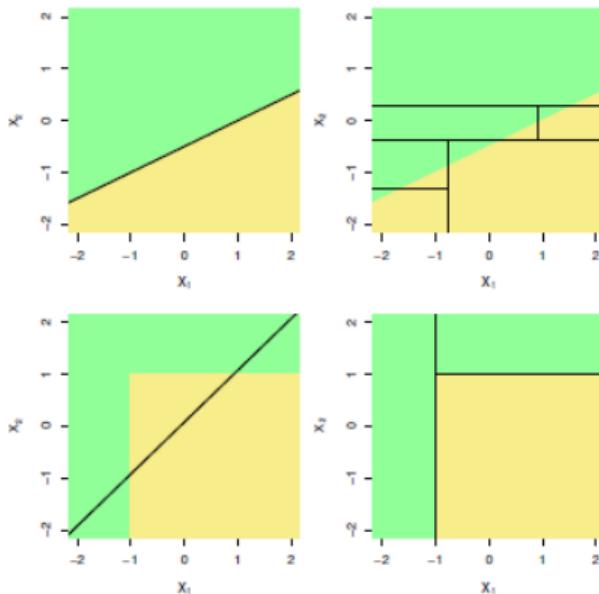


Figure 9: Figure 8.7. Extrait de ISLR.

## Comparativement aux modèles linéaires

- ▶ **Inclut naturellement les interactions.**
- ▶ Ne se base sur aucune hypothèse.
- ▶ Peut toujours être appris.
- ▶ Très variables (instable)
- ▶ Sans aucun garantie théorique.

## Exercises

- ▶ Lire le chapitre 8
- ▶ Lab: 8.3.1 et 8.3.2 (librairies différentes)
- ▶ Exercises 8.4.3, 8.4.4 et 8.4.8

## Références

- Friedman, J., Hastie, T., Tibshirani, R., et al. (2001). *The elements of statistical learning*. Springer.
- James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*, volume 112. Springer.
- Wainer, H. and Savage, S. (2008). Visual revelations: until proven guilty: false positives and the war on terror. *Chance*, 21(1):55–58.