# Apprentissage statistique: Arbres de décision

Marie-Anne.Poursat@math.u-psud.fr

Master 1 Mathématiques Appliquées
Université Paris-Saclay

11 avril 2019

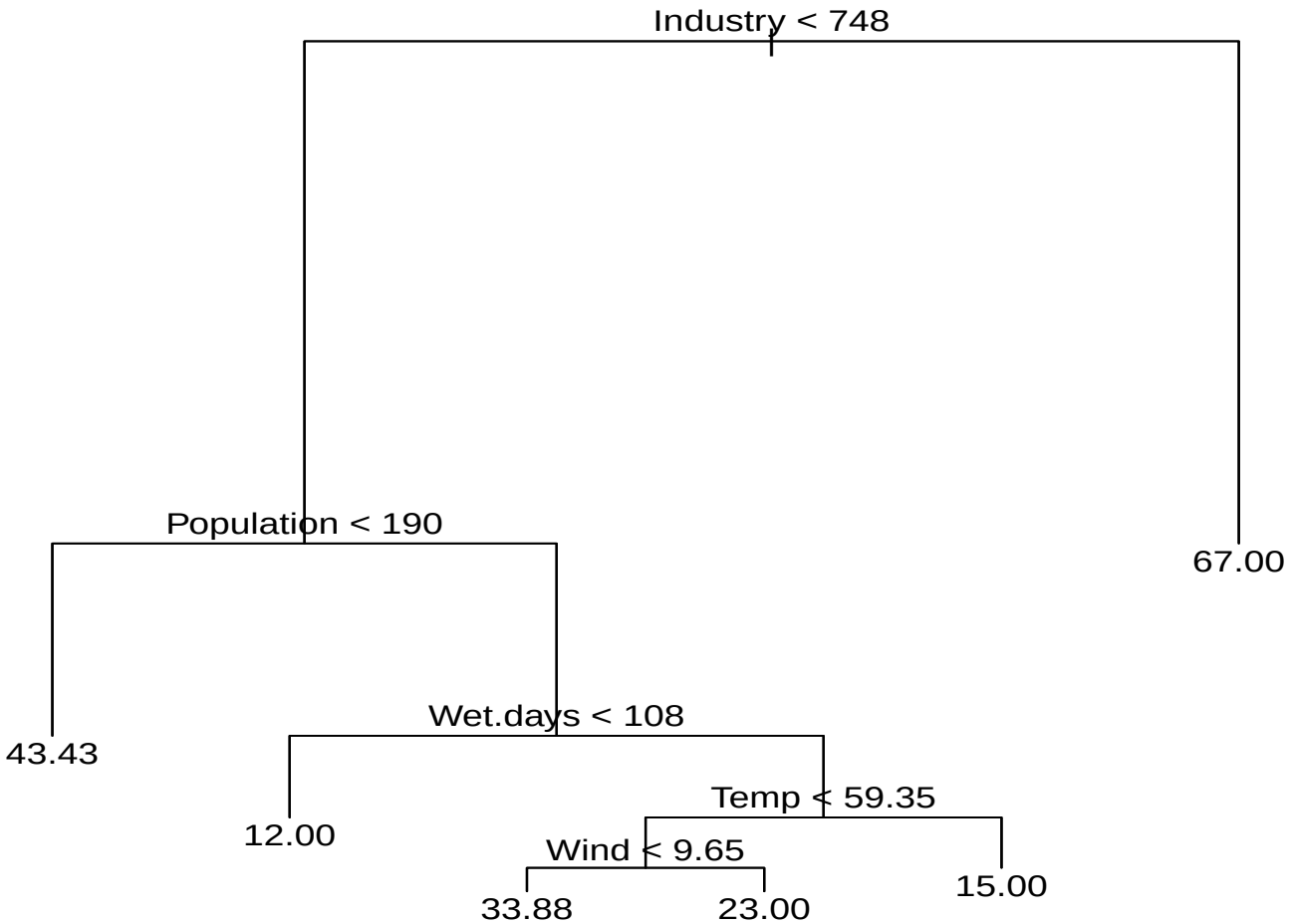
# Qui est-ce ?

- But : trouver le personnage de votre adversaire avant qu'il ne découvre le votre.
- Stratégie : à chaque tour, poser une question qui divise les personnages restants en deux groupes de taille similaire.

# Exemple : données de pollution

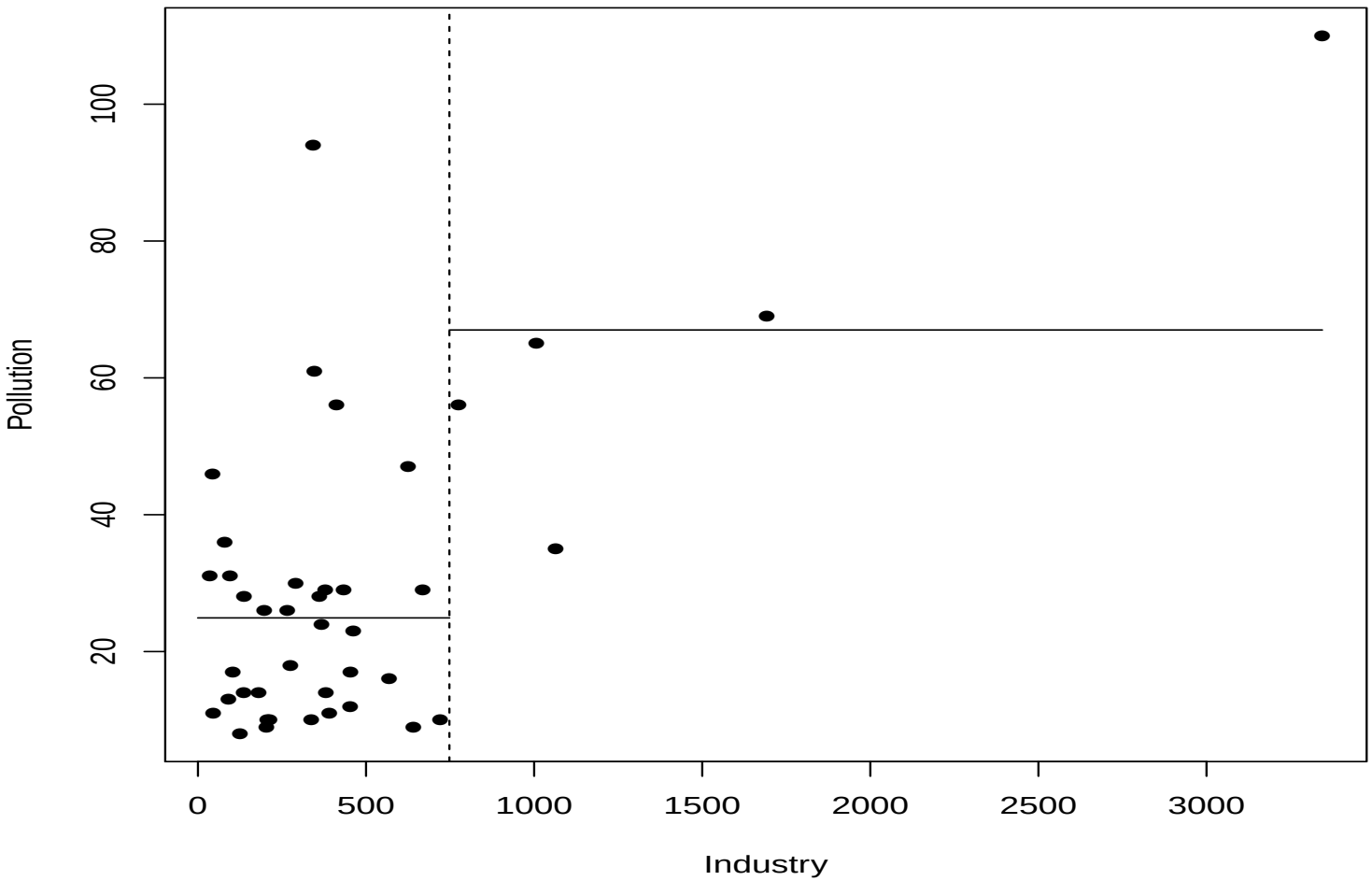
```
> Don.pollution<-read.table("Pollution.txt",header=T)
> attach(Don.pollution)
> names(Don.pollution)
[1] "Pollution"  "Temp"        "Industry"   "Population" "Wind"
[6] "Rain"       "Wet.days"

> head(Pollution)
  Pollution Temp Industry Population Wind  Rain Wet.days
1        24 61.5      368        497  9.1 48.34      115
2        30 55.6      291        593  8.3 43.11      123
3        56 55.9      775        622  9.5 35.89      105
4        28 51.0      137        176  8.7 15.17       89
5        14 68.4      136        529  8.8 54.47      116
6        46 47.6       44        116  8.8 33.36      135
```

# Exemple d'arbre de décision



# Construction de l'arbre

Partition des données selon la variable `Industry` :



# Construction de l'arbre

- Puis on recommence avec la variable `Population` sur chaque sous-ensemble des données.
- Les découpes correspondent à des hyperplans parallèles aux axes.
- Prédiction aux feuilles : moyenne des réponses pour les observations appartenant à la même région
- Pourquoi la branche droite se termine-t-elle par une feuille ? Quand décider qu'un noeud est terminal ?
- Comment les seuils sont-ils choisis ?
- Quelles variables utiliser ?

# Construction de l'arbre (régression)

But : découper l'espace des variables explicatives en régions $R_1, \ldots, R_J$ qui minimisent

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \widehat{y}_{R_j})^2$$

Problème : impraticable

Algorithme récursif (arbre binaire) :

- soit les hyperplans $R_1(j,s) = (X_j < s)$ et $R_2(j,s) = (X_j \geq s)$.
- idée : choisir en chaque noeud la coupe $(j,s)$ qui minimise (régression)

$$RSS = \sum_{i \in R_1(j,s)} (y_i - \widehat{y}_{R_1})^2 + \sum_{i \in R_2(j,s)} (y_i - \widehat{y}_{R_2})^2$$

variance inter-groupes

# Construction de l'arbre

Construction d'un ensemble de partitions de l'espace des variables explicatives par découpage dyadique récursif :

1. Choix d'une variable explicative, `Industry` et d'une valeur seuil (`Industry`=748)
2. Calcul des moyennes de la variable réponse de part et d'autre de la valeur seuil
3. Calcul d'une déviance (*RSS*)
4. Répéter pour les valeurs seuils possibles, chaque variable explicative et calculer les déviances associées : retenir le seuil qui minimise la déviance
5. Séparer les données (*split*) et recommencer sur chaque sous-ensemble jusqu'à ce que la déviance ne diminue plus (ou nombre minimal d'observations aux feuilles atteint).

# Règle d'arrêt

- On arrête les coupes si
  - la déviance ne décroit plus,
  - le nombre d'observations à un noeud est inférieur à une valeur prédéfinie,
  - la profondeur de l'arbre est supérieure à une valeur prédéfinie,
  - etc...
- Algorithme CART : élagage de l'arbre (*pruning*) au lieu des règles d'arrêt.

# Arbres de décision : éviter le sur-apprentissage

- L'arbre parfait n'existe pas en général
- une erreur faible sur les données d'apprentissage peut entraîner un faible pouvoir prédictif (sur-ajustement)
- Comment arrêter la croissance de l'arbre ?

Objectif : construire un arbre avec la plus faible erreur de prédiction possible.

En général (phénomène de *sur-apprentissage*)

- l'erreur d'apprentissage diminue à chaque étape
- l'erreur de prédiction (ou de généralisation) diminue, se stabilise puis augmente !

# Construction de l'arbre

Algorithme CART : un arbre se construit en 2 étapes

1. Construction d'un arbre maximal par découpes dyadiques récursives
   Les *splits* se font par seuillage d'une variable donnée.
2. Elagage : sélection d'un sous-arbre (pour éviter le *sur-apprentissage*)

La prédiction aux feuilles est la moyenne des observations (régression) ou le vote majoritaire (classification).

# Elagage (*pruning*)

## minimisation d'une mesure coût-complexité

Mesure de coût-complexité : à chaque valeur du paramètre $\alpha$ correspond un sous-arbre $T$ qui minimise

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \widehat{y}_{R_m})^2 + \alpha |T|$$

où

- $|T|$ est le nombre de feuilles de l'arbre,
- $R_m$ est la région correspondant à la $m$-ième feuille,
- $\widehat{y}_{R_m}$ est la valeur prédite à la $m$-ième feuille (moyenne des observations de $R_m$).

Le paramètre de complexité (CP) $\alpha > 0$ pénalise les grands arbres.

- si $alpha = 0$, $T$ est l'arbre maximal
- si $\alpha$ croit, les branches sont progressivement élaguées formant une suite de sous-arbres emboîtés

# Algorithme CART

1. Construction d'un arbre maximal
2. Pruning pour obtenir une suite de sous-arbres $\{T_\alpha\}$, $\alpha \nearrow$
3. Sélection de $\alpha$ par $K$-fold cross-validation par exemple
4. Retourner le sous-arbre de l'étape 2 correspondant à la valeur de $\alpha$ choisie.

# Arbres de classification

Un arbre de classification est similaire à un arbre de régression.
Ce qui change :

- En régression, la prédiction de $y$ est la moyenne des observations appartenant à la région $R_m$ de la feuille $m$ ; en classification, la classe prédite est la classe majoritaire parmi les observations appartenant à la région $R_m$ de la feuille $m$.
- le critère alternatif à $RSS$ est le taux d'erreurs de classification.
- Néanmoins, le taux d'erreurs de classification n'est pas toujours un critère assez sensible pour *construire* l'arbre (déterminer les *splits*) : on utilise plutôt l'indice de Gini ou le critère de cross-entropy.

# Indice de Gini

$Y \in \{0, 1\}$, $X$ covariable continue.

- $t$ split qui définit $A_1 = [-\infty, t]$ et $A_2 = [t, +-\infty[$ . Soit $\widehat{p}_s(j)$ la proportion d'observations dans $A_s$ telles que $Y_i = j$ :

$$\widehat{p}_s(j) = \frac{\sum_{i=1}^n I(Y_i = j, X_i \in A_s)}{\sum_{i=1}^n I(X_i \in A_s)}, \quad s = 1, 2, \ j = 0, 1.$$

- L'impureté du split $t$ est définie par

$$I(t) = \sum_s \gamma_s, \quad \gamma_s = 1 - \sum_j \widehat{p}_s(j)^2.$$

On choisit le split $t$ et la covariable $X$ qui minimise l'*indice de Gini*.

# Références


- *Classification and regression Trees*, Breiman, Friedman, Olshen, Stone (CART)
- *The elements of Statistical Learning*, Hastie, Tibshirani & Friedman
- *An introduction to Statistical Learning, with applications in R* James, Witten, Hastie & Tibshirani
- *Les forêts aléatoires avec R* Robin Genuer et Jean-Michel Poggi, 2019.


2 librairies R : `tree` et `rpart`

# Atouts et limites de CART

Atouts :

- Facile à lire et interpréter, fournit directement une classification
- Algorithme rapide
- Bien adapté aux vastes jeux de données
- Peu d'hypothèses sur les variables (méthode non-paramétrique)

Limites :

- Algorithme instable, sensible aux fluctuations d'échantillonage
- L'algorithme ne revient pas en arrière et ne remet pas en cause la hiérarchie induite.
- Utilisé seul, un arbre CART a des performances prédictives souvent inférieures à celles d'autres méthodes de classification ou de régression

Pour y remédier (diminuer la variance) :

- Bagging (Bootstrap aggregation),
- Forêts aléatoires (décorréler les arbres bootstrap),
- Boosting (apprentissage séquentiel de l'arbre).