CONTRIBUTEURS

ACT-3114 Apprentissage statistique en actuariat

aut. Alec James van Rassel
src. Marie-Pier Côté

1 Préparation de données

Repères visuels

1. La **position** sur une **même échelle** (variable numérique), souvent à l'aide de points ou de boîte à moustache ;
2. La **position** sur une **échelle identique**, mais **non alignée** (variable numérique) ;
   - Pour exemple, lorsqu'on utilise des facettes.
3. La **longueur** sur une **même échelle** (variable numérique) ;
   - Pour exemple, dans un diagramme à bande ou un histogramme.
4. L'angle ou **la pente** (variable numérique) ;
   - Pour exemple, en représentant les données sous forme de lignes (pente) ;
   - Pour exemple, la redoutable pointe à tarte (angle).
5. La **forme des points** (variable catégorielle) peut représenter le groupe ;
6. L'aire ou **le volume** (variable numérique) ;
   - Pour exemple, dans un diagramme à bande ou un histogramme ;
7. La **saturation de la couleur** (numérique ou catégorielle) peut représenter "à quel point" sur une échelle de claire à foncé ;
   - Pour exemple, gris vs noir.
8. La **teinte de couleur** (numérique ou catégorielle).
   - Pour exemple, bleu vs rouge.

Étapes du nettoyage de données

Cette liste n'est pas séquentielle, il est surtout important de *tout* le faire.

- ☐ **Comprendre** la structure des données ;
  - dimensions, types de variables, `str` et `summary`.
- ☐ **Visualiser** les données ;
  - `head`, `summary` et graphiques exploratoires.
- ☐ **Mettre en forme** ***(format)*** les données ;
  - Chaque ligne est une observation ;
  - Chaque colonne est une variable ;
  - Supprimer les doublons ;
- ☐ Vérifier et corriger les **types** de variables ;
  - booléens, entiers, numériques, facteurs, chaînes de caractères, dates, etc.
- ☐ **Manipuler** les **chaînes** de caractères ;
  - Corriger les typos ;
  - Changer la casse avec `tolower` ;
  - Extraire des informations avec les expressions régulières.
- ☐ Identifier les **données aberrantes** ;
  - Mettre `NA` et gérer plus tard.
- ☐ **Détecter** les **erreurs** flagrantes ou les changements structurels dans les données ;
  - Prendre compte des réformes ;
  - Constater les structures importantes.
- ☐ **Augmenter** les données à l'aide d'autres sources ;
  - optionnel.

Types de variables

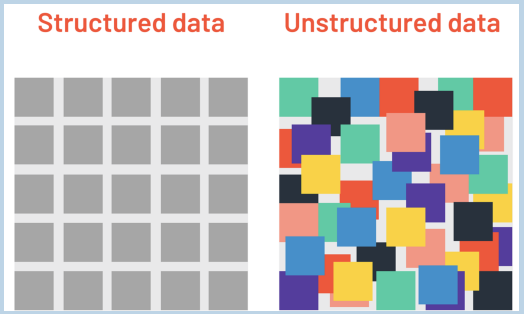
Les jeux de données peuvent être :

Structuré : Données ayant une structure prédéfinie.
Pour exemple :

- Tableaux ;
- « spreadsheet » ;
- « Relational databases ».

Non structuré : Données sans structure prédéfinie venant en toute forme et ne pouvant pas être facilement résumées à un tableau.
Pour exemple :

- Texte ;
- Images ;
- Audio ;



Types de variables :

Quantitatives : Données numériques pouvant être :

- Discrète, pour exemple des données de comptage ;
- Continue, pour exemple des montants de sinistre.

Qualitatives : Données catégorielles pouvant être :

- Nominales, pour exemple le sexe ;
- Ordinales, pour exemple des groupes d'âge.

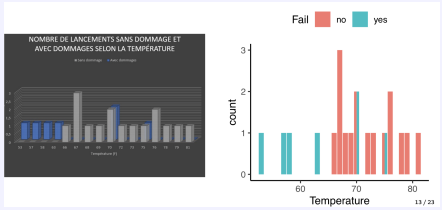
Temporelles Données chronologiques représentant un état dans le temps ;

- Pour exemple, la quantité totale de pluie au Canada le 1er juillet 1990.

Spatiales Données géographiques avec des coordonnées (pour exemple, la latitude et longitude).

Conseils

- Conserver un script séparé pour le traitement des données ;
- Documenter nos choix ;
  - Si l'on omet quelques observations en raison de données aberrantes, il faut être conscients que ça peut biaiser les résultats.
- Effacer l'encre superflue



- Considérer l'étique :
  - Avons-nous l'autorisation d'utiliser les données ?

    Pour exemple, les données ouvertes du gouvernement avec peu de restrictions ;
  - Est-ce qu'il y a des considérations éthiques à l'utilisation des données ?

    Pour exemple, Facebook qui utilise les données de façon immorale ;
  - Est-ce qu'il y a des biais dans les données pouvant causer préjudice ?

    Pour exemple, des données avec un biais sexiste ou raciste, des données financées par une compagnie.

Biais

d'échantillonnage Lorsque les données d'entraînement ne représentent pas de façon représentative la vraie population.
Pour exemple :

- Entraîner une voiture autonome seulement le jour alors qu'elles peuvent être conduites jour et soir ;
- Sonder les lecteurs du Devoir sur le parti pour lequel ils vont voter à l'élection et donc ignorer ceux ne lisant pas le journal.

de stéréotypes Données influencées par des stéréotypes (consciemment ou pas).
Pour exemple :

- Entraîner un algorithme pour comprendre comment les personnes travaillent avec des images d'hommes sur des ordinateurs et de femmes à la maison ;
- L'algorithme aura tendance à penser que les hommes sont des programmeurs et les femmes des cuisinières.

de mesure Données influencées par un problème avec l'instrument de mesure.
Pour exemple :

- Entraîner un algorithme de reconnaissance d'image avec des images d'une caméra avec un filtre de couleur ;
- L'algorithme serait fondé sur des images ayant systématique malreprésentée le vrai environnement.

de modèle Le compromis de biais-variance $B(\hat{\theta}) = \mathrm{E}[\hat{\theta}] - \theta$.
Pour exemple :

- Certaines variables ne sont pas considérées (ou mesurées) ;
- Le modèle n'est pas assez flexible (compromis linéaire vs lisse)

2 Données manquantes

Le chapitre utilise la **mise en contexte** suivante :

› Il y a une réclamation pour un accident d'auto en Ontario ;

› Le contrat d'assurance couvre les frais médicaux ;

› On désire calculer la probabilité de paiement (variable réponse) en fonction de :

1. La gravité de l'accident (variable explicative) ;

   3 niveaux : mineur-majeur-catastrophique ;

2. La souffrance du réclamant ;

   Échelle de 1 (peu) à 5 (beaucoup) ;

Problèmes de modélisation :

› Comment analyser les données malgré les valeurs manquantes ?

› Quels enjeux ou problèmes devrait-on considérer dans la modélisation ?

Terminologie

Notation

Y_{ij} : Valeur de la variable explicative j pour l'observation i où $j \in \{1,\dots,p\}$ et $i \in \{1,\dots,n\}$;

$\mathbf{Y}_{n\times p}$: Matrice contenant les données **complètes** ;

$\mathbf{Y}$ est partitionné en deux, $\mathbf{Y} = \{\mathbf{Y}_{obs}, \mathbf{Y}_{mis}\}$

$\mathbf{Y}_{obs}$: matrice avec les données ayant toutes les valeurs observées ;

$\mathbf{Y}_{mis}$: matrice avec les données comportant des valeurs manquantes ;

$\mathbf{R}_{n\times p}$: **Matrice de réponse** des variables indicatrices $R_{ij} = \mathbf{1}_{\{Y_{ij} \text{ observé}\}}$;

θ : **Paramètre de nuisance**

Exemple de notation

$$\boldsymbol{Y} = \begin{bmatrix} 2 & 3 \\ 8 & 6 \\ 3 & 12 \end{bmatrix}$$

$$\boldsymbol{Y}_{obs} = \begin{bmatrix} 2 & . \\ 8 & 6 \\ . & . \end{bmatrix} \qquad \boldsymbol{Y}_{mis} = \begin{bmatrix} . & 3 \\ . & . \\ 3 & 12 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$$

Mécanisme de non-réponse

La distribution de **R** est le *mécanisme de non-réponse* ;
Types de données manquantes :

1. **MCAR** : Missing Completely at Random ;
   › Le patron de non-réponse (pattern of missing values) est indépendant des données **Y** ;
   › Il s'ensuit que la probabilité de réponse $f(R|\mathbf{Y},\theta)$ ne dépend pas des données complètes **Y** :
   $$f(R|\mathbf{Y},\theta) = f(R|\theta)$$

Exemple avec un θ de 10%

On perd 10% des valeurs mesurées alors, $\forall i \in \{1,\dots,n\}, j \in \{1,\dots,p\}$, la distribution du mécanisme de non-réponse : $R_{ij} \sim \text{Bernoulli}(p = 1-\theta = 90\%)$

› **Tester** la différence de moyennes :

$\mathcal{H}_0 : \{p_{\text{Cat, mis}} - p_{\text{Cat, obs}} = 0\}$ et $\{p_{\text{Maj, mis}} - p_{\text{Maj, obs}} = 0\}$

Est équivalent à tester :

$\mathcal{H}_0$: les données sont MCAR

avec un test du khi-carré de Pearson ;

C'est-à-dire que le test est équivalent, mais **pas** le hypothèses.

2. **MAR** : Missing at Random ;
   Can think of it as Missing *Conditionally* at Random ;
   › La probabilité de réponse $f(R|\boldsymbol{Y}1616,\theta)$ dépend seulement des variables qui ont été observées dans le jeu de données $\mathbf{Y}_{\text{obs}}$:
   $$f(R|\mathbf{Y},\theta) = f(R|\mathbf{Y}_{\text{obs}},\theta)$$
   › Exemple de patients d'un hôpital : les données sont MAR lorsque la probabilité de non-réponse ne dépend pas de la qualité de vie sachant l'âge ;
   › Le négatif est qu'il est impossible de tester que, sachant l'âge, la probabilité de non-réponse ne dépend pas de la qualité de vie ;
   › Il est **inconcevable** d'avoir un test pour MAR.
3. **NMAR** : Not Missing at Random ;
   › Le patron de non-réponse pour **Y** est relié à sa valeur et les variables observées ;
   Ce même si on conditionne sur les valeurs observées ;
   › La probabilité de réponse $f(R|\mathbf{Y},\theta)$ dépend également de $\mathbf{Y}_{\text{mis}}$ et ne peut pas être simplifiée ;
   › Pour exemple, les patients malades ne répondent pas aux sondages en plus des patients plus jeunes et donc la probabilité de réponse dépend de la qualité de vie ;
   › Pour exemple, la probabilité de réponse dépend d'une autre variable non observée ;

Visuellement on peut comparer les 3 patrons de non-réponse :

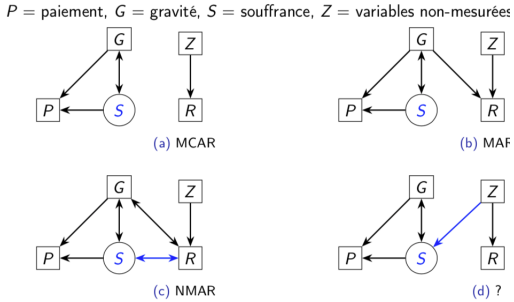
› On observe (MCAR) que la variable *indépendante* S ne dépend pas du patron de non-réponse R ;

› On observe (MAR) que G influe la variable réponse S et le patron de non-réponse R ;

› On observe (NMAR) qu'il y a un lien direct entre S et R ;

› En dernier, puisque Z n'est pas mesuré et que Y y dépend c'est NMAR.

Principe d'inclusion : Si une variable est exclue, cela peut créer une corrélation entre la variable indépendante S et le patron de non-réponse R. De plus, ça peut changer le patron lui-même !



Pour exemple, si on cherchait à supprimer des valeurs d'une BD :

MCAR supprime des valeurs aléatoirement ;

MAR supprime 60% des valeurs pour les femmes et 40 % des valeurs pour les hommes ;

NMAR plus il y a de sinistres observés, plus il y a de chances que les valeurs soient manquantes.

Ordre de restriction des différents patrons

1. MCAR : plus « restrictif », car les données doivent être manquantes complètement aléatoirement ;

   Même idée que l'hypothèse de normalité est restrictive puisque les données *doivent* l'être, **mais** cela nous permet d'utiliser plein de tests ;

   Comme une Bernoulli, il n'y a pas de paramètres ;

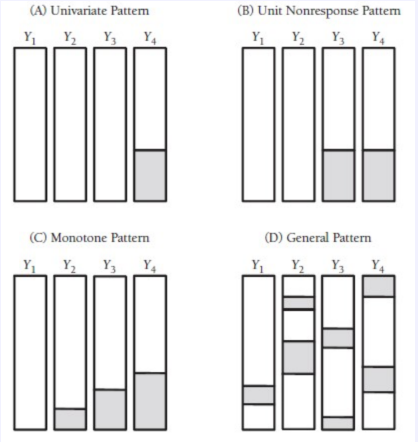
   plus restrictif alias moins flexible.
2. MAR : inclue les variables observées et donc il y a plus de paramètres ;
3. NMAR : inclue toutes les variables et est donc le patron le plus flexible (alias, le moins restrictif).

Visualisation et détection

Traiter les données manquantes

1. Détectez, visualisez et documentez les données manquantes ;
2. Identifier le patron de non-réponse ;

   Pour exemple, voici quelques patrons de non-réponse pour quelques variables :

   Exemple de patron univarié : concessionnaire ne pose jamais à ses assurés s'ils ont une piscine et donc la variable est manquante dans toutes ses données ;
3. Comparer les distributions des autres variables selon la valeur des variables indicatrices $R_{1j}, \ldots, R_{nj}$;

Identification des types de non-réponses

- Pour les variables continues, on fait un **test** t **sur les différences de moyenne** au lieu du khi-carré de Pearson comme pour MCAR ;
- Problème de comparaisons multiples ;
- Le test MCAR de Little est peu utile, mais peut adresser le problème de comparaisons multiple avec une hypothèse testant toutes les variables ;

Traitement des données manquantes

En continuant la mise en contexte, on suppose qu'on veut estimer le vecteur **fi** des coefficients de la régression logistique pour prédire la probabilité de paiement; Une question valide est si les options pour le traitement des données manquantes dépendent du **type** de non-réponse
Options de traitement :

1. Utiliser seulement les **cas complets** *(complete-case analysis)*;
   - L'option par défaut pour les fonctions :
     `lm, glm, na.rm, na.omit`
   - **Impact** :
     - ↓ taille de l'échantillon
     - ↓ taille de la corrélation des variables
     - ↑ variance des estimateurs
     - ↓ puissance des tests
   - Uniquement valide sous MCAR;
2. Utiliser seulement les **cas disponibles** *(available-case analysis)*;
   - Utilise uniquement les données observées pour l'analyse;
   - Rarement applicable;
   - ↓ la taille de l'échantillon **moins** qu'en utilisant d'uniquement les cas complets;
   - Sans biais **uniquement** sous MCAR;
3. **Imputation** simple par la **moyenne ou** la **médiane**
   - Substitue les `NA` par la moyenne ou médiane de la variable;
   - **Impact** :
     - ↓ variabilité de la variable
     - ↓ corrélation de la variable avec les autres
   - Même sous MCAR, les données sont **sévèrement** « distorted »;
4. Imputation simple par une régression;
   - Substitue les `NA` par la prévision d'une régression de la variable sur les autres avec les cas complets;
   - Si plusieurs variables ont des données manquantes, leurs patrons doivent être traités séparément;
   - L'inter **corrélation** des variables est **conservée**, mais est **surestimée** (même si MCAR);
   - La variance est **sous-estimée**, mais **moins** qu'avec l'imputation par la moyenne;
5. Imputation stochastique par une régression;
   - Ajoute un terme d'erreur ε (normalement distribué) à la prévision de la régression;
   - *Si plusieurs variables sont manquantes dans un patron, les erreurs sont corrélées*
   - Corrige les biais pour la méthode d'imputation par la régression (sous-estimation de la variance et surestimation de l'inter corrélation des variables);
   - La variance des paramètres est **sous-estimée**, *sauf si on en tient compte dans les calculs*;
   - Fonctions `R` utiles du paquetage `mice` :
     `mice.impute.norm.nob(), mice.impute.norm()`
6. Imputation simple *hot-deck*;
   - Substitue les valeurs `NA` d'une observation par les valeurs observées d'une autre observation choisie aléatoirement;;
   - Habituellement, cette observation fait parmi d'un sous-ensemble d'observations *proches* (pensez au *K*-NN, clustering, etc.);
   - Souvent utilisée pour les sondages;
   - N'altère par les distributions univariées
   - ↓ l'inter corrélation des variables;
   - Biais des estimations des coefficients **fi** de régression;
7. Imputation **multiple**;
   - Répète l'imputation stochastique et agrège les résultats;
   - Ce faisant, la variabilité additionnelle dût à l'imputation des valeurs manquante est adressée et la variance des estimateurs est *non biaisée*;

Autres méthodes :

- MLE avec données manquantes;
- Algorithme EM (expectation-maximisation)
- Inférence bayésienne;

Conseils

- Conserver un script pour le traitement de données manquantes et ne **pas hard-coder**;
- *Utiliser une méthode d'imputation qui respecte le format de la variable;*
- Plus la proportion de non-réponses est élevée, plus l'impact sur l'analyse sera important;
- S'il y a plusieurs patrons de non-réponse différents, l'ordre dans lequel les données sont imputées est important;

3 Analyse en composantes principales

Fonctions R

```
PCA(X = , scale = T)
```

› On donne la BD en premier argument et standardise avec le deuxième.

4 Apprentissage non supervisée

Liaisons

Complete linkage Mesure la distance maximale (les deux points les plus éloignés) des groupes et on choisit la plus petite distance.

Single linkage Mesure la distance minimale (les deux points les plus proches) des groupes et on choisit la plus petite distance.

Average linkage Mesure la moyenne des distances entre les points de chacun des groupes puis on choisit la plus petite distance.

Centroid linkage Assigne un centroïd à chaque groupe ayant comme coordonnées la moyenne des observations du groupe.

Ward La distance correspond à l'augmentation de l'EQM suite à la fusion d'un groupe. L'EQM commence à zéro puisque chaque observation à son propre groupe. Ceci est utile lorsque l'on croit que l'EQM devrait être petite.

Concepts de distance et similarité

Classification

supervisée Lorsqu'on connaît les **étiquettes** des groupes et on veut prédire l'appartenance à un groupe pour de nouvelles observations ;

› « *Classification* ».

non supervisée Lorsqu'on ne sait pas combien de groupes il y a ni comment qu'ils sont formés ;

› Alias partitionnement ou regroupement ;

› « *Clustering* ».

Conditions des mesures de distance

Une mesure de distance dans l'espace $\mathbb{E}$ est une application $d : E \times E \to \mathbb{R}^+$ respectant ces 3 conditions :

1. Symétrique
   $d(i,j) = d(j,i), \forall i,j \in E$;
2. Nulle pour un même objet
   $d(i,j) = 0 \Leftrightarrow i = j, \forall i,j \in E$;
3. Satisfait l'inégalité du triangle
   $d(i,k) \leq d(i,j) + d(j,k), \forall i,j,k \in E$.

Distance de Minkowski ℓ_q

Soit 2 points $(x_{11}, \ldots, x_{1d})$ et $(x_{21}, \ldots, x_{2d})$ dans l'espace $\mathbb{R}^d$.
Alors, pour $q \geq 1$:

$$\begin{aligned} \ell_q &= \|\mathbf{x_1} - \mathbf{x_2}\|_q \\ &= \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^q \right)^{1/q} \end{aligned}$$

Deux cas particuliers sont bien connus :

› $q = 1$: Distance de Manhattan ℓ_1 ;

› $q = 2$: Distance euclidienne ℓ_2.

La version standardisée de la distance euclidienne se simplifie à :

$$d(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{\sum_{i=1}^{d} \left(\frac{x_{1i} - x_{2i}}{s_i} \right)^2}$$

Conditions des indices de similarité

Une mesure de similarité entre 2 objets dans l'espace $\mathbb{E}$ est une application $s : E \times E \to [0,1]$ respectant ces 2 conditions :

1. Symétrique
   $s(i,j) = s(j,i), \forall i,j \in E$;
2. Un même objet est identique et maximise la similarité
   $s(i,j) = 1 \geq s(i,j), \forall i,j \in E$.

De plus, l'indice de similarité peut être obtenu d'une mesure de distance $s(i,j) = \frac{1}{1+d(i,j)}$.

Également, on peut définir un indice de **dissimilarité** avec $\tilde{d}(i,j) = 1 - s(i,j)$.

Types de variables binaires

symétrique Si une variable binaire est **symétrique**, c'est que ses deux niveaux sont aussi fréquents ;

› Par exemple, qu'une personne soit un homme ou une femme.

asymétrique Si une variable binaire est **asymétrique**, c'est qu'un niveau est plus rare que l'autre.

› On dénote le niveau plus rare par 1 ;

› Par exemple, deux personnes ayant brisé leur orteil sont plus semblables que deux personnes ayant un mal de tête.

Mesures de similarité (variables binaires ou ordinales)

Variable binaire symétrique

La **proportion d'accord** (« *simple matching coeffi-*

cient ») :

$$s(\mathbf{x_1} - \mathbf{x_2}) = \frac{1}{d}\sum_{i=1}^{d} \mathbf{1}_{\{x_{1i}=x_{2i}\}}$$

Variable binaire asymétrique

L'indice de Jaccard :

$$s(\mathbf{x_1} - \mathbf{x_2}) = \frac{\sum_{i=1}^{d} x_{1i}x_{2i}}{\sum_{i=1}^{d}\{1-(1-x_{1i})(1-x_{2i})\}}$$

Variable binaire asymétrique

On assigne un score numérique (positif) à chaque niveau et traite la variable comme une variable numérique.

Similarité de Gower

Dans le cas où les variables sont de plusieurs types :

$$G(\mathbf{x_1} - \mathbf{x_2}) = \frac{\sum_{i=1}^{d} w_i\gamma_i(x_{1i}, x_{2i})s_i(x_{1i}, x_{2i})}{\sum_{i=1}^{d} w_i\gamma_i(x_{1i}, x_{2i})}$$

où le poids des variables $w_i > 0$.
Également, avec l'étendu dénoté $r_i = \max(x_{1i}, \ldots, x_{ni}) - \min(x_{1i}, \ldots, x_{ni})$, si $\mathbf{x_i}$ est :

- numérique ou ordinale, $\gamma_i(x_{1i}, x_{2i}) = 1$ et $s_i(x_{1i}, x_{2i}) = 1 - \frac{|x_{1i}-x_{2i}|}{r_i}$;
- binaire symétrique, $\gamma_i(x_{1i}, x_{2i}) = 1$ et $s_i(x_{1i}, x_{2i}) = 1_{\{x_{1i}=x_{2i}\}}$;
- binaire asymétrique, $\gamma_i(x_{1i}, x_{2i}) = 1 - (1 - x_{1i})(1 - x_{2i})$ et $s_i(x_{1i}, x_{2i}) = 1_{\{x_{1i}=x_{2i}\}}$.

Similarité du Cosinus

Un exemple d'application de la similarité du Cosinus est l'analyse de texte. On assigne une variable indicatrice pour chaque mot selon s'il est présent ou non (il y a donc *beaucoup* de colonnes). Par la suite, on assigne un poids w_i à chaque variable selon son importance. Donc, par exemple, la mesure peut calculer la similarité entre deux textes.

$$s_i(x_{1i}, x_{2i}) = \frac{\sum_{i=1}^{d} w_i x_{1i}x_{2i}}{\sqrt{\sum_{i=1}^{d} w_i x_{1i}^2 \sum_{i=1}^{d} w_i x_{2i}^2}}$$

K-means clustering

Notation

- Soit les observations $\mathbf{x_1}, \ldots, \mathbf{x_n}$ formées de variables **quantitatives** ;
- Soit les sous-ensembles $C_1, \ldots, C_k$;
- Chaque observation appartient seulement à un groupe et donc les ensembles sont **mutuellement exclusifs**

$$C_j \cap C_{j'} = \emptyset, \forall j \neq j', j, j' \in \{1, \ldots, k\}$$

Hierarchical clustering

Fonctions R

`hclust(d = dist(data), method = "complete").`

- On spécifie la méthode de liaison avec `method =`;
- On donne en argument une matrice de distance pour `d = .`

On peut élaguer avec `cutree(tree = , k = , h = ))`.

- On donne en premier argument le dendrogramme ;
- On peut soit spécifier un nombre de groupes avec `k =` ou une hauteur avec `h = .`

Paramètres

Un paramètre est appris par l'entraînement.
Un hyperparamètre informe l'entraînement comme tel et est fixé à l'avance.
Par exemple, k est un hyperparamètre dans le K-NN, et c'est le seul.

Échantillonage

Pour atténuer le problème de débalancement :

sur-échantillonnage dupliquer au hasard des observations sous observées.

sous-échantillonnage ignorer une partie des observations ayant l'étiquette plus fréquemment rencontrée.

D'une manière ou l'autre, le jeu de données étudié est alors plus balancé.

5 Apprentissage supervisé

Inférence

Comprendre le lien entre les prédicteurs et la *variable réponse*. Par exemple, tests d'hypothèses, intervalles de confiance et interprétations.

De façon générale, on préfère des modèles moins flexible puisqu'ils sont plus interprétables pour faire de l'inférence.

Prévisions

Prédire le plus précisément possible la valeur de Y étant donné les valeurs des covariables.

Si $\hat{f}$ peut estimer f, une prévision pour la variable d'intérêt Y étant donné les valeurs de $x_1, \ldots, x_p$ est $\hat{Y} = \hat{f}(x_1, \ldots, x_p)$.

Il y a deux types de techniques qui servent à estimer f :

1. L'apprentissage automatique et
2. La modélisation statistique.

Apprentissage automatique

But principal Prédire la valeur de Y ;

Y On ne cherche pas à explicitement estimer ni interpréter l'effet de la valeur des variables x_j sur Y ;

Fluctuation aléatoire N'est **pas** un intérêt majeur ;

hypothèses à priori sur f Peu d'hypothèses à priori, on se base sur les données.

Modélisation statistique

But principal Estimer explicitement et interpréter l'effet de la valeur des variables x_j sur Y ;

Y Peut prédire Y avec le modèle ;

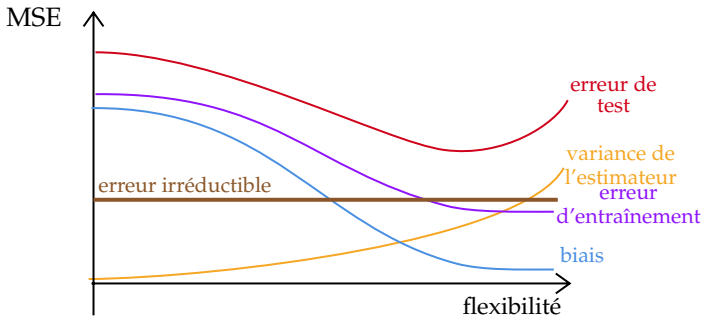
Fluctuation aléatoire Est un intérêt puisqu'on veut parfois simuler de nouvelles observations de Y ;

hypothèses à priori sur f Sont requises afin d'estimer seulement ce qui est inconnu de f (c.-à-d., les paramètres).

Flexible vs inflexible :

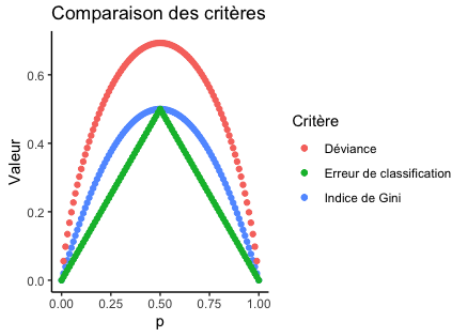
- With alot of data and few predictors, a flexible approach would perform better. The amount of data permits us to fit more accurate predictors and the variance is limited by the number off parameters.
- With a small number of observations and many predictors, an inflexible approach would perform better. The small number of observations leads to worse parameter estimates and the combined large number of predictors leads to the possiblity of overfitting.
- A flexible approach would fit better when there is a non-linear relationship between the predictors and the response variable. An inflexible is more restrained and can't capture it.
- A flexible approach would fit worse when there is high variance amongst the error terms. The flexible model would try to fit this error and drastically increase variance.

Décomposition Biais-Variance



6 Arbres de classification et de régression

CART Classification and Regression Tree.



Indice de Gini

Mesure de la variance totale entre les K classes :

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

› L'indice de Gini mesure la **pureté des nœuds**, si les proportions échantillonnales sont près de 0 ou 1, G sera très petit.

› C'est la mesure par défaut avec la fonction pour une classification `rpart(method = "class")`.

Déviance, ou entropie croisée

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

Élagage par coût-complexité

cp Paramètre de complexité.

|T| Nombre de feuilles (« *terminal nodes* »).

Critère de coût-complexité

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

› Le critère permet d'ajuster le modèle en balancant la **complexité** de l'arbre ***et*** la **qualité** de l'ajustement.

Le seul point négatif des CART est leur précision qui a tendance à être mauvaise.

Note Le point de séparation doit être un qui parmi l'ensemble donné. Par exemple, pour $\{1, 3\}$ on ne peut pas stipuler un point de séparation à 2.

Fonction R

Paquetage `rpart` (Recursive Partioning And Regression Trees) dont la fonction `rpart()`.

› On utilise `weights =` si on veut pondérer des observations.

› On utilise `cost =` pour spécifier un vecteur de coûts associés à chaque variable dans la séparation des données.
Cela peut s'avérer utile pour pénaliser une variable qu'on voudrait idéalement utiliser seulement si son pouvoir prédictif est très grand.

› `method =` permet de spécifier si l'arbre doit retourner une réponse de type "class" ou "numeric".

› `control = list(maxdepth = )` permet de spécifier la profondeur de l'arbre d.

Élagage : `cv.tree(tree, FUN = )`

› Pour un arbre de régression, on utilise `FUN = prune.tree;`

› Pour un arbre de classification, on utilise `FUN = prune.misclass`.

7 Bagging et forêts aléatoires

En raison du nombre faible d'observations dans plusieurs des nœuds :

› La variance des prévisions est très élevée.

› L'arbre n'est pas robuste—des différents échantillons mènent à des points de séparation différents et donc des prévisions différentes.

Pour aider à atténuer ce problème, nous voyons deux **méthodes d'ensemble**.

Méthode d'ensemble

Algorithme de prévision qui agrège les prévisions de plusieurs modèles différents.

› Cela est particulièrement utile si les modèles ont une **grande variance** et donc est rarement utilisé avec des GLMs.

Bagging

Méthode d'ensemble générale qui permet de réduire la variance d'un modèle d'apprentissage statistique.

› Bien que le bagging peut être appliqué à n'importe quel modèle, on le voit uniquement dans le contexte d'arbres ;

› Prendre la moyenne des prévision obtenues avec différents échantillons d'entrainement permet de réduire la variance des prévisions ;

› Il s'ensuit que les prévisions sont améliorées :

$$\mathrm{E}\left[\begin{array}{c}\text{erreur quadratique}\\ \text{de prévision}\end{array}\right] = biais^2 + \text{variance} + \varepsilon$$

Algorithme de bagging sur les arbres

Pour $b = 1, 2, \ldots, B$,

1. Générer un **échantillon bootstrap** $\mathcal{D}^b$ de l'échantillon d'entrainement ;
   › L'échantillon est de la même taille que l'échantillon d'entrainement initial.
2. Ajuster un (gros) arbre de régression sur $\mathcal{D}^b$ pour obtenir la prévision $f^b(\boldsymbol{x})$.
   › Il n'y a pas d'élagage avec $cp = 0$;
   › Les arbres ont une grande variance et un faible biais ;
   › La taille de l'arbre est contrôlée avec `maxdepth` (nombre maximal de niveaux) et `minbucket` (taille minimale d'un nœud).

$$f_{\text{bag}}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} f^b(\boldsymbol{x})$$

Note Le nombre d'arbres B doit être assez gros (100, 1 000, etc.) et **une grande valeur ne mène pas à un sur ajustement** *habituellement* ; c'est les autres paramètres qui vont contrôler le sur ajustement.

Prévision

› La probabilité de ne pas piger une observation parmi n, avec remise, est $1 - \frac{1}{n}$.
La probabilité qu'elle ne soit pas dans l'échantillon de taille n du tout est $(1 - \frac{1}{n})^n$.
Lorsque $n \to \infty$, $\lim_{n\to\infty}(1 - \frac{1}{n})^n = \mathrm{e}^{-1} = 36.79\% \approx 1/3$.
On dit donc que, en moyenne, chaque arbre dans le bagging n'utilise pas environ $1/3$ des données ;

› Les observations qui ne sont pas utilisées sont les observations dites "Out-Of-Bag" (OOB) ;

› Pour chaque itération du bagging, on calcule la prévision pour les observations OOB.
Pour chaque observation i, on calcule la moyenne des prévision pour en obtenir une seule ;

› L'EQM des ces prévisions calculée à la fin est **l'erreur OOB**.
Lorsque B est suffisamment grand, elle se stabilise pour être équivalente à l'erreur obtenue avec LOOCV.

Limitations

\+ Les prévisions sont beaucoup plus "lisses" (voir p. 24) ;

– Les prévisions sont beaucoup moins interprétables.

  – Notamment, avec plus de 2 variables, on ne peut pas visualiser les prévisions et donc on ne peut pas comprendre pourquoi l'algorithme fait ses décisions.

Forêts aléatoires

Méthode d'ensemble qui s'inspire du bagging mais qui permet de **décorréler les arbres** en :

› On choisit un échantillon bootstrap d'une plus petite taille `sampsize`, par exemple 50% ou 75% de la taille de l'échantillon d'entrainement ;

  – On note que logiquement le plus il y a de prédicteurs alors le plus les arbres seront corrélés entre eux (voir p. 31 des NDC).

› À chacun des nœuds de chacune des itérations dans la construction de l'arbre, on choisit aléatoirement m prédicteurs à considérer pour la séparation.

  – La valeur de $m \in \{1, \ldots, d\}$ est un hyperparamètre mais on choisit souvent $m \approx \lfloor d/3 \rfloor$ en régression et $m \approx \sqrt{d}$ en classification.

Les améliorations apportées par la forêt sont plus utiles lorsqu'ils y a beaucoup de prédicteurs corrélés dans les données. La corrélation sera encore plus importante s'il y a certaines variables qui très importantes puisque tous les arbres vont l'avoir ce qui augmente la corrélation.

Mesure de l'importance des variables

Permutation On substitue aléatoirement la variable j pour une observation et effectue une prévision avec cette substitution qui n'a aucun rapport avec le restant de la ligne.

- Si la nouvelle prévision est mauvaise alors la variable est importante, et vice-versa ;
- En `R` cette méthode s'appelle « *mean decrease in accuracy* ».

Diminution totale dans la fonction de perte due à une séparation sur la variable j, et on compare pour $j = 1, 2, \ldots, d$.

- Pour chaque séparation qui a lieu sur la variable j, je vais calculer la diminution de la fonction de perte (donc à tous les nœuds de tous les arbres) ;
- Souvent on divise par la valeur maximale pour avoir des importances entre 0 et 1 ;
- En `R` cette méthode s'appelle « *mean decrease in node impurity* ».

On peut les visualiser en `R` avec `varImpPlot`.
La mise en garde des arbres est que le modèle est beaucoup moins interprétable et beaucoup plus lourd à calculer.

8 Boosting et Gradient Boosting

Terminologie

λ Taux d'apprentissage (alias, le paramètre de régularisation).

› Le diminuer mène toujours à une meilleur performance, au prix d'un plus grand nombre d'arbres T.

› En pratique, on choisit la plus petite valeur de λ qui permet un temps de calcul raissonable.

T Nombre d'itérations.

› Normalement assez grand mais peut mener à un surajustement.

d Profondeur des arbres à chaque itération.

› Souvent, $d = 1$ pour des souches ce qui mène à un modèle additif ;

› Plus d est élevé, plus les interactions d'ordre élevé peuvent être captées mais il est rare que $d > 7$ soit utile.

δ Pourcentage de sous-échantillonnage.

› Souvent 50% ou 75%.

Boosting

Le boosting permet de combiner plusieurs modèles faibles en un prédicteur puissant.

› À chaque itération, un arbre tente d'améliorer ce que le modèle ne prédit pas bien.

› Le boosting est itératif et donc, en contraste aux forêts aléatoires, l'ajustement des arbres est impacté par l'ordre.

Algorithme de boosting (arbres de régression)

On initialise :

$$\hat{f}_0(\boldsymbol{x}) = 0 \quad \text{et} \quad \rho_{i,0} = y_i, \forall i = 1, \dots, n$$

Pour chaque itération de $t = 1, \dots, T$,

1. Ajuster un arbre de régression de profondeur d sur les résidus $(\boldsymbol{x}, \rho_t)$.
   › Donc, on contrôle le nombre maximal de feuilles avec 2^d feuilles ;
   › Par exemple, si $d = 1$ c'est une souche avec 2 feuilles ;
   › La fonction de prévision est notée $\hat{f}^t_{\text{arbre}}$.
2. Mettre à jour $\hat{f}$ en ajoutant une version rapetissée de l'arbre :
   $$\hat{f}_t(\boldsymbol{x}) = \hat{f}_{t-1}(\boldsymbol{x}) + \lambda \hat{f}^t_{\text{arbre}}(\boldsymbol{x})$$
   › Donc on ne prend pas directement la prévision de l'arbre $\hat{f}^t_{\text{arbre}}$ mais plutôt une fraction de la prévision selon le taux d'apprentissage λ.
3. Mettre à jour les résidus :
   $$\rho_{i,t} = \rho_{i,t-1} - \lambda \hat{f}^t_{\text{arbre}}(\boldsymbol{x}_i)$$

La prévision du boosting pour un arbre de régression est :

$$f_{\text{boost}}(\boldsymbol{x}) = \hat{f}_T(\boldsymbol{x}) = \sum_{t=1}^{T} \lambda f^t_{\text{arbre}}(\boldsymbol{x})$$

Adaptive Boosting (Adaboost)

Puisque les résidus n'ont pas de sens en classification, on défini la méthode Adaboost.

Au lieu d'avoir un même taux d'apprentissage pour tous les arbres, chacun des arbres aura une contribution différente dans la prévision finale.

Algorithme Adaboost (arbres de classification)

On initialise les poids des observations :

$$w_i = \frac{1}{n}, \ \forall i = 1, \dots, n$$

Pour chaque itération de $t = 1, \dots, T$,

1. Ajuster un arbre de classification $\hat{f}_t(\boldsymbol{x})$ de profondeur d sur les données d'entraînement en utilisant l'indice de Gini pondéré par les poids w_i.
2. Calculer le taux d'erreur :
   $$\text{err}_t = \frac{\sum_{i=1}^{n} w_i \times \mathbf{1}_{\{y_i \neq \hat{f}_t(x_i)\}}}{\sum_{i=1}^{n} w_i} = \frac{\text{la somme des poids pour les observations mal classées}}{\text{la somme totale des poids}}$$
3. Calculer la contribution de l'arbre :
   $$\alpha_t = \log\left(\frac{1 - \text{err}_t}{\text{err}_t}\right)$$
4. Réajuster, alias ***ada***pter, les poids $\forall i = 1, \dots, n$,
   $$w_i \leftarrow w_i \exp\left\{\alpha_i \mathbf{1}_{\{y_i \neq \hat{f}_t(x_i)\}}\right\}$$

Le classificateur est :

$$\hat{f}_{\text{ada}}(\boldsymbol{x}) = \text{signe}\left\{\sum_{T}^{t=1} \alpha_t \hat{f}_t(\boldsymbol{x})\right\}$$

› On traite les prévisions $\hat{f}_t$ comme étant soit -1 ou 1 représentant les classes et la classe la plus présente sera la prédiction.

Fonctions R

`boosting(boos = F, coeflearn = "Freund")` du paquetage `adabag`.
`Adaboost.M1` de `caret`.

- `coeflearn` est la formula pour α.
- `boos = F` utilise l'indice de Gini pondéré.
- `boos = T` (utilisé par StatQuest) utilise un algorithme qui est comme un boosting pondéré.

Boosting de Gradient

Peut généraliser l'idée du boosting pour toute fonction de perte $\mathcal{L}$ d'intérêt (e.g., la déviance pour des lois Bernoulli, Poisson ou Tweedie).

L'idée est qu'à l'itération t, on essaie d'améliorer le plus possible (de façon « *greedy* ») la prévision en trouvant l'arbre $\hat{f}^t_{\text{arbre}}$ qui minimise :

$$\sum_{i=1}^{n} \mathcal{L}\{y_i, \hat{f}_{t-1}(\boldsymbol{x}_i) + \hat{f}^t_{\text{arbre}}(\boldsymbol{x}_i)\}$$

- Cette fonction de perte pourrait être l'EQM, la déviance, etc. ;
- Les deux arguments sont la valeur observée et la prévision à l'itération t.

Descente la plus à pic

L'idée est qu'à l'itération t, on fait un pas dans la direction du gradient négatif menant à une baisse de la valeur de la fonction de perte.
Le gradient pour l'itération t est :

$$\left[\frac{\partial \mathcal{L}\{y_i, f(\boldsymbol{x}_u)\}}{\partial f(\boldsymbol{x}_i)}\right]_{f=\hat{f}_{t-1}}$$

- En anglais, « *steepest descent* » ;
- Le gradient est seulement défini sur les observations est donc on trouve un arbre dont les prévisions sont les plus proches possible du gradient négatif, qu'on surnomme pseudo-résidus, en utilisant l'EQM ;
- On ajoute une erreur aléatoire et sélectionne un nouveau sous-échantillon de taille δn pour ajuster l'arbre afin d'obtenir le boosting de gradient **stochastique**

Algorithme de Gradient boosting (arbres de régression)

On initialise :

$$\hat{f}_0(\boldsymbol{x}) = \arg\min_a \sum_{i=1}^{n} \mathcal{L}(y_i, a)$$

- Donc on trouve la constante a qui minimise la fonction de perte ;
- Pour une déviance d'une poisson ou l'EQM, ce serait la moyenne des observations $\bar{y}$;
- Dans la majorité des cas, a est facile à trouver.

Pour chaque itération $t = 1, \ldots, T$,

1. Échantillonner aléatoirement sans remise δn observations notées $\mathcal{D}^t$.
2. Calculer le pseudo-résidu pour chaque observation i dans l'échantillon $\mathcal{D}^t$:
$$\rho_{i,t} = -\left[\frac{\partial}{\partial f(\boldsymbol{x}_i)} \mathcal{L}\{y_i, f(\boldsymbol{x}_i)\}\right]_{f=\hat{f}_{t-1}}$$
3. Ajuster un arbre de régression (perte EQM peu importe la fonction de perte $\mathcal{L}$) de profondeur d aux pseudo-résidus $\rho_{i,t}$ pour obtenir les régions (alias les feuilles) $\mathcal{R}_{j,t}, \forall j = 1, \ldots, J_t$.
   - Donc, souvent $J_t = 2^d$ mais peut être un peu plus petit ;
   - Donc cette étape sert à trouver les points de coupure.
4. Pour $j = 1, \ldots, J_t$, trouver $\hat{a}_{j,t} = \arg\min_a \sum_{i:\boldsymbol{x}_i \in \mathcal{R}_{j,t}} \mathcal{L}\{y_i, \hat{f}_{t-1}(\boldsymbol{x}_i) + a\}$;
5. Mettre à jour $\hat{f}$ en ajoutant une version rapetissée de l'arbre :
$$\rho_{i,t} = \rho_{i,t-1} - \lambda \hat{f}^t_{\text{arbre}}(\boldsymbol{x}_i)$$
6. Mettre à jour :
$$\hat{f}_t(\boldsymbol{x}) = \hat{f}_{t-1}(\boldsymbol{x}) + \lambda \sum_{j=1}^{J_t} \hat{a}_{j,t} \times \mathbf{1}_{\{x \in \mathbb{R}_{j,t}\}}$$

La prévision est :

$$f_{\text{gbm}}(\boldsymbol{x}) = \hat{f}_T(\boldsymbol{x})$$

Fonctions R

Paquetage `gbm` ou `method = "gbm"` dans le paquetage `caret` pour régler les hyperparamètres suivants :

- Nombre d'itérations dans le boosting `n.trees`;
- Profondeur maximal de l'arbre `interaction.depth`;
- Le taux d'apprentissage λ `shrinkage`;
- Nombre d'observations minimal dans un nœud `n.minobsinnode`;

XGBoost

- Traite les valeurs manquante correctement si elles sont MAR;
- Comportement des prévisions peut être inattendu parfois;
- Pour la plupart des problèmes, la méthode boosting par gradient stochastique est la plus précise et la plus complexe.

9 Communication et interprétations

Les **graphiques de dépendance partielle (PDP)** nous permettent de mieux comprendre l'effet *global* d'une variable explicative sur la prévision.

- Peut être utilisé avec n'importe quel modèle de prévision (Boosting, Arbres, etc.)

On évalue la prévision pour une variable d'intérêt x_ℓ avec $\ell \in \{1, \dots, p\}$ et on prend la moyenne sur les valeurs possibles des autres variables :

$$\bar{f}_\ell(x_\ell) = \frac{1}{n}\sum_{i=1}^{n} f_{\text{model}}(x_\ell, \boldsymbol{x}_i^{(-\ell)})$$

- Le vecteur $\boldsymbol{x}_i^{(-\ell)}$ contient les valeurs observées de toutes les variables explicatives pour l'observation i sauf x_ℓ.

Les **graphiques d'espérance conditionnelle individuelle (ICE)** nous permettent de comprendre l'effet d'une variable explicative sur *une prévision en particulier* et de *détecter des interactions*.