Module 4 - Méthodes de régression

Dans ce module, nous allons aborder certaines méthodes de régressions. Nous supposons donc que nous avons une variable quantitative y que nous allons essayer d'expliquer/de prédire avec une ou des variables explicatives x. Vous pouvez créer le fichier R `Module4.R`.

À avoir en tête : la distinction entre **expliquer** et **prédire** est essentielle dans l'utilisation des modèles de régression/de machine learning. Les deux objectifs utilisent souvent des techniques similaires et ils peuvent parfois être complémentaires, mais leurs finalités et implications sont différentes.

- Expliquer = comprendre les relations causales, construire des modèles qui s'appuient sur la régularité des phénomènes sous-jacents → importance des hypothèses et des tests statistiques pour évaluer la significativité des variables, privilégie les modèles simples et interprétables
- Prédire = anticiper des données futures ou prévoir des données non observées, généraliser → importance de la performance, de la généralisation, privilégie les modèles complexes et performants mais moins interprétables

Définir sa problématique dès le début permettra de guider les choix à réaliser tout au long de l'analyse.

N'hésitez pas à visiter le site *spurious correlations* :
`https://www.tylervigen.com/spurious-correlations`

1 La régression linéaire simple

C'est le cas le plus simple. On essaye d'expliquer ou prédire Y avec une variable X et une relation linéaire de la forme :

$$Y = \beta_0 + \beta_1 X + \epsilon$$

où ϵ est le bruit ou erreur de mesure. β_0 (l'ordonnée à l'origine) et β_1 (la pente) sont inconnus, le but est de les estimer. Pour cela on utilise la méthode des moindres carrés ordinaires, c'est-à-dire qu'on minimise la somme des carrés des écarts entre la variable à expliquer et la droite de régression :

$$\text{Min}_{\beta_0,\beta_1} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

Les estimateurs obtenus sont notés $\hat{\beta}_0$ et $\hat{\beta}_1$.

La prédiction $\hat{Y}_i$ est simplement $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$, et les résidus estimés $\hat{\epsilon}_i = Y_i - \hat{Y}_i$.
L'analyse des résidus studentisés (c'est-à-dire de même variance) est intéressante pour vérifier l'ajustement individuel du modèle (présence de points aberrants ?) et l'ajustement global (y-a-t-il une structure dans les données ?).

Pour présenter les régressions linéaires, on va utiliser le jeu de données `Advertising` qui permet d'obtenir le nombre de ventes en fonction de la publicité sur 3 canaux différents : la télévision, la radio et les journaux.

```
rm(list=ls())
gc()

df<-read.csv("Data/Advertising.csv", header = T, row.names = 1)

summary(df)
```

On peut commencer à faire des statistiques descriptives sur les corrélations entre les différentes variables, par exemple avec les commandes :

```
pairs(df)

cor(df)
```

Pour réaliser une régression linéaire, on utilise la commande `lm`. On utilisera la commande `summary` pour avoir des détails sur les résultats de cette régression linéaire.

```
rl<-lm(df$Sales ~ df$TV)
# équivalent : rl <- lm(Sales ~ TV, data = df)

summary(rl)
```

On obtient : l'estimation du coefficient, son écart-type estimé, la valeur de la statistique du test d'hypothèse $H_0 : \beta_i = 0$ contre $H_1 : \beta_i \neq 0$ et la probabilité pour la statistique de test sous H_0

de dépasser la valeur estimée.

Dans notre cas, l'hypothèse nulle est rejetée pour β_0 et pour β_1 : la constante doit apparaître dans le modèle et la liaison entre les dépenses en publicité TV et les ventes est positive et significative (pour 100 euro de publicité supplémentaire à la TV, les ventes augmentent d'environ 5 euros).

La valeur du R^2 indique la part de la variabilité de Y expliquée par notre modèle (ici, par une seule variable X).

$$R^2 = \frac{\hat{Var}(\hat{Y})}{\hat{Var}(Y)}$$

Le R^2 est compris entre 0 et 1 : plus il est proche de 1, plus notre modèle a un pouvoir explicatif important.

On peut effectuer le code suivant (avec ou sans la librairie `ggplot2`) pour faire le graphe des ventes sur les dépenses en publicité TV et la droite de régression associée.

```
# En R base
plot(df$Sales ~ df$TV)
abline(rl, col="red")

# Avec ggplot et geom_smooth()
ggplot(df) + aes(x=TV,y=Sales) + geom_point() +
geom_smooth(method="lm") + theme_light()

# Avec ggplot et les prédictions calculées à la main
sales_pred<-predict(rl)
ggplot(df) +
    aes(x=TV,y=Sales) +
    geom_point() +
    geom_line(aes(y=sales_pred)) +
    theme_light()
```

Les commandes suivantes permettent de représenter les résidus studentisés. En théorie 95% des résidus studentisés se trouvent dans l'intervalle $[-2;2]$, ce qui est bien le cas ici.

```
residus <- rstudent(rl)
plot(residus, pch=15, cex=.5, ylab="Résidus", ylim=c(-3,3))
abline(h=c(-2,0,2), lty=c(2,1,2))
```

À noter que d'autres diagnostics sur les résidus sont disponibles en appliquant la fonction `plot()` sur l'objet de la régression.

2 Régression linéaire multiple

La régression linéaire multiple est utilisée lorsque nous avons plusieurs variables explicatives quantitatives à notre disposition. C'est une généralisation du modèle de régression linéaire simple.
Par exemple :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

Avec X le vecteur colonne contenant les variables X_1, X_2 et X_3 on peut écrire :

$$Y = X'\beta + \epsilon$$

Avec R, ce sera avec la même commande lm mais en ajoutant des $+$ pour les variables explicatives comme ci-dessous. Une alternative est de mettre un point (.) pour indiquer à R que toutes les variables du jeu de données (sauf Y) sont explicatives.

```
rl<-lm(Sales ~ TV + Newspaper + Radio, data = df)
rl<-lm(Sales ~ ., data = df)

summary(rl)
```

La régression linéaire multiple permet de solutionner (en partie) le potentiel problème du **biais de variable omise**. On peut sur(sous)-estimer l'effet d'une variable sur une autre lorsque l'on ne contrôle pas pour d'autres variables. Par exemple, lorsque l'on fait une régression simple des ventes sur les dépenses de publicité dans les journaux, on trouve un coefficient positif et significatif :

```
rl_news<-lm(df$Sales ~ df$Newspaper)

summary(rl_news)
```

Alors que lorsque l'on fait la régression en rajoutant les variables des dépenses de publicité TV et radio, le coefficient n'est plus significatif !

```
rl_tot<-lm(df$Sales ~ df$Newspaper + df$TV + df$Radio)

summary(rl_tot)
```

L'idée sous-jacente ici que les dépenses de publicité dans les journaux sont corrélées à celles à la TV et à la radio, mais ces dernières expliquent davantage la variabilité des ventes.

Pour autant, si nous ne cherchons pas une interprétation causale aux liens entre les variables et que nous effectuons des tâches de prédictions, le biais de variable omise n'est pas forcément important.

3 Et avec des variables explicatives qualitatives ?

Pour expliquer une variable Y par des variables X qualitatives, il est possible de faire des régressions multiples en incluant les indicatrices correspondant à chaque modalité de X. Par exemple si X peut valoir 0, 1 ou 2, on peut effectuer la régression suivante :

$$Y = \beta_0 + \beta_1 \mathbb{1}_{\{X=1\}} + \beta_2 \mathbb{1}_{\{X=2\}} + \epsilon$$

Dans ce cas, $X = 0$ est considérée comme la situation de *référence*, et les coefficients β_1 et β_2 s'interpèteront relativement à cette référence.

Il est aussi possible d'adopter le formalisme de l'ANOVA que nous n'abordons pas ici mais pour lequel vous pouvez vous reporter aux ressources pédagogiques suivantes :

- Le site internet `https://www.datanovia.com/en/fr/lessons/anova-dans-r/`
- La section 9.3 du livre R pour la statistique et la science des données : `https://r-stat-sc-donnees.github.io/`

4 Régression logistique

Lorsque Y, la variable que l'on cherche à expliquer est qualitative, on peut utiliser la méthode de la régression logistique. Nous présentons ici la méthode pour une variable Y valant 0 ou 1 avec des variables explicatives X.
Dans ce modèle on essaye d'expliquer/de prédire la probabilité que Y = 1 sachant les variables explicatives X en utilisant la modélisation suivante :

$$P(Y = 1|X = x) = \frac{exp(X'\beta)}{1 + exp(X'\beta)}$$

Les coefficients β sont estimés par la méthode du maximum de vraisemblance.

Pour regarder empiriquement ce que cela donne, nous allons utiliser un jeu de données où l'on essaye d'expliquer/prédire la réaction d'un client suite à une campagne marketing (souscription ou non).

```
souscr <- read.table("data/Souscription.csv",
                     header=TRUE,
                     sep=";")

str(souscr)
```

Nous commençons par transformer la variable Y d'intérêt codée oui/non (si le client a souscrit ou non au produit) en variable numérique binaire 0/1 pour ensuite pouvoir appliquer le modèle logistique.

```
library(dplyr)

souscr<-mutate(souscr,y=case_when(y=="no" ~ 0, TRUE ~ 1))

str(souscr)

summary(souscr)
```

Pour effectuer la régression logistique, nous allons utiliser la commande `glm` qui regroupe les méthodes de modélisation linéaires généralisées. Il faut spécifier une famille de loi de probabilités : loi binomiale pour une régression logistique.

```
logit_complet<-glm(y~.,data=souscr,family=binomial)

summary(logit_complet)
```

Ici on constate que plusieurs variables sont non significatives (loanyes, dayofweek...). On pourrait donc retirer ces variables du modèle pour gagner en **parcimonie**, lisibilité, interprétabilité.

Plusieurs procédures et critères permettent de sélectionner les variables pertinentes pour un modèle. Ici nous utilisons la commande `step()` qui est basée sur la minimisation du critère AIC (Akaïke Information Criterion). La procédure de sélection est effectuée pas à pas : à chaque étape, la variable dont le retrait du modèle conduit à la diminution la plus grande du critère AIC est enlevée ; le processus s'arrête lorsque le retrait d'aucune variable ne permet de diminuer le critère. Nous relançons ensuite notre régression logistique avec le modèle apuré avec 8 variables (au lieu de 20 initialement).

```
final.lm = step(logit_complet)

logit_step=glm(y ~ contact + month + duration + campaign + poutcome
+ emp.var.rate + cons.price.idx + cons.conf.idx, data=souscr,
family=binomial)

summary(logit_step)
```

Nous pouvons enfin regarder le pouvoir explicatif de notre modèle en comparant les prédictions et les vraies valeurs de Y.

```
# Prédiction des probabilités :
prev<-predict(logit_step, type="response")

# Quand la proba est > 0.5 alors y = 1 ; sinon y = 0
prev<-as.numeric(prev>0.5)

# Tableau de fréquence des prédictions 0/1
table(prev)
```

```r
# Taux de bonnes prédictions
mean(prev==souscr$y)

table(prev,souscr$y)
```

Dans 101 cas notre modèle prédit 1 à tort, dans 248 cas notre modèle prédit 0 à tort.

Le critère selon lequel évaluer la performance du modèle dépend de la problématique métier. Ici par exemple, il y a très peu de clients ayant souscrits au produit, les Y = 1 sont rares. En prédisant très fréquemment Y = 0, on peut donc facilement obtenir des bonnes performances d'un certain point de vue ; pour autant, l'intérêt pour le métier est nul puisqu'il s'agit justement d'arriver à prédire les cas où le client va souscrire au produit...

De manière générale, il est souvent compliqué de faire un bon modèle de prédiction sur les jeux de données avec des modalités de Y rares. Dans ce cas une solution peut-être de sélectionner un échantillon aléatoire avec les modalités non rares de Y et la (quasi-)totalité des modalités rares de Y afin de rééquilibrer le jeu de données, estimer un modèle et répéter plusieurs fois cette opération.

5 Arbre de décision

Le principe des arbres est de scinder lespace en deux de manière récursive. Les arbres sont des outils permettant d'explorer, d'expliquer ou encore de classer des données. Dans cette section nous abordons la méthode CART (Classification And Regression Trees) qui est une des méthodes les plus populaires. Cette méthode vise à construire un arbre binaire qui construit des classes homogènes d'individus par rapport à la variable d'intérêt. L'arbre construit est composé de « noeuds » et de « feuilles », ce qui rend sa lecture très intuitive. Nous utiliserons le même jeu de données que pour la régression logistique, et la librairie `rpart` (autre package possible : `tree`).

```r
rm(list=ls())
library(rpart)

souscr <- read.table("data/Souscription.csv",header=TRUE,sep=";")

arbre <- rpart(y~.,data=souscr,cp=0.02)
```

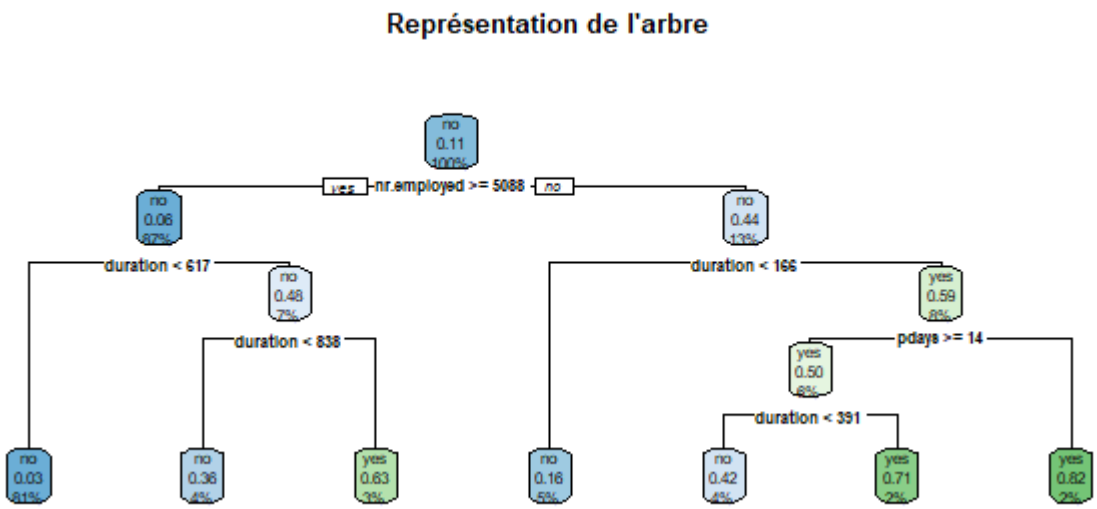
```
arbre
```

L'option `cp` permet de fixer le paramètre de complexité de notre arbre. Plus `cp` est petit, plus notre arbre sera complexe, c'est-à-dire profond/large. Bien que l'objectif soit de produire des groupes les plus homogènes possibles, on ne veut pas avoir un arbre trop profond (avec beaucoup de feuilles) car il risque d'avoir une mauvaise capacité de généralisation à de nouveaux individus : on parle de surapprentissage.
L'approche classique consiste à construire un arbre maximal puis à l'élaguer : pour cela, la fonction `printcp()` renvoie les caractéristiques d'une suite d'arbres construits avec différents paramètres de complexité. On choisit généralement celui qui minimise l'erreur de classification.

La librairie `rpart.plot` permet de résumer l'information de l'arbre et de le représenter graphiquement :

```
install.packages("rpart.plot")
library(rpart.plot)

rpart.plot(arbre,main="Représentation de l'arbre")
```



Dans cet arbre, il y a des feuilles pures (toutes les observations ont un label identique, 0 ou 1) et d'autres non pures (mélange des deux labels). Pour les feuilles pures, la règle de classification est simple : on affecte la classe à l'unique label observé dans la feuille ; dans le cas contraire, on peut décider d'affecter la classe la plus représentée dans la feuille. Par exemple dans la feuille

en bas à gauche de l'arbre, on a 81% de l'échantillon et 3% d'entre eux pour qui $Y = 1 \rightarrow$ la règle est donc : si nr.employed > 5088 & duration < 628 alors $Y = 0$.

Regardons les prédictions et la performance de notre arbre :

```
# Prédictions sous forme de probabilités
pred_arbre<-predict(arbre,data=souscr)

# Prédictions sous forme de variable binaire
pred_arbre_cl<-predict(arbre,data=souscr,type="class")

# Comparaison avec les "vraies" valeurs de Y
table(pred_arbre_cl, souscr$y)

# Taux de bonnes prédictions
mean(pred_arbre_cl==souscr$y)
```

Remarque : un des défauts des arbres est qu'ils sont assez instables : ils peuvent changer rapidement si on ajoute une observation ou si on modifie les paramètres du modèle. Le principe des forêts aléatoires permet de contourner ce problème en agrégeant un nombre important d'arbres.

6 Exercice - Module 4

Rappel : les exercices sont à faire en Quarto (ou Rmarkdown) et doivent être rendu dans le même fichier. Vous devez donc compléter votre fichier `Exercice_R_NOM_PRENOM.qmd`.

Partie 1

Pour cette première partie, on utilisera le jeu de données "Ozone.txt" où l'on cherche à expliquer la variable maxO3 (la concentration en ozone) en fonction de diverses variables (températures, nébulosité, vent, pluie).

1. Charger les données "Ozone.txt" et effectuer quelques statistiques descriptives et graphiques pour explorer les données. Commenter.
2. Étudier la relation entre la variable `maxO3` et `T15` (la température à 15h). Effectuer une régression linéaire simple pour expliquer la concentration en ozone en fonction de cette

température. Commenter les coefficients et le R^2. Réaliser le graphe représentant le nuage de points (T15, maxO3) et la régression linéaire.

3. On veut à présent effectuer une régression linéaire multiple. Sélectionner des variables en justifiant votre choix et effectuer une telle régression en commentant les résultats.

Partie 2

Pour cette seconde partie, on utilisera le jeu de données `AdClick.csv` où l'on cherche à expliquer la variable `click` qui vaut 1 si l'utilisateur a cliqué sur une publicité (0 sinon) à partir de variables explicatives (le genre, l'âge et le salaire estimé).

1. Charger les données `AdClick.csv` et effectuer quelques statistiques descriptives et graphiques pour explorer les données. Commenter.
2. Effectuer une régression logistique pour expliquer les clics. Commenter les résultats et commenter les erreurs de classifications grâce à une table croisant le clic prédit et le clic effectif.
3. Construisez un arbre de classification, représenter-le et commenter les résultats de la même manière.