Polycopié de la ressources *R6.02 - Méthodes statistiques pour le Big Data* dans le cadre de la formation *SD* Grenoble.

Département SD IUT2 de Grenoble

Cette ressource s'appuie principalement sur les ressources et SAÈ suivantes :

- R1.03 - Bases de la programmation 1
- R1.04 - Statistique Descriptive 1 : Polycopié disponible à l'adresse https://membres-ljk.imag.fr/Vincent.Brault/Cours/Cours_1A.pdf
- R2.01 - Reporting et Datavisualisation
- R2.03 - Bases de la programmation 2
- R2.04 - Programmation statistique
- R2.05 - Statistique descriptive 2
- R2.06 - Probabilités 2
- R2.08 - Statistique inférentielle
- SAÉ2.02 - Estimation par échantillonnage
- SAÉ2.06 - Analyse de données, reporting et datavisualisation
- R3.06 - Tests d'hypothèses pour l'analyse bi-variée
- SAÉ 3.03 - Description et prévision de données temporelles
- R4.03 - Classification automatique
- R4.02 - Méthodes factorielles
- R4.EMS.08 - Modèle linéaire
- SAÉ4.EMS.01 - Expliquer ou prédire une variable quantitative à partir de plusieurs facteurs
- R5.02 - Data mining
- R5.EMS.06 - Modélisation statistique avancée
- R5.VCOD.08 - Renforcement informatique 2
- SAÉ5.03 - Mise en oeuvre d'un processus de Datamining
- R6.01 - Big Data : enjeux, stockage et extraction

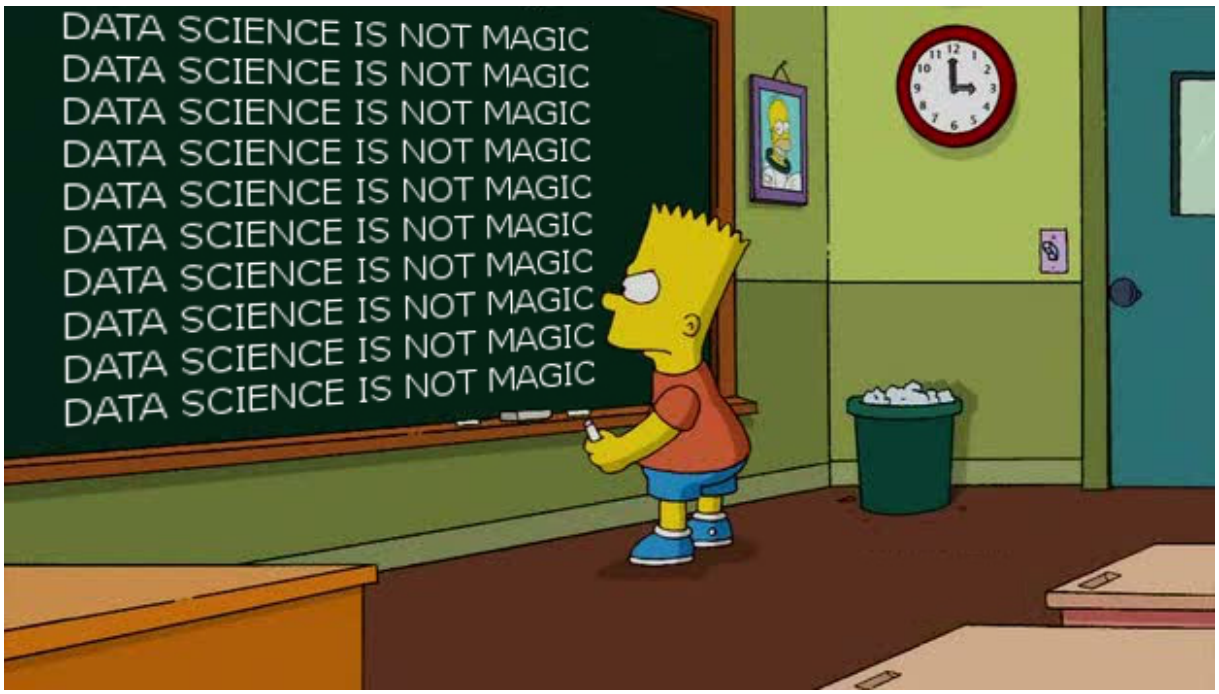


FIGURE 1 – Petit rappel que nous ne pouvons pas faire de la magie avec le *Big Data*

Table des matières

Avant propos

Ce cours généralise les concepts enseignés jusque là au cas du *Big Data*. Il se repose donc sur une grande partie des ressources vues jusque là. En plus des ressources du BUT mises sur la page de garde, ce polycopié s'appuie sur les références suivantes :

- *Analyse de données avec R* de Husson et al. (2016).
- *Python pour les SHS. Introduction à la programmation pour le traitement de données* de Schultz et Bussonnier (2021).
- *Wikistats* de Besse (2022)

Nous vous encourageons à les consulter si vous voulez plus de précisions ou une vision différente des notions présentées.

De plus, je remercie les lectrices et les lecteurs attentif·ve·s pour les corrections de coquilles. À ce titre, merci donc à Kylian Deschamps, Jérémy Macé et Célia Tropel (BUT3 - 2023/2024).

Planning

- Cours/TD 1 : Introduction
- TP 1 : Les limites des méthodes statistiques en grande dimension

---

- Cours/TD2 : Volume et début de la vélocité
- TP 2 : Volume et vélocité (1/2)

- Cours/TD3 : Fin de la vélocité et variété
- Cours/TD4 : Véracité
- TP 3 : Volume et vélocité (2/2)

- Cours/TD5 : Valeurs et visualisations
- TP 4 : Données manquantes
- TP 5 : LASSO

Chapitre 1

Introduction

"Le Big Data, c'est comme le sexe chez les adolescents : tout le monde en parle, personne ne sait vraiment comment faire, tout le monde pense que tout le monde le fait, donc tout le monde prétend le faire."

Dan Ariely [1] en 2013.

"Les seuls qui en ont vraiment fait n'en parlent pas car cela ne s'est généralement pas très bien passé."

Anonyme.

Dans ce chapitre, nous expliquons un l'origine du mot *Big Data* et du vocabulaire associé avant de présenter les problèmes que cela implique pour les analyses statistiques.

1.1 Origine du vocabulaire

Avant de parler de *Big Data*, revenons sur le terme *Informatique Décisionnelle* ou *Business Intelligence* (*BI*) notamment avec la définition proposée par Futura Science [2].

Définition 1 (Informatique Décisionnelle ou *Business Intelligence* (*BI*))

L'**informatique décisionnelle** (ou **Business Intelligence** en anglais), souvent raccourci par **BI**, désigne un ensemble de méthodes, de moyens et d'outils informatiques utilisés pour piloter une entreprise et aider à la prise de décision : tableaux de bord, rapports analytiques et prospectifs.

Sur la figure 1.1, nous proposons un petit rappel historique de quelques évènements clefs.

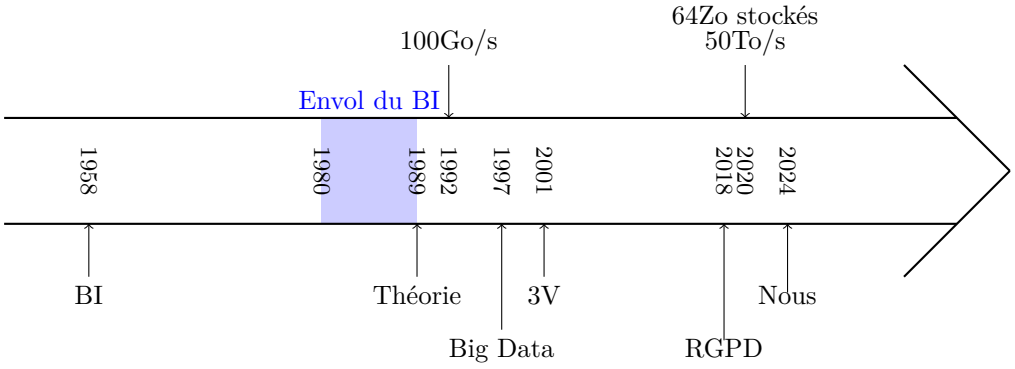


FIGURE 1.1 – Chronologie de quelques moments clefs de l'évolution des termes pour amener au *Big Data*. Pour rappel, 1Go = 10^9octets, 1To = 10^{12}octets et 1Zo = 10^{21}octets.

Le *BI* a été théorisé pour la première en 1958 par Hans Peter Luhn, un informaticien allemand [3], mais n'a connu un essor que dans la décennie des années 1980. Dans les années 1990, la taille des données

1. professeur de psychologie et d'économie comportementale israélo-américain qui enseignait à l'Université Duke
2. Source : https://www.futura-sciences.com/tech/definitions/informatique-informatique-decisionnelle-15057/
3. Source : https://www.salesforce.com/fr/blog/2016/06/une-breve-histoire-de-la-business-intelligence.html

collectées grossissant, le terme *Big Data* fut utilisé pour la première dans des articles scientifiques en 1997 [4] puis, en 2001, le *Meta Group* évoqua pour la première fois les *3V* dont nous parlerons plus tard.

Devant l'étendue du flot de données collectées (50To par seconde en 2020) et stockées (64Zo en 2020 [5]), la commission européenne décida de légiférer sur l'utilisation de ces données avec le *Règlement Général sur la Protection des Données* ou *RGPD* qui encadre, depuis 2018, la collecte et l'utilisation des données [6].

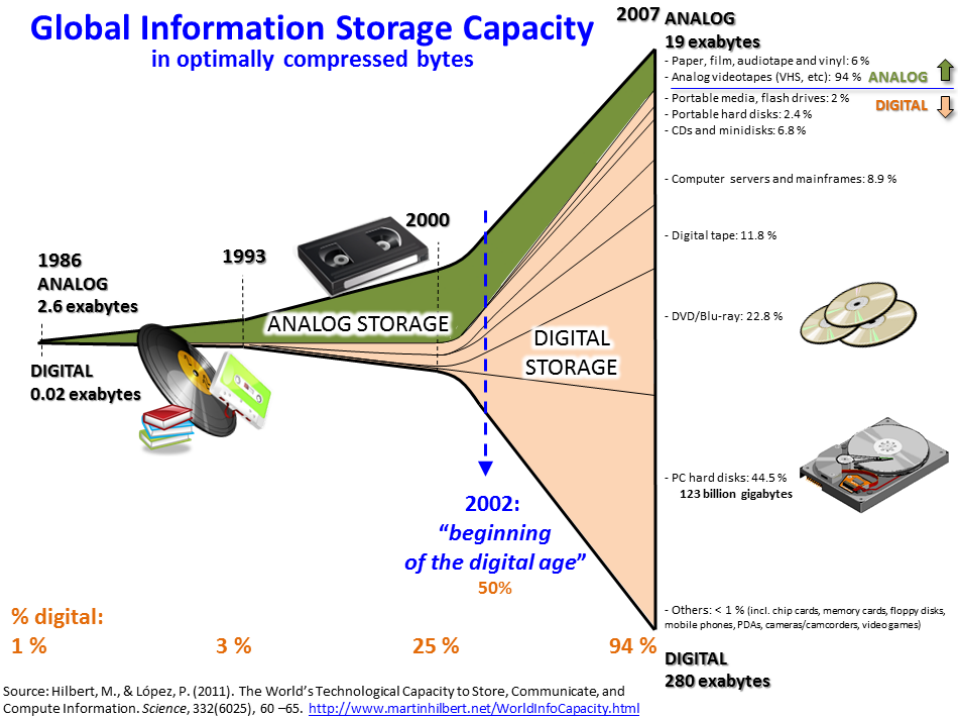
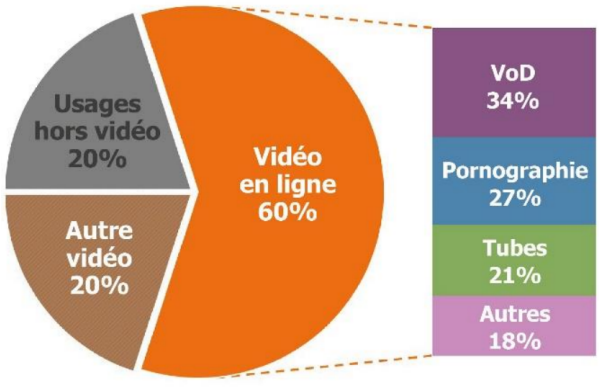


FIGURE 1.2 – Evolution des systèmes de stockage entre l'analogique et le numérique entre 1986 et 2007.

Dans leur article de 2011, Hilbert et López (2011) montrent que le stockage était essentiellement analogique (cassette, disque, livre...) jusqu'en 2002 où il est devenu surtout numérique (voir la figure 1.2).



Répartition des flux de données en ligne
entre les différents usages en 2018 dans le monde
[Source : *The Shift Project 2019* - à partir de (Sandvine 2018), (Cisco 2018) et (SimilarWeb 2019)]

FIGURE 1.3 – Estimation de l'utilisation du flux des données mondiales sur internet en 2019.
Source : Project et Efoui-Hess (2019)

Le rapport de Project et Efoui-Hess (2019) se base notamment sur Sandvine (2016); Cisco (2018) et

4. Source : http://www.archimag.com/univers-data/2015/11/26/infographie-big-data-mots-chiffres
5. Source : https://fr.statista.com/infographie/17800/big-data-evolution-volume-donnees-numeriques-genere-dans-le-monde/
6. Source : https://www.consilium.europa.eu/fr/policies/data-protection/data-protection-regulation/

tente d'estimer l'utilisation du flux mondial de données. D'après celui-ci, 80% du flux internet en 2018 était dû à l'utilisation de vidéos avec 60% pour les vidéos en lignes (voir la figure 1.3). Parmi ces 60%, ils ont estimé que l'utilisation était majoritairement pour les Vidéos à la demande (34%) suivies par la pornographie (27%) puis par les vidéos en streaming (par exemple Youtube) pour 21% et enfin d'autres vidéos comme celles mises sur TikTok, Facebook, Twitter...

Cette grande quantité de données circulant a permis de repenser certaines utilisations de la statistique et du *machine learning*

1.2 Big Data et Data science

Comme expliqué dans la section introductive 1.1, la notion de *Big Data* est récente en comparaison de celles de la statistique et même de l'informatique. De plus, sa définition varie légèrement suivant les points de vue. Dans ce cours, nous inclurons sous le terme *Big Data* deux types de combinaisons données/objectifs :

- les données extrêmement vastes et/ou à croissance rapide ;
- les données nécessitant un traitement extrêmement rapide.

Une autre façon de le voir est que nous chercherons à comprendre ce qu'il se passe lorsque la combinaison entre le type de données et l'objectif *dépasse la capacité des méthodes de traitement traditionnelles*. Il sera donc intéressant de prendre des cas classiques et de voir où sont les limites dans les procédures usuelles.

Devant cette nécessité de se renouveler dans l'analyse de données, des équipes ont grossi notamment sur LinkedIn dans laquelle travaillait Dhanurjay Patil et, pour cela, il fallait définir un nom englobant des profils variés. Dans un interview [7], Dhanurjay Patil explique qu'il a réfléchi avec Jeff Hammerbacher (à l'époque chez Facebook) pour trouver ce terme, je cite

> Alors, on s'est dit : analyste, ça fait trop Wall Street ; statisticien, ça agace les économistes ; chercheur scientifique, ça fait trop académique. Pourquoi pas "data scientist" ?

Concrètement, il est assez compliqué de définir un *data scientist* car les offres de postes avec ce terme peuvent avoir des missions très variées. Dans ce cours, nous resterons sur la définition attribuée à Josh Wills (Cloudera) :

> Personne qui est meilleure en statistique que n'importe quel·le ingénieur en informatique et meilleure en logiciel que n'importe quel·le statisticien·ne.

1.3 Les 6V

En 2001, Laney et al. (2001) présentaient les *3V* du *Big Data* qui furent ensuite accompagnés de trois autres. Cette notion revient régulièrement quand il s'agit de *Big Data* (voir par exemple Mayer-Schönberger et Cukier (2013)) et nous articulons le cours autour de ces notions que nous rappelons donc :

- **Volume** : la première notion présente dans le *Big Data*, comme son nom l'indique, est que la taille des données peut être *gigantesque*. Notons que cette notion est subjective et va dépendre de l'utilisation des données. Nous pouvons notamment inclure dans ce point le stockage et comment organiser les données pour un meilleur stockage ou encore le problème d'avoir des données situées à plusieurs endroits.
- **Vélocité** : à l'opposé, il est question de la vélocité. Avec les systèmes embarqués, par exemple, il est nécessaire de traiter beaucoup d'informations rapidement. Dans ce cas, il va falloir réfléchir à optimiser les procédures pour obtenir des résultats en un temps raisonnable suivant les objectifs.

7. Source : https://www.nouvelobs.com/tech/20170411.OBS7885/dj-patil-ex-sorcier-des-chiffres-d-obama-l-ethique-devrait-fai
html

- **Variété** : le grand nombre de données va aussi avec une plus grande variété. Jusqu'à présent, les méthodes statistiques étudiées durant votre BUT s'appliquaient à un type particulier de données. Par exemple, nous avons l'analyse en composantes principales (ou ACP) pour les données quantitatives et l'analyse factorielle des correspondances (ou AFC) pour les variables qualitatives. La question de la variété est de trouver des solutions pour pouvoir analyser les deux en même temps.

- **Véracité** : lorsque nous avons des données, une des premières questions à se poser (cf la ressource *R1.04 - Statistique Descriptive 1*) est la véracité des réponses. Avec une petite base, il est possible de regarder les valeurs pour voir des incohérences. Mais lorsque la base de données est trop grande, il est nécessaire de trouver des stratégies pour détecter les erreurs et/ou réussir à limiter leur influence. Le problème des données manquantes va également se poser.

- **Valeur** : un des dangers abordés dans ce point est la question de l'importance des variables dans une décision. Cette réflexion peut se faire avant le recueil ou l'analyse pour ne pas récupérer des données trop personnelles inutiles par exemple ou a posteriori quand on cherche les variables les plus pertinentes.

- **Visualisation** : visualiser les données est intéressant pour voir les corrélations entre deux variables, la proximité de deux individus ou d'autres notions. Si elle est facile en deux dimensions, elle devient compliquée en trois et facilement biaisée à partir de quatre. Du coup, l'un des enjeux est de trouver des solutions pour explorer la base et mettre en évidence les résultats lorsque la dimension est encore plus grande.

Ces notions seront également abordées dans le ressource *R6.01 - Big Data : enjeux, stockage et extraction.*

Remarque

Bien que nous ayons proposé une description pour chacun des 6V, il est important de garder en tête que ces notions sont imbriquées. Par exemple, un stockage sera plus facile si les types des variables sont identiques.

1.4 Quelques exemples de bases de données

En 2017, pour internet, nous estimions que [8]

- 72h de vidéo étaient ajoutées sur Youtube chaque minute ;
- environ 144,8 milliards de courriels étaient échangés quotidiennement ;
- 822 240 sites web étaient créés chaque jour ;
- 4 milliards d'actions avaient lieu sur Facebook de façon journalière.

Dans d'autres registres, nous pouvons aussi noter :

- ***Netflix*** : En 2009, la société *Netflix* remettait un prix d'une valeur de 1 000 000 de dollars à l'équipe *BellKor's Pragmatic Chaos* pour avoir su prédire 10,06% des valeurs manquantes de sa matrice composée de 100 480 507 cases (480 189 abonnés pour 17 770 films) ; voir par exemple Bennett et Lanning (2007). Au 31 décembre 2015, *Netflix* comptait 70 millions d'abonnés dans le monde pour une offre de 200 000 films et séries rien que pour la France (soient 1.4×10^{13} cases).

- **ADN** : Le génome d'un être vivant est une suite de petites molécules appelées **bases** permettant de mieux comprendre l'ADN (acide désoxyribonucléique). Pour la bactérie *Escherichia coli*, on recense $4,6 \times 10^6$ bases et $3,4 \times 10^9$ pour l'homme.

- **Optique** : En mai 2016, l'un des portables les plus performants faisait des photos avec $2,6 \times 10^7$ pixels et $1,1 \times 10^7$ pour les films (un film d'une minute contiendra en tout $1,584 \times 10^{10}$ pixels).

8. Source : *Blog du modérateur* http://www.blogdumoderateur.com/infobesite-quand-la-quantite-prime-sur-la-qualite-du-contenu/.

- **Transport** : Depuis 2004, le *PASS Navigo* proposé par la *RATP* (*Régie Autonome des Transports Parisiens*) envoie des informations, notamment les validations, au *Stif* (*Syndicat des transports d'Île-de-France*) qui a reçu le droit de la *CNIL* (*Commission Nationale de l'Informatique et des Libertés*) de les conserver au plus 48h dans le cadre de la lutte contre les fraudes [9]. En admettant que chacun des 2,5 millions porteurs du *pass* ne valide que deux fois par jour (aller et retour), cela représente donc 10^7 validations qui doivent être traitées quasiment en temps réel.

1.5 Les méthodes traditionnelles de statistique et le *Big Data*

En statistique, il est d'usage d'utiliser des théorèmes de convergence comme ceux vus dans la ressource *R2.06 - Probabilités 2* pour, par exemple, l'estimation ponctuelle ou l'estimation par intervalle de confiance (voir la ressource *R2.08 - Statistique inférentielle*). Dans ce cadre, la première intuition est qu'un très grand nombre de données permet donc de mieux utiliser ces théorèmes. Néanmoins, cela implique une réflexion sur l'utilisation des outils que nous détaillons dans la suite de cette partie.

1.5.1 Tests

Les tests servent à confronter deux hypothèses $\mathcal{H}_0$ (nulle ou conservatrice) et $\mathcal{H}_1$ (alternative) (voir la ressource *R3.06 - Tests d'hypothèses pour l'analyse bi-variée*). Le principe est que, étant donnée une probabilité d'erreur $\alpha \in]0;1[$, un test de niveau exact α est défini de telle sorte que, si les données vérifient l'hypothèse $\mathcal{H}_0$, il rejettera l'hypothèse *à tord* avec une probabilité α quelque soit la taille de l'échantillon. Toutefois, la puissance du test s'améliore généralement avec la taille. Ainsi, même si nous sommes pas dans l'hypothèse $\mathcal{H}_0$ le plus légèrement possible alors avec une taille suffisamment importante, l'hypothèse sera rejetée.

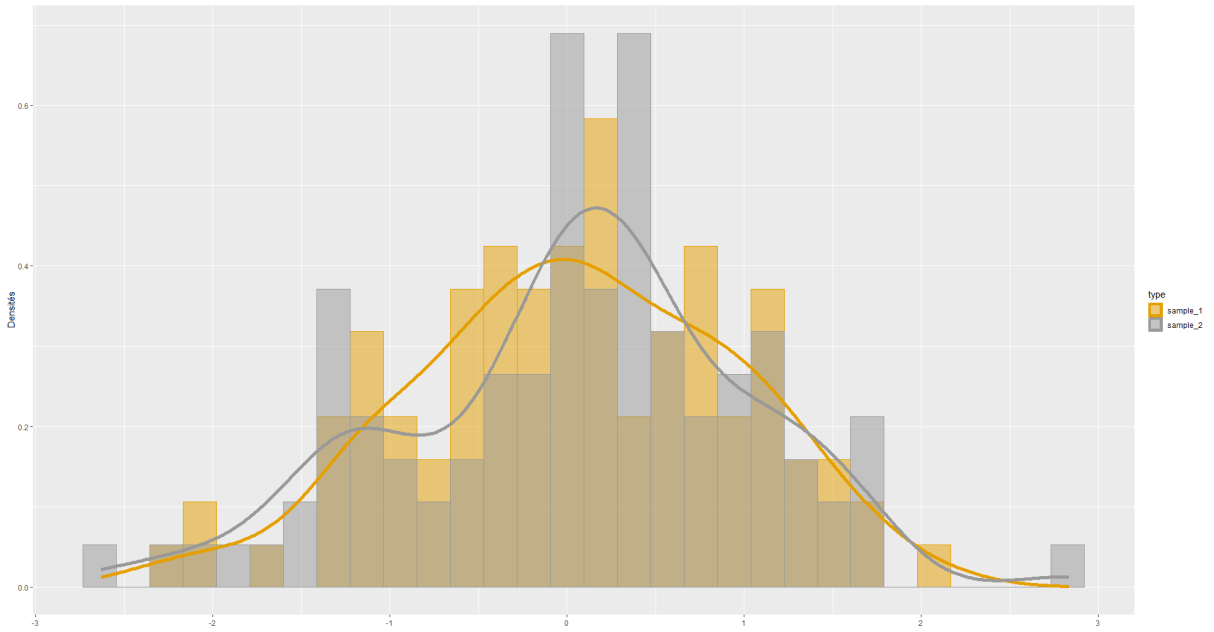


FIGURE 1.4 – Représentation sous forme d'histogramme de deux échantillons indépendants de loi $\mathcal{N}(0,1)$ de taille 100 (en or et gris). Les densités estimées ont été ajoutées.

Par exemple, sur la figure 1.4, nous avons représenté les histogrammes de deux échantillons indépendants et de même loi $\mathcal{N}(0,1)$ de taille 100. Dans ce cas, la p valeur du t test de Student qui teste l'égalité des moyennes vaut environs 0.99 donc nous conservons fortement et à juste titre l'hypothèse d'égalité des moyennes.

En revanche, sur la figure 1.5, nous avons représenté les histogrammes de deux échantillons indépendants de même taille : le premier avec des observations indépendantes et de même loi $\mathcal{N}(0,1)$ (en or) et le second avec des observations indépendantes et de même loi $\mathcal{N}\left(0, 5 \times 10^{-3}\right)$ (en gris). Sur la figure, les

9. http://www.lexpress.fr/high-tech/navigo-le-mouchard-de-la-ratp_562348.html

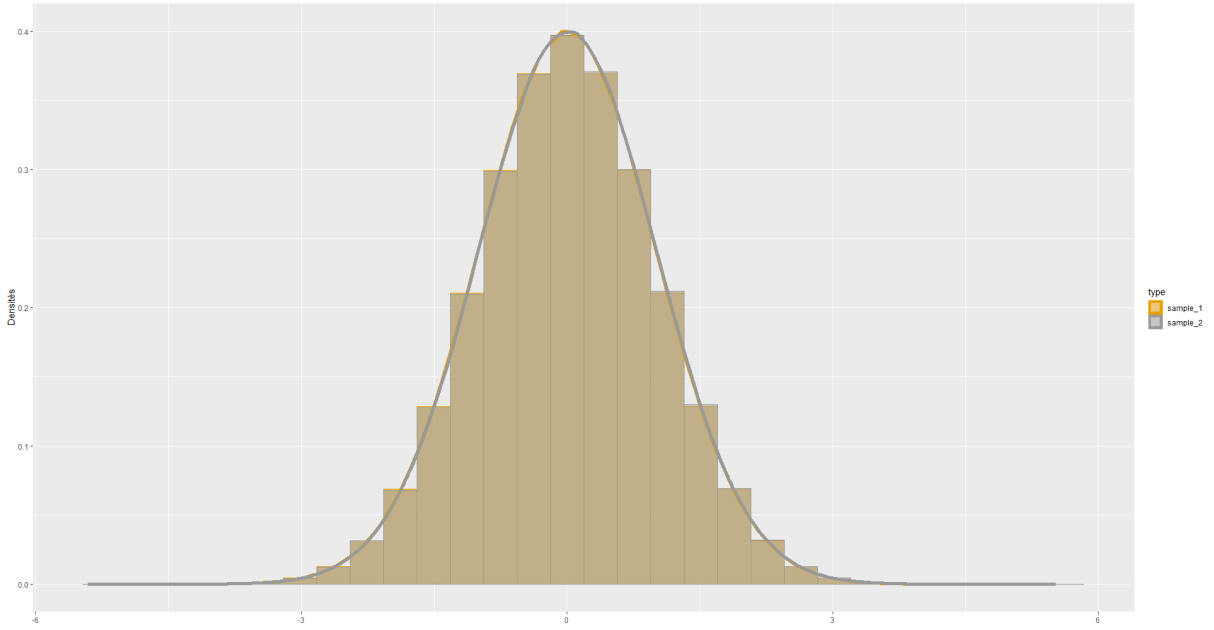


FIGURE 1.5 – Représentation sous forme d'histogramme d'un échantillon de loi $\mathcal{N}(0,1)$ de taille 10^7 (en or) et d'un échantillon de loi $\mathcal{N}\left(5\times10^{-3},1\right)$ de la même taille (en gris). Les densités estimées ont été ajoutées.

densités semblent très proches (surtout en comparaison de la figure 1.5) mais si nous faisons un t test de Student, nous obtenons une p value inférieure à 2.2×10^{-16} et nous rejetons donc l'hypothèse d'égalité des moyennes.

De plus, dans cet exemple, nous *savions* que les données étaient issues d'une loi gaussienne. Néanmoins, il faudrait tester cette normalité par un test de Shapiro-Wilk par exemple. Dans ce cas, il est possible que le test ne soit pas significatif.

Par conséquent, l'utilisation des tests dans le domaine du *Big Data* semble inutile puisque l'hypothèse $\mathcal{H}_0$ sera rejetées avec une très grande probabilité. Ceci se généralise à tous les tests.

1.5.2 Estimation ponctuelle et intervalle de confiance

Dans la ressource *R2.08 - Statistique inférentielle*, l'estimation ponctuelle et par intervalle de confiance ont été étudiées. Par exemple, dans le cas d'un modèle gaussien de variance 1 dont nous cherchons à estimer la moyenne, c'est-à-dire que les observations sont indépendantes et de même loi gaussienne $\mathcal{N}(\theta^\star,1)$, un estimateur de $\theta^\star$ est la moyenne empirique :

$$\widehat{\theta}=\overline{X}_n=\frac{1}{n}\sum_{i=1}^{n}X_i.$$

La loi de l'estimateur est une loi gaussienne $\mathcal{N}(\theta^\star,1/n)$ et un intervalle de confiance de niveau $1-\alpha$ avec $\alpha\in]0;1[$ est donc :

$$\left[\overline{X}_n\pm\frac{q_{1-\alpha/2}}{\sqrt{n}}\right] \tag{1.1}$$

où $q_{1-\alpha/2}$ est la quantile d'ordre $1-\alpha/2$ d'une loi gaussienne centrée réduite c'est à dire la valeur à partir de laquelle une variable gaussienne centrée réduite Z a une probabilité $1-\alpha/2$ d'être en dessous :

$$\mathbb{P}\left(Z\leq q_{1-\alpha/2}\right)=1-\alpha/2.$$

Indépendamment de la valeur de l'observation de $\overline{X}_n$, la longueur de l'intervalle de confiance de l'équation (1.1) est donc $2q_{1-\alpha/2}/\sqrt{n}$ et décroît donc en racine de n. Ainsi, si n est trop grand la longueur sera tellement petite qu'il n'y aura presque plus de différence, pour un être humain, entre l'estimation ponctuelle et celle par intervalle de confiance.

Exercices 1.1

Dans un certain nombre de logiciels comme R ou `Excel` voir `Python` dans le format usuel de stockage des nombres, la limite numérique, c'est-à-dire la limite en dessous de laquelle le logiciel ne fait plus la différence, est d'environ 10^{-16}. Étant donnée une valeur $\alpha \in]0;1[$, calculez à partir de quelle taille n de l'échantillon la longueur de l'intervalle passe en dessous de l'approximation numérique.

1.5.3 Grand nombre de variables

Enfin, un autre exemple de problème est quand la grande dimension est sur le nombre de variables. Ceci peut se retrouver dans le cas de l'analyse du génome ou avec les données fournies par les outils connectées par exemple. Dans ce cas, le nombre de variables peut (largement) dépasser le nombre d'individus et donc créer des problèmes d'estimations. Par exemple, dans le cas d'un modèle linéaire

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{1.2}$$

où $\boldsymbol{Y}$ est un vecteur d'observation de longueur n, $\boldsymbol{X}$ une matrice de covariables de taille $n \times p$, $\boldsymbol{\beta}$ un vecteur de paramètres de longueur p et $\boldsymbol{\varepsilon}$ est un vecteur aléatoire de loi $\mathcal{N}(\mathbb{0}_n, \mathbb{I}_n)$ où $\mathbb{0}_n$ est le vecteur nul de longueur n et $\mathbb{I}_n$ est la matrice identité de taille n. Pour estimer les paramètres $\boldsymbol{\beta}$ du modèle (1.2), si la matrice $\boldsymbol{X}^T\boldsymbol{X}$ est inversible avec $\boldsymbol{X}^T$ représentant la matrice transposée de $\boldsymbol{X}$, nous avons la formule suivante :

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y}. \tag{1.3}$$

Or, pour que la matrice $\boldsymbol{X}^T\boldsymbol{X}$ de l'équation (1.3) soit inversible, il est nécessaire qu'il n'y ait pas de colinéarité entre les colonnes ; c'est-à-dire une combinaison linéaire entre les colonnes. Dès que p est plus grand que n, il y a forcément colinéartié.

Chapitre 2

Volume

"Je ne suis pas gros ! Non monsieur ! Enveloppé, tout juste enveloppé"

Obélix [1]

Dans ce chapitre, nous abordons le problème d'avoir un grand jeu de données et ce que cela implique. Dans ce chapitre, nous présenterons le point de vue d'un·e statistin·ne cherchant à réfléchir au stockage pour ses analyses. La question de l'optimisation du stockage *en général* étant abordée dans la ressource *R6.01 - Big Data : enjeux, stockage et extraction.*

Quand nous parlons de (très) grande dimension, il faut mettre en lien avec notre objectif. Par exemple, pour calculer une moyenne, il faut un nombre immense de valeurs pour que ce calcul commence à être compliqué. En revanche, certaines méthodes demandent beaucoup de manipulations pour avoir un résultat et la taille sera considérée comme trop grande plus vite (nous verrons cela dans le chapitre 3). Les notions de volume et de vélocités sont donc très imbriquées et faire une frontière nette entre les deux est compliquée.

Dans ce chapitre, nous aborderons surtout les questions de la façon de faire un traitement si les données ne sont pas stockées au même endroit et que nous ne pouvons pas les rapatrier facilement.

2.1 Données sur plusieurs serveurs

Dans cette partie, nous supposons que les données sont stockées sur N_S serveurs (voir la figure 2.1). Pour ce faire, nous introduisons quelques notations.

Hypothèse

Nous supposons avoir n individus et N_S serveurs. Nous faisons l'hypothèse que, pour chaque individu $i \in \{1, \dots, n\}$, ses p informations (correspondantes aux réponses des p variables associées) sont stockées sur un seul et unique serveur.

Notations

Pour chaque serveur $s \in \{1, \dots, N_S\}$, nous notons $\mathcal{S}_s$ l'ensemble des individus présents sur le serveur s et n_s le cardinal de cet ensemble, c'est à dire le nombre d'individus présents sur le serveur s.

1. Personnage de fiction créé par René Goscinny et Albert Uderzo dans la bande dessinée Astérix en 1959

Corollaire 1

Comme nous faisons l'hypothèse qu'un individu est stocké sur un et un seul serveur, le groupe d'ensembles $(\mathcal{S}_1, \ldots, \mathcal{S}_{N_S})$ est une partition de $\{1, \ldots, n\}$; c'est-à-dire que nous avons :

$$\bigcup_{s=1}^{N_S} \mathcal{S}_s = \{1, \ldots, n\} \text{ et } \forall s \neq s' \in \{1, \ldots, n_S\}^2, \mathcal{S}_s \cap \mathcal{S}_{s'} = \phi$$

et nous avons :

$$\sum_{s=1}^{N_S} n_s = n.$$

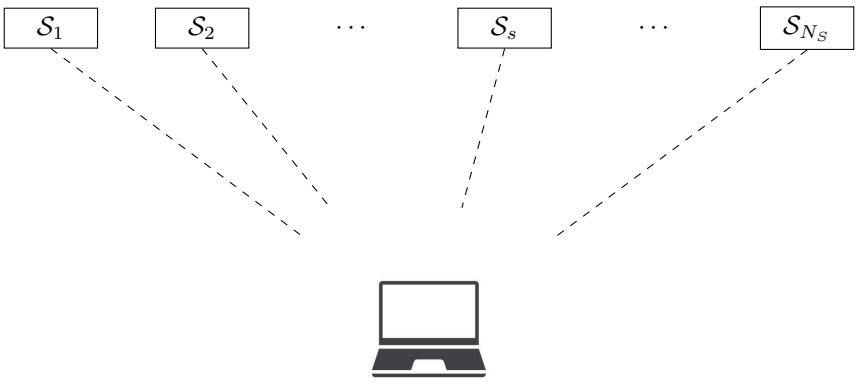


FIGURE 2.1 – Représentation schématique du fait d'avoir les données sur plusieurs serveurs et que nous puissions juste les interroger individuellement.

L'idée étant de réfléchir à l'adaptation des méthodes pour permettre une analyse statistique.

Exemple

Si les données sont stockées sur N_S serveurs et que nous souhaitons faire une moyenne, il suffit de décomposer la formule. Rappelons que nous avons :

$$\overline{Y}_n = \frac{1}{n} \sum_{i=1}^{n} Y_i.$$

Or, par le corollaire 1, nous savons que d'une part

$$n = \sum_{s=1}^{N_S} n_s \tag{2.1}$$

et, d'autre part, nous avons :

$$\sum_{i=1}^{n} = \sum_{s=1}^{N_S} \sum_{i \in \mathcal{S}_s} Y_i. \tag{2.2}$$

Donc, si nous demandons à chaque serveur s de nous transmettre le couple $\left(n_s, \sum_{i \in \mathcal{S}_s} Y_i\right)$, nous pourrons calculer la moyenne en combinant les équations (2.1) et (2.2) :

$$\begin{aligned} \overline{Y}_n &= \frac{1}{n} \sum_{i=1}^{n} Y_i \\ &= \frac{1}{\sum_{s=1}^{N_S} n_s} \sum_{s=1}^{N_S} \sum_{i \in \mathcal{S}_s} Y_i. \end{aligned}$$

Exercices 2.2

Est-il possible de calculer la variance si les données sont réparties sur N_S serveurs ? Si oui, expliquez la procédure. Si non, expliquez pourquoi.

2.2 Mémoire vive et mémoire morte

Quand nous pensons *très grande dimension*, nous pensons à la taille des données stockées dans la *mémoire morte* ; c'est-à-dire la mémoire informatique *non volatile* qui n'est pas prévu pour être modifiée. Or, lorsque nous étudions des données, nous créons et manipulons des objets qui peuvent potentiellement être très gros et ils sont utilisées dans la *mémoire vive*. Or, la taille de la mémoire vive est souvent plus petite que celles de la mémoire morte et il est nécessaire de faire attention à ce que nous manipulons.

Codage en R

Dans le langage R, si la taille d'un objet est trop grand pour la mémoire vive allouée, nous voyons généralement apparaître le message :

```
Error: cannot allocate vector of size 7450.6 Gb
```

Ce message a été obtenu après avoir créer l'objet suivant :

```
1 matrix(runif(10^12),ncol=10^6)
```

Une façon de contourner ce problème est d'augmenter la taille de la mémoire vive allouée à notre logiciel durant le temps que nous l'utilisons ou de limiter l'utilisation d'interfaces pouvant consommer une partie de la mémoire vive (comme l'utilisation du logiciel R `studio`).

Une autre solution est d'utiliser des outils adaptés à l'optimisation de la gestion mémoire (voir la section 2.3.2 ou la ressource *R6.01 - Big Data : enjeux, stockage et extraction.*

2.3 Stockage pertinent

Une autre solution est de réfléchir en amont, lors de la définition du protocole (voir la ressource *R1.04 - Statistique Descriptive 1*), de ce dont nous devons stocker et ce dont nous avons besoin. Dans cette partie, nous présentons donc plusieurs pistes d'exploration.

2.3.1 Stocker le nécessaire uniquement

La première piste, en lien avec le chapitre 6, est de ne stocker que les informations dont nous avons besoin pour l'analyse. D'abord, cela se place dans une réflexion *RGPD* et ensuite, si nous 1 ou 2 variables à stocker, le volume de stockage va certainement doubler.

2.3.2 Format de stockage

Ensuite, suivant les besoins de l'analyse, nous pouvons nous questionner sur la nécessité de stocker les informations de chaque individu ou des résumés.

Résumés des informations : Par exemple, si l'objectif final n'est d'analyser que la moyenne d'une variable quantitative discrète, nous pouvons ne stocker que le tableau de contingence.

Exemple

Par exemple, si nous créons un vecteur aléatoire de taille 10^7 suivant une loi uniforme sur l'ensemble $\{1, \dots, 10\}$ alors la taille mémoire de vecteur est de l'ordre 40Mb en R et `Python`. Si nous prenons *juste* la table de contingence, le stockage est de l'ordre du Kb soit un rapport de 24 752 fois plus petit dans notre exemple pour le language R et 106 383 pour le language `Python`.

Codage en R

Le code de l'exemple précédent est le suivant :

```r
# Définition du vecteur
x <- sample(1:10,10^7,replace=TRUE)
# Taille du vecteur
format(object.size(x), units="Mb")
# Taille du tableau de contingence
format(object.size(table(x)), units="Kb")
# Rapport
object.size(x)/object.size(table(x))
```

Codage en Python

Le code de l'exemple précédent est le suivant :

```python
import numpy as np
from sys import getsizeof
from collections import Counter

# Définition du vecteur
x = np.random.choice(range(1, 11), 10**7, replace=True)

# Taille du vecteur
print(f"Taille du vecteur : {getsizeof(x) / (1024 ** 2):.2f} Mb")

# Taille du tableau de contingence
table_x = Counter(x)
print(f"Taille du tableau de contingence : {getsizeof(table_x) / 1024:.2f} Kb")

# Rapport
rapport = getsizeof(x) / getsizeof(table_x)
print(f"Rapport : {rapport:.2f}")
```

Attention au piège

Notons que si nous stockons qu'une partie des informations, nous prenons le risque de rater une information importante. Nous revenons au problème de bien définir le protocole déjà aborder dans la ressource *R1.04 - Statistique Descriptive 1.*

Précisions dans le stockage : Une autre question est la précision du stockage. Si nous avons une variable continue et, suivant les objectifs, nous pouvons nous poser la question de la précision du stockage.

Exercices 2.3

Simulez un $n = 10^7$ échantillons de loi gaussienne centrée réduite puis calculer la taille du vecteur et faites l'estimation de la moyenne et de la variance. En reprenant le même vecteur d'observations, calculez la taille mémoire du vecteur et faites l'estimation de la moyenne et la variance en prenant une précision de 4, 3, 2 et 1 décimales et en prenant les valeurs entières. Pour chacun de ces vecteurs tronqués, calculez la taille mémoire du tableau de contingence associé.

Utilisations des outils optimisés : Enfin, il existe des solutions systèmes pour améliorer la gestion de ces objets. Par exemple, les matrices *sparses*, c'est-à-dire celles avec un grand nombre de 0, il existe des packages pour un meilleur stockage. Toutefois, il est aussi important de noter que tous les langages ne gèrent pas de la même façon le stockage.

Exemple

Si nous simulons une matrice *sparse* contenant en moyenne 10^{-4} valeurs proches de 1 et les autres cases égales à 0, nous obtenons dans le langage R, une taille mémoire d'environ 762.9 Mb alors que si nous

utilisons le package `Matrix`, nous obtenons une taille mémoire de 0.2 Mb soit environ 4955 fois moins. En revanche, dans le langage `Python`, nous passons de 128 octets en taille mémoire avec la bibliothèque `numpy` pour 48 octets si nous utilisons la fonction `csr_matrix` de la bibliothèque `scipy.sparse`.

Codage en R

Le code de l'exemple précédent est le suivant :

```r
# Création de la matrice sparse
eps <- 10^(-3)
n<-10^4
x<-runif(n*n)
y<-x
y[x<(1-eps)]<-0
A <- matrix(y,ncol=n,nrow=n)
# Taille de la matrice
format(object.size(A), units="Mb")
# Utilisation d'outils
library(Matrix)
format(object.size(Matrix(A)), units="Mb")
```

Codage en Python

Le code de l'exemple précédent est le suivant :

```python
import numpy as np
from scipy.sparse import csr_matrix
from sys import getsizeof

# Création de la matrice sparse
eps = 1e-3
n = 10**4
x = np.random.rand(n * n)
y = np.copy(x)
y[x < (1 - eps)] = 0
A = np.reshape(y, (n, n))

# Taille de la matrice
print(f"Taille de la matrice : {getsizeof(A)} octets")

# Utilisation d'outils
from scipy.sparse import csr_matrix
A_sparse = csr_matrix(A)
print(f"Taille de la matrice sparse : {getsizeof(A_sparse)} octets")
```

2.3.3 Transformation des données avant le stockage

Enfin, une dernière idée est de réfléchir en amont s'il est nécessaire de stocker les données telles quelles ou au contraire une transformation.

Exemple

Une entreprise cherchait à mettre à disposition les moyennes de températures entre deux temps t_1 et t_2 sur une année avec des mesures faites toutes les secondes (soient 31 536 000 pour une année non bissextile). La première intuition serait de stocker les températures T_t à chaque instant et de calculer les moyennes en faisant la somme :

$$\overline{T}_{t_1:t_2} = \frac{1}{t_2 - t_1} \sum_{t=t_1+1}^{t_2} Y_t.$$

Néanmoins, dans un soucis de vitesse (voir le chapitre 3), nous pouvons aussi décider de stocker les sommes S_t jusqu'à chaque instant :

$$S_t = \sum_{t'=1}^{t} Y_{t'}$$

et ainsi, le calcul de chaque moyenne serait quasi instantané :

$$\overline{T}_{t_1:t_2} = \frac{1}{t_2 - t_1} \sum_{t=t_1+1}^{t_2} Y_t = \frac{1}{t_2 - t_1} \left[\sum_{t=1}^{t_2} Y_t - \sum_{t=1}^{t_1} Y_t \right] = \frac{S_{t_2} - S_{t_1}}{t_2 - t_1}.$$

Exercices 2.4

Peut-on utiliser une technique similaire pour la variance ?

Attention au piège

Dans le cadre de l'exemple précédent, le fait de stocker des sommes pourraient amener aussi à des problèmes numériques (surtout les valeurs sont trop grandes). Une technique est alors de stocker la somme des écarts à une valeur $\mu \in \mathbb{R}$; par exemple, la moyenne annuelle des températures. Ainsi, nous aurons :

$$S_t^{(\mu)} = \sum_{t'=1}^{t} (Y_{t'} - \mu)$$

et pour calculer la moyenne, nous remarquons que :

$$\begin{aligned} \frac{S_{t_2}^{(\mu)} - S_{t_1}^{(\mu)}}{t_2 - t_1} &= \frac{1}{t_2 - t_1} \left[\sum_{t=1}^{t_2} (Y_t - \mu) - \sum_{t=1}^{t_1} (Y_t - \mu) \right] \\ &= \frac{1}{t_2 - t_1} \sum_{t=t_1+1}^{t_2} (Y_t - \mu) \\ &= \frac{1}{t_2 - t_1} \left[\sum_{t=t_1+1}^{t_2} Y_t - (t_2 - t_1)\,\mu \right] \\ &= \overline{T}_{t_1:t_2} - \mu. \end{aligned}$$

Ainsi, en prenant $\frac{S_{t_2}^{(\mu)} - S_{t_1}^{(\mu)}}{t_2 - t_1} + \mu$, nous obtenons la moyenne souhaitée.
De plus, le fait de prendre une estimation de la moyenne annuelle pour μ permettrait que les valeurs stockées oscillent autour de 0.

Chapitre 3

Vélocité

"Vous [les statisticien·ne·s] n'êtes jamais contents : avant, nous n'avions pas assez de données et les résultats obtenus n'étaient pas souvent concluants ; maintenant, nous en avons trop et nous ne pouvons pas avoir de résultats dans des temps corrects."

Discussion avec un biologiste.

Comme expliquer dans le chapitre 1 d'introduction et le chapitre 2 sur les volumes, la problématique de la vélocité est souvent complémentaire de celle du volume. Dans ce chapitre, nous commencerons par expliquer comment évaluer la vitesse d'une procédure avant de proposer quelques pistes de réflexion.

3.1 Complexité temporelle d'un algorithme

Pour un algorithme, il existe deux types de complexité :

- la **complexité spatiale** qui estime la place **mémoire vive** (donc occupée pendant l'exécution) dont nous avons un peu survolé l'idée dans la section 2.2 et qui est étudié dans la ressource *R6.01 - Big Data : enjeux, stockage et extraction.*
- la **complexité temporelle** qui estime le temps de calcul pour exécuter un algorithme. Nous nous concentrerons sur cette complexité dans ce chapitre. Elle ne sera que survolée dans ce cours mais il exsite de nombreuses ressources sur internet pour les élèves curieux·ses d'approfondir cette notion [1].

Hypothèse

Par abus de langage, nous utiliserons le terme *complexité* sans préciser *temporelle* dans la suite de ce chapitre.

La complexité repose sur la notion d'*opération élémentaire* :

Définition 2 (Opération élémentaire)

Une **opération élémentaire** est une **instruction** à la base de toute programmation ou alors une **étape indivisible** qui peut être exécutée en temps constant.

Exemple

Dans les opérations élémentaires, nous avons :

- Les opérations arithmétiques de bases comme l'addition, la soustraction, la multiplication et la division. Mais aussi le reste et le quotien d'une division euclidienne.
- Les affectations.
- Les comparaisons (égalité ou différence) de deux éléments.

1. Par exemple, nous appuyons largement sur le programme d'informatique de PCSI et notamment sur la ressource disponible sur le lien suivant : https://cahier-de-prepa.fr/psi-michelet/download?id=239

Attention au piège

Quand nous parlons d'égalité d'éléments, il s'agit des éléments de bases. Par exemple, si a et b sont deux scalaires, la comparaison de a et b est une opération élémentaire. En revanche, si $a \in \mathbb{R}^2$ et $b \in \mathbb{R}^2$, la comparaison des deux ne sera plus élémentaire car nous comparons généralement chaque coordonnée (donc deux opérations élémentaires).

Exercices 3.5

Calculez le nombre exact d'opérations élémentaires pour calculer la moyenne d'un n échantillon de variables quantitatives continues. Qu'en est-il si c'est une variable quantitative discrète ?

Le principe de la complexité temporelle est de compter le nombre d'opérations. Toutefois, il n'est pas nécessaire d'avoir le nombre exact mais un *ordre de grandeur*. Pour cela, nous utilisons la notion de *domination* de suites.

Définition 3 (Notation de Landau)

Étant données deux suites numériques $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$, nous disons que la suite $(u_n)_{n\in\mathbb{N}}$ **domine** la suite $(v_n)_{n\in\mathbb{N}}$ que nous notons $v_n = \mathcal{O}(u_n)$ s'il existe une une valeur $n_0 \in \mathbb{N}$ et une constante strictement positive $C \in \mathbb{R}_+^\star$ telles qu'à partir de la valeur n_0, la suite $(C\,|u_n|)_{n\in\mathbb{N}}$ est toujours plus grande que la suite $(|v_n|)_{n\in\mathbb{N}}$:

$$\forall n \geq n_0,\ |v_n| \leq C\,|u_n|\,.$$

Exemple

Une fonction élémentaire a une complexité temporelle en $\mathcal{O}(1)$.Toute fois, une fonction élémentaire effectuée n fois (par exemple, dans une boucle allant de 1 à n) a une complexité de $\mathcal{O}(n)$.

D'après l'exercice précédent, le nombre d'opérations pour calculer une moyenne d'un n échantillon de variables continues est en $\mathcal{O}(n)$.

Proposition 2 (Simplification)

- **Equivalence** : Soient trois suites $(u_n)_{n\in\mathbb{N}}$, $(v_n)_{n\in\mathbb{N}}$ et $(w_n)_{n\in\mathbb{N}}$ telles que $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ sont **équivalentes**, c'est-à-dire que $u_n = \mathcal{O}(v_n)$ et $v_n = \mathcal{O}(u_n)$, alors nous avons la relation :
$$w_n = \mathcal{O}(u_n) \Rightarrow w_n = \mathcal{O}(v_n)\,.$$
- **Transitivité** : Soient trois suites $(u_n)_{n\in\mathbb{N}}$, $(v_n)_{n\in\mathbb{N}}$ et $(w_n)_{n\in\mathbb{N}}$ telles que $v_n = \mathcal{O}(u_n)$ et $w_n = \mathcal{O}(v_n)$ alors $w_n = \mathcal{O}(u_n)$.

Preuve

La transitivité impliquant l'équivalence, nous ne démontrons que la deuxième propriété. D'après l'énoncé, $v_n = \mathcal{O}(u_n)$ donc, par définition, il existe $n_1 \in \mathbb{N}$ et une constante strictement positive $C_1 \in \mathbb{R}_+^\star$ telles que :

$$\forall n \geq n_1,\ |v_n| \leq C_1\,|u_n|\,.$$

De même, comme $w_n = \mathcal{O}(v_n)$, il existe $n_2 \in \mathbb{N}$ et une constante strictement positive $C_2 \in \mathbb{R}_+^\star$ telles que :

$$\forall n \geq n_2,\ |w_n| \leq C_2\,|v_n|\,.$$

Donc, si nous prenons $n_0 = \max(n_1, n_2)$, dès que n est plus grand que n_0, il est plus grand que n_1 et n_2 et nous avons donc pour tout n plus grand que n_0 :

$$\begin{aligned} |w_n| &\leq C_2\,|v_n| \\ &\leq \underbrace{C_2 C_1}_{=C>0}\,|u_n| \end{aligned}$$

ce qui est la définition de $w_n = \mathcal{O}(u_n)$.

Exemple

Par exemple, nous avons $n + n^2 = \mathcal{O}(n^2)$ ou encore $n + 1 = \mathcal{O}(n)$.

Définition 4 (Complexité temporelle)

Pour calculer la **complexité temporelle** d'un algorithme, nous faisons l'hypothèse qu'elle est proportionnelle au nombre d'opérations élémentaires **dans le pire des cas**.

Remarque

La notion de **pire des cas** signifie que, si plusieurs possibilités existent, nous devons toujours prendre le choix le plus long.

Exemple

Dans l'exercice précédent, nous avons vu que la complexité pour le calcul d'une moyenne pour une variable continue est en $\mathcal{O}(n)$. Si nous avons une variable quantitative discrète dont nous connaissons le tri à plat, nous pouvons améliorer la vitesse. Néanmoins, sans information et même si, au final, ça aurait été une variable quantitative discrète, nous devons considérer que c'est une variable continue.

Définition 5 (Complexités usuelles)

Si nous notons $T(n)$ le temps d'exécution en fonction de la taille n des objets en entrée, nous disons que :

- La complexité est **constante** si $T(n) = \mathcal{O}(1)$.
- La complexité est **logarithmique** si $T(n) = \mathcal{O}(\log n)$.
- La complexité est **linéaire** si $T(n) = \mathcal{O}(n)$.
- La complexité est **quasi-linéaire** si $T(n) = \mathcal{O}(n \log n)$.
- La complexité est **quadratique** si $T(n) = \mathcal{O}(n^2)$.
- La complexité est **cubique** si $T(n) = \mathcal{O}(n^3)$.
- Plus généralement, la complexité est **polynomiale** si $T(n) = \mathcal{O}(n^\alpha)$ avec $\alpha > 0$.
- La complexité est **exponentielle** s'il existe $a > 1$ telle que $T(n) = \mathcal{O}(a^n)$.

Exercices 3.6

Comment appelle-t-on la complexité du calcul d'une moyenne ?

Exercices 3.7

Donnez la complexité de chacun des algorithmes suivants codés en R :

```
## Algorithme 1
x <- rnorm(n)
xbar<-mean(n)
s<-0
for (i in 1:n){
    s<-s+(x[i]-xbar)^2
```

```
}
s<−sqrt(s/n)
## Algorithme 2
x <− rnorm(n)
sd(x)∗(n−1)/n
## Algorithme 3
x <− rnorm(n)
s<−0
for (i in 1:n){
    xbar<−mean(n)
    s<−s+(x[i]−xbar)^2
}
s<−sqrt(s/n)
```

Exemple

Dans le classement du Top 500 mis à jour en mai 2023, le supercalculateur le plus puissant est `Frontier` (États-Unis) avec 1,194 Exaflop/s soient 10^{18} opérations par secondes. Dans le tableau 3.1, nous avons repris les ordres de grandeurs des données de la section 1.4 et avons calculé combien il faudrait de temps à `Frontier` pour faire le calcul des valeurs [2]. Par comparaison, nous avons aussi afficher en gris le temps mis par `Sequoia` d'IBM qui était le plus puissant en 2012 avec $1,632 \times 10^{16}$ opérations à la seconde. Nous pouvons déjà remarquer qu'en 11 ans, nous n'avons gagné *qu'un* rapport de 100 et, bien que cela rend certains calculs réalisables en temps raisonnable (par exemple, une puissance quadratique pour traiter les $1,584 \times 10^{10}$ pixels d'un film d'une minute d'un téléphone performant des années 2016), il est impossible de faire des traitements quadratiques pour des tailles de 10^{13}. En revanche, si on arrive à passer d'une complexité cubique à quadratique, il est tout à fait possible de faire en quelques secondes sur `Sequoia` un calcul quadratique qui prendrait quelques dizaines de millénaires en cubique sur `Frontier`. À l'opposé, avoir une complexité exponentielle est impossible dès qu'on a 100 individus même pour les supercalculateurs.

TABLE 3.1 – Estimation du temps pour les supercalculateurs `Frontier` (en noir) et `Sequoia` (en gris) en fonction de la taille des données et des opérations.

	$\log(n)$	n	n^2	n^3	2^n
$n = 100$	4 as 282 as	84 as 6 fs	8 fs 613 fs	845 fs 61 ps	814 millénaires 2461 millénaires
$n = 1000$	6 as 423 as	845 as 61 fs	845 fs 61 ps	845 ps 61 ns	5×10^{266}×âge de l'univers 10^{267}×âge de l'univers
$n = 4,6 \times 10^6$	13 as 940 as	4 ps 282 ns	18 μs 1 ms	1 min 2 h	∞ ∞
$n = 10^7$	14 as 988 as	8 ps 613 ns	84 μs 6 ms	14 min 17 h	∞ ∞
$n = 10^9$	18 as 1 fs	845 ps 62 ns	845 ms 1 min	6 siècles 2 millénaires	∞ ∞
$n = 3.4 \times 10^9$	19 as 1 fs	3 ns 208 ns	10 s 12 min	76 millénaires 76 millénaires	∞ ∞
$n = 1,584 \times 10^{10}$	20 as 1 fs	13 ns 971 ns	4 min 4 h	25 millénaires 7717 millénaires	∞ ∞
$n = 1,4 \times 10^{13}$	26 as 2 fs	12 μs 858 μs	5 ans 380 ans	10^5×âge de l'univers 4×10^5×âge de l'univers	∞ ∞

attoseconde (as ou 10^{-18}s), femtoseconde (fm, 10^{-15}s), picoseconde (ps, 10^{-12}s), microseconde (μs, 10^{-9}s), nanoseconde (ns, 10^{-6}s), milliseconde (ms, 10^{-3}s), seconde (s), minute (min), heure (h), année (ans), siècle (100 ans), millénaire (1000 ans), âge du soleil ($4,57 \times 10^9$ années) et âge de l'univers ($1,379 \times 10^{10}$ années). ∞ signifie que le calcul a dépassé la capacité numérique du logiciel R.

2. Source https://www.silicon.fr/supercalculateurs-10-systemes-puissants-465864.html

Exercices 3.8

Étant donné un n-échantillon, quelle est la complexité pour extraire toutes les parties de cette échantillon (c'est-à-dire, l'ensemble vide, tous les individus, tous les couples, tous les triplets et ainsi de suite jusqu'à l'échantillon complet) ?

Attention au piège

Une complexité est indépendante de la machine que nous utilisons, c'est un **ordre de grandeur**. Par exemple, une machine très performante pourra aller plus vite avec une procédure de complexité quadratique qu'une machine moins performante avec une procédure avec une complexité linéaire à une taille donnée (voir le tableau 3.1). Néanmoins, cette tendance finira par s'inverser lorsque la taille augmentera.

3.2 Optimisation d'une procédure

Pour optimiser une procédure, il existe plusieurs techniques qui s'acquièrent avec la pratique. Nous listons quelques points de vigilances permettant de gagner plus ou moins de temps dans un code.

3.2.1 Code propre

Vous avez déjà vu en ressources *R1.03 - Bases de la programmation 1*, *R2.03 - Bases de la programmation 2* et *R2.04 - Programmation statistique* notamment, l'importance d'avoir un code propre. Pour cela, il faut déjà réfléchir à l'organisation notamment à l'emplacement de certaines commandes.

Exemple

Dans l'exercice 3.1, l'algorithme 3 calculer à chaque itération de la boucle la moyenne alors que la valeur ne change pas. La sortir (comme dans l'algorithme 1) permet d'améliorer la vitesse de convergence. Sur les figures 3.1 et 3.2, nous avons représenté les résultats de 10 lancers pour chacun des algorithmes en prenant $n = 10^5$. Nous voyons que les algorithmes 1 et 2 sont du même ordre de grandeur (de l'ordre de la dizaine de millisecondes) tandis que l'algorithme 3 est de l'ordre de la centaine de millisecondes. Cet écart serait accentué si on augmente la valeur de n.

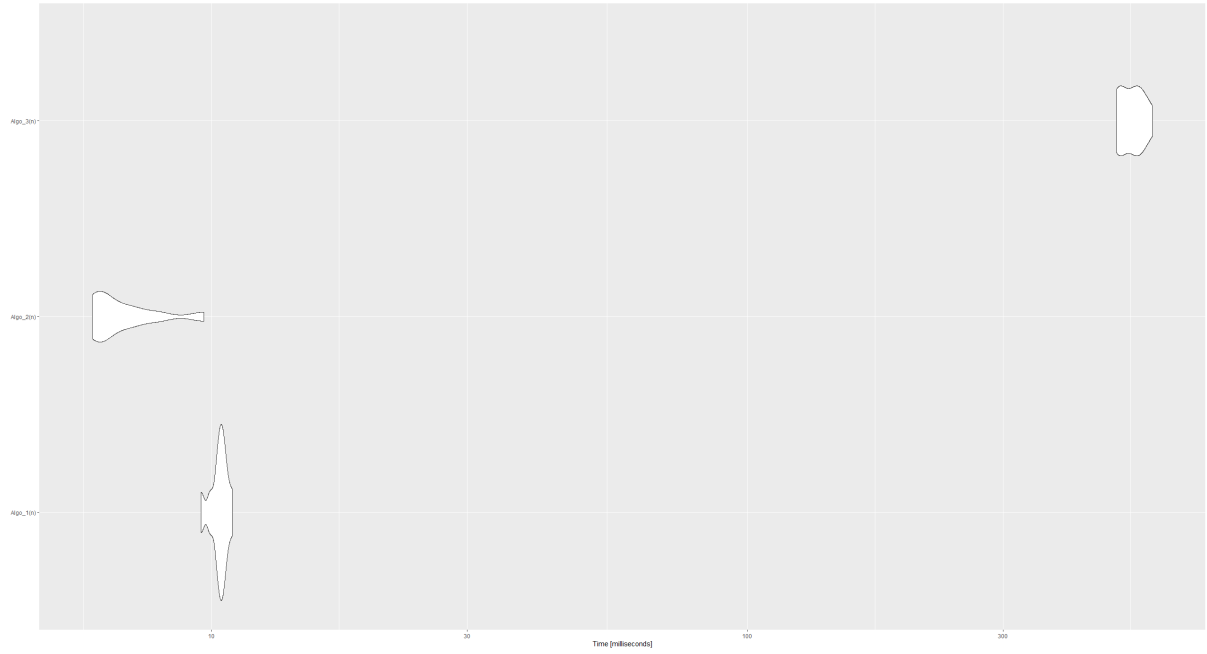


FIGURE 3.1 – Représentation sous forme de *violin* plot de dix lancers des trois algorithmes de l'exercice 3.1 (version R).

Codage en R

Le code utilisé pour faire la figure 3.1 est :

```
#### Chargement des packages
library(microbenchmark)
library(ggplot2)
#### Définition des algorithmes
## Algorithme 1
Algo_1<-function(n){
  x <- rnorm(n)
  xbar<-mean(n)
  s<-0
  for (i in 1:n){
    s<-s+(x[i]-xbar)^2
  }
  sqrt(s/n)
}
## Algorithme 2
Algo_2<-function(n){
  x <- rnorm(n)
  sd(x)*(n-1)/n
}
## Algorithme 3
Algo_3<-function(n){
  x <- rnorm(n)
  s<-0
  for (i in 1:n){
    xbar<-mean(n)
    s<-s+(x[i]-xbar)^2
  }
  sqrt(s/n)
}
#### Calcul du temps
n <-10^5
res <- microbenchmark( Algo_1(n),Algo_2(n),Algo_3(n), times=10L)
#### Affichage
autoplot(res)
```

Codage en Python

Le code `Python` (qui donne la figure) est le suivant :

```
import numpy as np
import timeit
import seaborn as sns
import matplotlib.pyplot as plt

# Définition des algorithmes
def algo_1(n):
    x = np.random.randn(n)
    xbar = np.mean(x)
    s = np.sum((x - xbar) ** 2)
    return np.sqrt(s / n)

def algo_2(n):
    x = np.random.randn(n)
    return np.std(x) * (n - 1) / n

def algo_3(n):
    x = np.random.randn(n)
    s = 0
    for i in range(n):
```

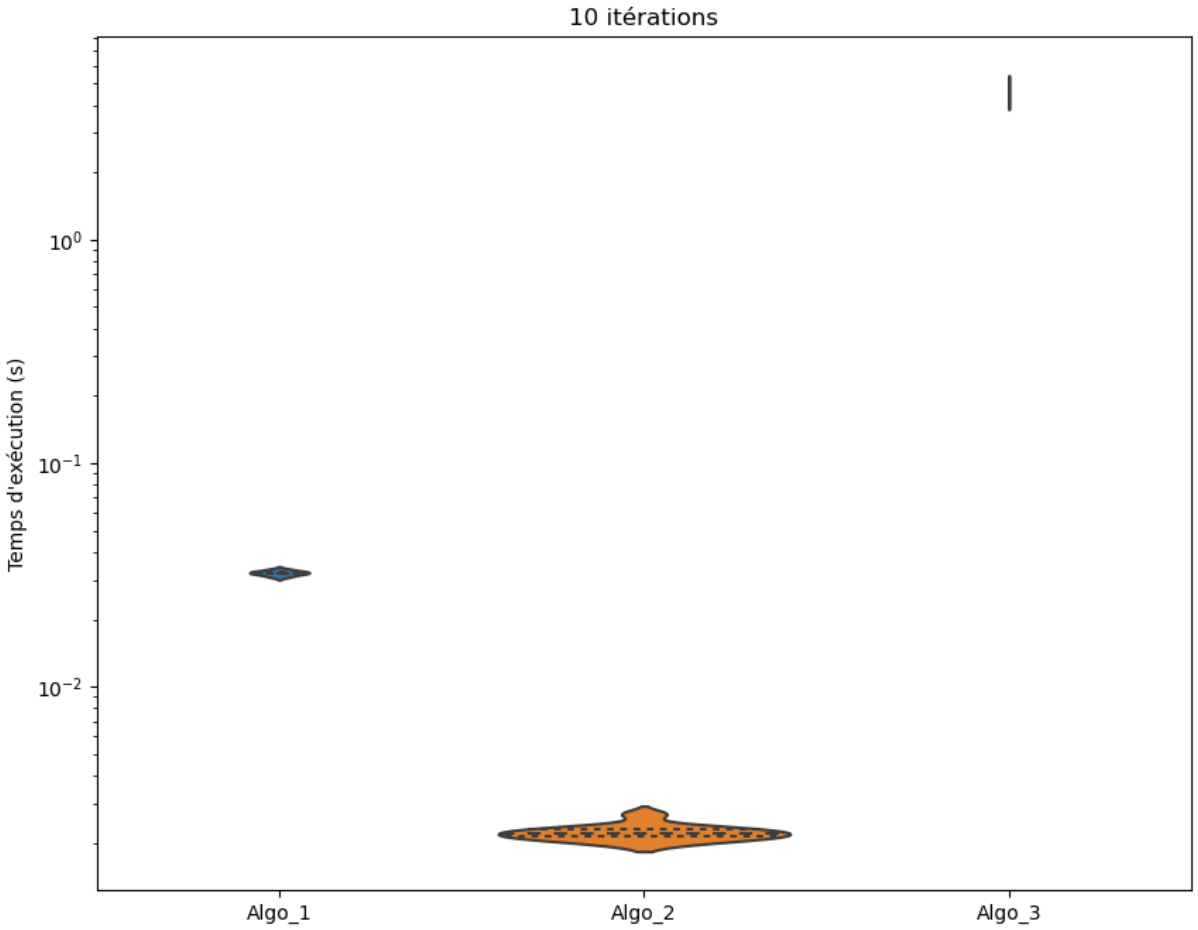


FIGURE 3.2 – Représentation sous forme de *violin* plot de dix lancers des trois algorithmes de l'exercice 3.1 (version `Python`).

```
        xbar = np.mean(x)
        s += (x[i] - xbar) ** 2
    return np.sqrt(s / n)

# Paramètres
n = 10**5
times = 10

# Mesure du temps d'exécution
results_algo_1 = [timeit.timeit(lambda: algo_1(n), number=1) for _ in range(times)]
results_algo_2 = [timeit.timeit(lambda: algo_2(n), number=1) for _ in range(times)]
results_algo_3 = [timeit.timeit(lambda: algo_3(n), number=1) for _ in range(times)]

# Affichage du violin plot
data = [results_algo_1, results_algo_2, results_algo_3]
labels = ["Algo_1", "Algo_2", "Algo_3"]

sns.violinplot(data=data, inner="quartile")
plt.xticks(range(len(labels)), labels)
plt.ylabel("Temps d'exécution (s)")
plt.yscale("log") # Échelle logarithmique
plt.title(f"{times} itérations")
plt.show()
```

Notons que les temps d'exécutions pour l'algorithme 3 sont plutôt de l'ordre de la seconde voir du dixième de secondes.

Parfois, cela peut être l'interversion de deux boucles `for` imbriquées qui permettra d'améliorer le temps car des calculs récurrents seront faits moins souvent ou que l'espace mémoire sera mieux géré. N'hésitez pas à faire des tests variés et à comparer vos procédures.

3.2.2 Améliorer la complexité

Une solution pour améliorer la vitesse est de diminuer en complexité. Par exemple, pour inverser une matrice $A \in GL_n(\mathbb{R})$, si nous utilisons la **formule de Laplace** :

$$A^{-1} = \frac{1}{|A|}\text{com}(A)^T$$

avec $|A|$ le déterminant de la matrice A et $\text{com}(A)$ la comatrice, alors la complexité est de $\mathcal{O}(n!)$ c'est à dire plus de l'ordre de $\mathcal{O}\left(n^{n+1/2}\right)$ (par la **formule de Stirling**).

Ceci est principalement dû aux calculs des déterminants dans la comatrice. Nous pouvons donc améliorer cette complexité en prenant le **pivot de Gauss** pour le calcul du déterminant et ainsi obtenir une complexité totale de $\mathcal{O}\left(n^5\right)$.

Toutefois, quitte à utiliser le pivot de Gauss, autant le faire directement pour inverser la matrice. Ainsi, le **pivot de Gauss** ou l'**l'élimination de Gauss-Jordan** permet d'inverser une matrice avec une complexité de $\mathcal{O}\left(n^3\right)$.

On pourrait se dire qu'il serait compliqué de faire mieux mais l'**algorithme de Strassen** découvert en 1969 va encore améliorer la complexité en décomposant les matrices par blocs et en ne faisant que 7 opérations au lieu de 8 dans les blocs atteignant ainsi une complexité de $\mathcal{O}\left(n^{\log_2(7)}\right)$.

Cette complexité avait déjà été atteinte en 1902 dans le cas d'une matrice **définie positive** à l'aide d'une **factorisation de Cholesky**.

Après, il existe des cas particuliers où la complexité sera plus faible. Par exemple, si la matrice est triangulaire, le pivot de Gauss donne une complexité de $\mathcal{O}\left(n^2\right)$.

Et enfin, si la matrice est diagonale, la complexité est linéaire puisqu'il s'agit d'inverser chaque valeur de la diagonale donc $\mathcal{O}(n)$.

Il est donc important de faire un travail en amont pour voir s'il n'existe pas des procédures plus rapides ou si nos données ne sont pas suffisamment particulières pour optimiser certains calculs.

Exemple

Dans leur article, Brault et al. (2017) utilisaient une méthode avec une matrice A de taille $n^2 \times n^2$ donc le produit de la matrice par un vecteur v était d'une complexité $\mathcal{O}\left(n^4\right)$. Toutefois, la structure de la matrice était un produit de Kronecker de deux matrices triangulaires remplies de 1. En mettant les valeurs du vecteur v dans une matrice carrée de taille $n \times n$, il suffisait de faire des sommes cumulées sur les lignes et les colonnes ce qui permettait de passer à une complexité de $\mathcal{O}\left(n^2\right)$.

Une technique d'amélioration classique est la technique algorithmique du **diviser pour mieux régner** utilisée, par exemple, dans l'algorithme de Strassen. L'idée est de décomposer les valeurs en plusieurs sous partie dont le calcul est plus rapide et de faire remonter les informations.

Exemple

Étant donnés un entier $n \in \mathbb{N}^\star$ et un scalaire $a \in \mathbb{R}^\star$, le calcul de a^n peut se faire de façon linéaire en multipliant a par a n fois ; dans ce cadre, nous avons une complexité linéaire. Une autre solution est de constater que nous avons les deux relations suivantes :

$$\begin{cases} a^{2n} = a^n \times a^n = \left(a^n\right)^2, \\ a^{2n+1} = a^n \times a^n \times a = \left(a^n\right)^2 \times a. \end{cases}$$

Ainsi, nous obtenons l'algorithme récursif qui va regarder si n est pair ou impair puis faire le sous calcul. Ici, nous obtenons une complexité logarithmique.

Attention au piège

Comme la complexité est un ordre de grandeur, une procédure qui serait plus rapide pour une taille d'échantillon pourrait devenir plus lente pour une autre. Par exemple, sur la figure 3.3, nous avons représenté les temps pour calculer $1,0001^n$ avec $n = 10^3$ (à gauche) et $n = 10^5$ (à droite) pour quatre algorithmes :

- `power_rec` utilise la formule de la récurrence présentée dans l'exemple,
- `power_R` utilise la fonction naturelle de R,
- `power_prod` fait un calcul naïf avec des fonctions optimisées,
- `power_naif` qui utilise une boucle `for`.

Nous observons que, pour $n = 10^3$, la fonction naïve avec des fonctions optimisées fait mieux que la récurrence mais que pour $n = 10^5$, la récurrence va plus vite. Notons aussi que nous avons dû prendre une valeur pour a très proche de 1 car, dès que n est un peu grand, nous dépassons la capacité mémoire.

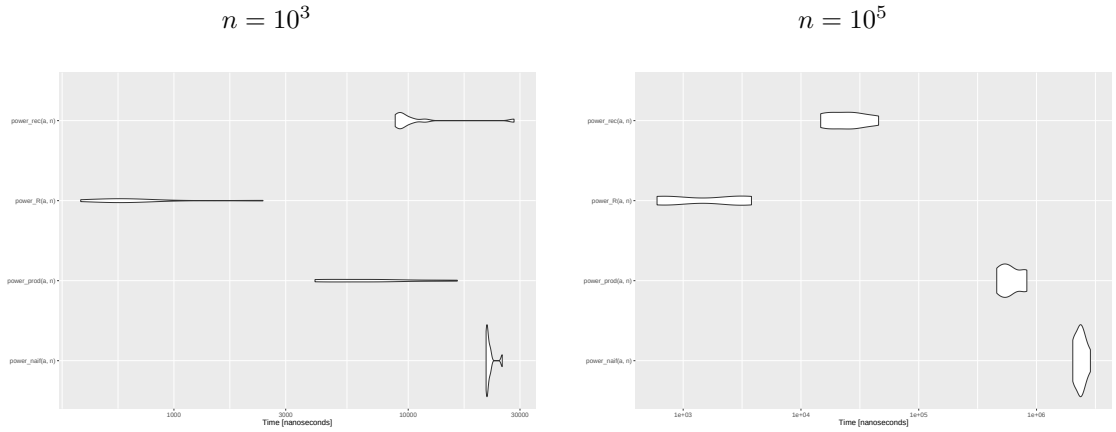


FIGURE 3.3 – Représentation des temps de quatre algorithmes pour calculer a^n avec $a = 1,0001$ et $n = 10^3$ (à gauche) et $n = 10^5$ (à droite) lancés chacun 10 fois : de haut en bas, nous avons `power_rec` qui utilise la formule de la récurrence, `power_R` qui utilise la fonction naturelle de R, `power_prod` qui fait un calcul naïf avec des fonctions optimisées et `power_naif` qui utilise une boucle `for`.

Codage en R

Le code utilisé pour la figure 3.3 est le suivant :

```
#### Power le plus naïf possible
power_naif<-function(a,n){
  p<-1
  for (i in 1:n){
    p<-p*a
  }
  p
}
#### Power utilisant des fonctions intelligentes
power_prod<-function(a,n){
  prod(rep(a,n))
}
#### Power de R
power_R<-function(a,n){
  a^n
}
#### Power récursif
power_rec<-function(a,n){
  if (n==0){
    return(1)
  }else{
```

```
    temp<-power_rec(a,n%/%2)
    if (n%%2==0)
    {
      return(temp*temp)
    }else{
      return(temp*temp*a)
    }
  }
}
#### Packages
library(microbenchmark)
library(ggplot2)
#### Calcul du temps
a<-1.0001
for (n in 10^c(3,5)){
  res <- microbenchmark( power_naif(a,n),power_prod(a,n),power_R(a,n),power_rec(a,n), times=10L)
  #### Affichage
  print(autoplot(res,main=paste0("n=",n)))
}
```

Il est donc nécessaire de vérifier que nos changements de codes ont effectivement améliorer les temps pour nos utilisations.

3.2.3 Parallélisation

Un point abordé dans la ressource *R6.01 - Big Data : enjeux, stockage et extraction* est le fait de paralléliser son code. Ceci permet de diminuer le temps de calcul mais ne change rien à la complexité. Par exemple, si nous arrivons à paralléliser le code sur 10 coeurs ou serveurs, nous divisons au mieux le temps par 10 mais, par exemple, un dixième de millénaire reste un temps trop grand.

3.2.4 Stockage intelligent

Comme vue dans la section 2.3.3, nous pouvons jouer sur le stockage des données.

Exemple

Dans l'exemple sur les températures, nous passons d'une complexité linéaire à une complexité constante voir instantanée car nous n'avons que 4 opérations à faire.

3.2.5 Précision et vitesse

Comme pour le stockage, une façon de gagner du temps est de renier sur la précision. Par exemple, il n'est pas forcement nécessaire de faire des calculs avec une forte précision car, la plupart du temps, les erreurs d'approximation vont se compenser.

Une autre solution est de prendre une sous partie de la base et de procéder aux calculs sur cette sous-partie ; si elle est suffisamment représentative, l'estimation obtenue sera très proche de la valeur si nous avions pris la base entière.

Exemple

Avec la mise à disposition de la version 4 de ChatGPT, plusieurs utilisateurs ont signalé des problèmes laissant penser que la version était de moins bonne qualité [3]. Ceci a été corrélée avec une plus grande rapidité. Une des hypothèses (entre autres) est qu'il y ait une légère dégradation de la qualité au profit de la rapidité d'exécution ; surtout que cette rapidité est d'autant plus nécessaire que le nombre d'utilisateur·trice·s a augmenté.

3. Voir par exemple l'article https://www.lebigdata.fr/chatgpt-gpt-4-devenue-bete ou la vidéo de MICODE https://youtu.be/rAG5YtBsBTw?si=WpxOb1EY_yKG27x3

3.3 Pour aller plus loin

Voici quelques exercices pour aller plus loin.

Exercices 3.9

Pour rappel, l'algorithme de l'Analyse en Composante Principale (ou ACP) vu dans la ressource *R4.02 - Méthodes factorielles* est le suivant :

Entrée : la matrice de données $\boldsymbol{x}$.

---

1. Centrage et réduction des variables de la matrice $\boldsymbol{x}$.
2. Calcul du produit $\boldsymbol{x}^T\boldsymbol{x}$.
3. Recherche des valeurs propres de la matrice $\boldsymbol{x}^T\boldsymbol{x}$ ainsi que les vecteurs propres associés.
4. Projection sur les sous-espaces propres des variables en calculant $\boldsymbol{x}u_s$.
5. Projection des individus sur les sous-espaces en calculant $\frac{1}{\lambda_s}\boldsymbol{x}^T\boldsymbol{x}u_s$.

Sortie : affichage graphique des résultats.

Calculez la complexité. Que se passe-t-il si p est plus petit que n ?

Chapitre 4

Variété

"L'univers est infini dans sa variété. Supposer sans savoir, raisonner par analogie, considérer l'inconnu comme allant de soi, ne pouvait que mener au désastre."

L'Enfant tombé des étoiles (1954) de Robert Anson Heinlein

L'augmentation du nombre de données a également été associé à une grande diversité de variables.

4.1 Rappel sur les types de variables

Comme vue dans la ressource *R1.04 - Statistique Descriptive 1*, il existe deux grandes familles de variables : les **quantitatives** sur lesquelles il est possible de faire des opérations arithmétiques et les **qualitatives** sur lesquelles ce n'est pas possible. Ceci entraîne donc des traitements différents (voir par exemple la ressource *R1.04 - Statistique Descriptive 1*) ou encore, dans la ressource *R4.02 - Méthodes factorielles*, les variables quantitatives sont analysées avec l'**Analyse en Composante Principale** (ou **ACP**) tandis que les variables qualitatives sont étudiées avec l'**Analyse Factorielle des Correspondances** (ou **AFC**) ou l'**Analyse des Correspondances Multiples** (ou **ACM**).

Toutefois, l'analyse des deux types de variables, en même temps, va nécessiter des adaptations dans le type des variables ou dans les outils utilisés.

4.2 Transformation des données

L'une des premières solutions est de transformer les données pour les *projeter* dans un ensemble cohérent.

4.2.1 Variables ordinales et variables quantitatives

Nous pouvons transformer une variable quantitative en une variable qualitative ordinale. Pour cela, nous faisons des classes et étudions les classes comme des variables ordinales.

À l'opposé, il arrive qu'on considère des variables qualitatives ordinales comme des variables quantitatives discrètes puisque le codage est parfois associé à une échelle de 1 au nombre de modalités.

Attention au piège

Contrairement au cas où nous utilisons les classes comme des variables ordinales, le fait de considérer une variable qualitative ordinale comme une variable quantitative discrète implique des hypothèses qui n'ont pas de raisons d'être valables.

4.2.2 Copule ?

4.3 Cas particuliers

Dans certains cas, nous pouvons faire des adaptations des procédures.

4.3.1 Distances

Dans le cas de procédures comme la **Classification Ascendante Hiérarchique** (**CAH**), l'algorithme des K**-moyennes** (ou K***-means*** en anglais) vus dans la ressource *R4.03 - Classification automatique* ou encore la méthode des K plus proches voisins vue dans la ressource *R5.02 - Data mining*, tout repose sur la notion des distances.

Dans ce cadre, nous supposons que chaque individu i possède p variables et pour chaque variable $X_{i,j}$, il existe une distance d_j telle que, si nous avons un autre individu i', nous pouvons mesurer la distance entre $X_{i,j}$ et $X_{i',j}$. Ainsi, nous pouvons définir une distance entre $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ en prenant p poids $\boldsymbol{\omega} = (\omega_j)_{1\leq j\leq p} \in \mathbb{R}_\star^+$ et en définissant la distance suivante :

$$d_{\boldsymbol{\omega}}\left(\boldsymbol{X}_i, \boldsymbol{X}_{i'}\right) = \sum_{j=1}^{p} \omega_j d_j\left(X_{i,j}, X_{i',j}\right). \tag{4.1}$$

Exercices 4.10

Étant donnés deux scalaires $\boldsymbol{x} \in \mathbb{R}^p$ et $\boldsymbol{y} \in \mathbb{R}^p$, nous définissons la distance α-Höldérienne avec $\alpha > 0$ de la façon suivante :

$$d_\alpha\left(\boldsymbol{x}, \boldsymbol{y}\right) = \left(\sum_{j=1}^{p} \left|x_j - y_j\right|^\alpha\right)^{1/\alpha}.$$

Rappelez les noms des distances pour $\alpha \in \{1, 2\}$.

Étant donnée une matrice $\boldsymbol{X} \in \mathcal{M}_{n\times p}\left(\mathbb{R}\right)$ et la matrice $\widetilde{\boldsymbol{X}}$ créée en centrant et en réduisant chaque variable. Montrez que pour deux individus i et i', la distance $d_1\left(\widetilde{\boldsymbol{X}}_{i,.}, \widetilde{\boldsymbol{X}}_{i',.}\right)$ est la même que la distance $d_{\boldsymbol{\omega}}\left(\boldsymbol{X}_{i,.}, \boldsymbol{X}_{i',.}\right)$ avec des distances d_1 pour chaque variable dont on précisera les poids.

Exemple

Pour les distances par variables, nous pouvons par exemple utiliser la distance **tout ou rien** pour les variables qualitatives ou la distance α-Höldérienne (voir l'exercice précédent) pour les variables quantitatives.

Chapitre 5

Véracité

"Un mensonge peut faire le tour de la terre le temps que la vérité mette ses chaussures."
Mark Twain.

Devant la grande quantité de données, il est plus difficile de détecter des valeurs erronées. Or, l'histoire a déjà montré qu'une erreur pouvait se propager longtemps avant d'être réfutée.

La (mauvaise) utilisation de la statistique à travers les âges

Un exemple classique d'erreur qui a encore la vie dure dans notre inconscient collectif est le fait que les épinards soient remplis de fer [1] :

- En 1870, le biochimiste allemand nommé E. von Wolf découvre que les épinards contiennent environ 2,7mg de fer pour 100g. L'histoire raconte que, quand sa secrétaire recopia la valeur, elle oublia la virgule ce qui multiplia par 10 la teneur.
- En 1881, un chercheur nommé Gustav von Bunge réévalue la teneur en fer mais se trompe entre le poids des épinards frais et des épinards déshydratés. Comme les épinards sont constitués à 90% d'eau, l'erreur était à nouveau de 1 pour 10.
- Dans les années 1930 à 1937, la communauté scientifique réévalua cette teneur et découvrit des deux erreurs mais elle ne réussit pas à convaincre le public. Il fallut attendre l'article de **?** de 1981, soit plus d'un siècle après, pour que les médias reprennent l'information.

5.1 Détection au stockage

Une première solution est de mettre des sécurités dans la récupération des données (voir par exemple la ressource *R1.04 - Statistique Descriptive 1*). Pour cela, il est important de correctement réfléchir aux limites mais aussi au fait que l'utilisateur aura peut-être une autre vision.

Exemple

Si nous souhaitons recueillir une note entre 0 et 5, nous pouvons stocker un réel ou réel positif. Toutefois, nous pouvons être plus précis·e en mettant 5 comme borne maximale. Enfin, si nous souhaitons une note entière, nous pouvons mettre une sécurité imposant une note dans l'ensemble $\{0, 1, 2, 3, 4, 5\}$. Cette dernière possibilité a l'intérêt de verrouiller complètement toutes les autres entrées ; néanmoins, si quelqu'un souhaite mettre des demis voir des quarts de point, ce n'est plus possible.

Ceci peut toutefois être compliqué pour des variables où l'une des bornes est compliquée à estimer. Par exemple, en France, la taille moyenne d'un homme en 2006 était de 1,75m et moins d'un pourcent dépassait les 2m [2]. Dans ce contexte, une borne à 2,20m ou 2,50m semble suffire. Néanmoins, nous savons que le plus grand homme de l'histoire moderne faisait 2,72m [3] et il aurait donc dépassé notre bornes.

1. Nous conseillons le site *Science & fourchette* recensant les nombreuses erreurs commises au fil des ans sur les épinards : http://sciencefourchette.com/2014/04/11/popeye-est-une-supercherie/ dont nous avons tiré une partie des informations.
2. Source : *Santé Publique France*. Voir l'url https://www.santepubliquefrance.fr/content/download/56011/file/etat-nutritionnel-adultes-anthropometrie-1.pdf
3. Source : Guiness Book. Voir l'url https://www.wetall.fr/qui-sont-les-plus-grands-hommes-de-tous-les-temps/

De plus, Hatton (2014) montre que la taille des hommes européens a augmenté d'environ 11cm durant le 20ème siècle ; par conséquent, une borne qui serait correcte à un instant donné pourrait être caduque ultérieurement.

La (mauvaise) utilisation de la statistique à travers les âges

Dans le logiciel OpenStreetMap, logiciel participatif, un immeuble de la ville *Fawker* s'est vu attribué 212 étages au lieu de 2 ce qui fait 49 étages de plus que le plus grand immeuble du monde (soit un peu plus de 30%). L'erreur fut découverte par des joueurs du jeu *Flight Simulator 2020* qui adaptait ces données pour générer le paysage [4].

Enfin, il est important de garder l'esprit ouvert vis-à-vis des possibles utilisations des utilisateur·trice·s ; plusieurs exemples montrent que l'utilisation n'est pas toujours celle qui est prévue (voir la figure humoristique 5.1).

FIGURE 5.1 – Image humoristique pour bien garder en mémoire que les utilisateur·trice·s ont parfois une façon différente d'utiliser une application que celles prévues par les développeur·euse·s [6].

La (mauvaise) utilisation de la statistique à travers les âges

Dans le but de dénoncer l'impact du numérique, l'artiste allemand Simon Weckert a récupéré 99 portables, les a programmés sur le même itinéraire avant de les promener dans une petite remorque. En remontant les rues *très lentement*, l'application a cru qu'il y avait un embouteillage [7]. Or, 99 portables dans une zone si réduite est compliquée ce qui aurait pu être mis en sécurité préalable (même si, ce type de vérifications demanderait certainement beaucoup de calculs inutiles pour *juste* contrer ce type d'initiatives).

5.2 Détection lors de l'exploration des données

Toutefois, ces sécurités ne sont pas toujours possibles voir, même si elles sont mises en place, ne garantissent pas que les données recueillies seront correctes. Voici quelques techniques pour repérer des erreurs lors de l'exploration des données.

5.2.1 Corrélations entre plusieurs variables

Dans certains cas, des données prises séparément peuvent être dans des valeurs correctes mais, prises ensembles, peuvent donner des comportements étranges. Par exemple, une personne ayant une taille de 2m est possible (même si on est dans un centile assez haut), une personne faisant 50 kg est aussi possible

4. L'article de *next* suivant résume l'anecdote : https://next.ink/5804/une-bourde-sur-openstreetmap-devient-immeuble-212-etages-sur-flight-simulator-2020/

6. Crédit image : Mathilde Bermond sur LinkedIn https://www.linkedin.com/posts/mathilde-uxdesigner_ux-designer-environnement-activity-6993467886509916160-lv0j/?trk=public_profile_like_view&originalSubdomain=fr

7. La vidéo est disponible par exemple sur Youtube : https://youtu.be/k5eL_al_m7Q?si=tF5EXEtZzBsqZyhX

(même si on est dans un centile assez bas) mais une personne faisant à la fois 2m et 50kg aurait un indice de masse corporelle (ou IMC) de 12,5 ce qui est très faible et place l'individu en dessous du quantile d'ordre 0.1%[8].

De manière générale, nous pouvons utiliser différents indicateurs corrélations comme ceux vus dans la ressource *R1.04 - Statistique Descriptive 1*, les méthodes factorielles (ressource *R4.02 - Méthodes factorielles*), les méthodes de classifications (ressources *R4.03 - Classification automatique* et *R5.02 - Data mining*) ou encore des méthodes estimant les liens entre différentes variables (ressources *R4.EMS.08 - Modèle linéaire* et *R5.VCOD.08 - Renforcement informatique 2*) et de chercher les individus qui ne sont pas cohérents.

5.2.2 K plus proches voisins

Pour repérer certaines erreurs, nous pouvons faire par rapprochement. Par exemple, en octobre 2023, il y eut beaucoup de pluies dans le sud ouest de la France. Sur la gauche de la figure 5.2, nous avons représenté une capture d'écran des quantités de pluie tombée en 72h (image issue de Météociel.fr). Nous pouvons remarquer que, dans la zone blanche, la plupart des stations dépassent les 150mm d'eau tandis que la station du Croix Millet (à droite sur la figure 5.2) n'a recensé aucune précipitation. Vue la position géographique de cette station, nous pouvons émettre un doute sur la valeur de cette donnée : est-ce qu'il y a eu un problème ? Est-ce que, comme la station est plus haute que ses voisines, il n'y a effectivement aucun nuage qui n'a atteint la station ?

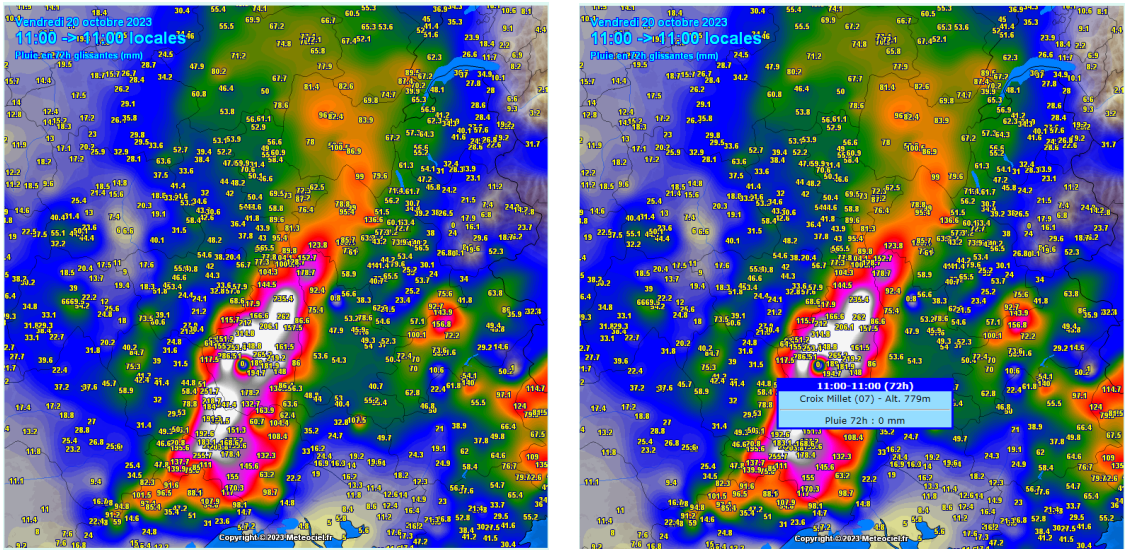


FIGURE 5.2 – Exemple de données où l'une des entrées semblent étranges compte-tenu des données spatialement voisines (à gauche) avec des informations plus précises sur la station suspecte (à droite)[10].

Cette façon de procéder peut être généralisée en utilisant les K plus proches voisins dans un espace de n'importe quelle dimension.

5.3 Données manquantes

Avec les grandes dimensions et les erreurs, nous pouvons être confrontés à des **données manquantes**. Avant toute chose, rappelons les différents types de données manquantes.

Définitions 6 (Données manquantes)

Parmi les **données manquantes**, il existe deux grande familles :

8. Calcul fait pour une personne de 19 ans sur le site https://www.msdmanuals.com/medical-calculators/BodyMassIndexBoys-fr.htm

10. Images issues du site internet `Météociel` : https://www.meteociel.fr/observations-meteo/rr.php?mode=72h&region=r11

- Les **données manquantes complètement aléatoirement** ou **données MCAR** (pour *Missing Completely At Random*) et les **données manquantes aléatoirement** ou **données MAR** (pour *Missing At Random*) sont des données dont la présence ou non dépend d'une loi de probabilité inconnue. La différence entre les données MCAR et MAR est que la probabilité des premières est indépendantes des individus et du type de variables tandis que les probabilités des secondes peuvent dépendre de paramètres propres aux variables.
- À l'opposé, si les données ne sont pas manquantes *par hasard*, on parle de **données manquantes non aléatoirement** ou **données MNAR** (pour *Missing Not At Random*) ou **données NMAR** (pour *Not Missing At Random*) suivant les auteurs. Notamment, ce type données inclut les **données censurées**.

Exemple

Pour comprendre la différence entre les trois types de données, prenons un exemple avec des capteurs défectueux :

- Si les capteurs ont tous le même défaut, les données seront manquantes avec la même probabilité quelque soit le capteur et nous aurons donc des données *MCAR*.
- Si le problème vient de la transmission et que la probabilité augmente avec le fait que le capteur est d'autant plus loin, nous avons une probabilité qui dépend de la localisation, nous sommes donc dans le cas de données *MAR*.
- Enfin, si le problème est une saturation des capteurs qui renvoient une valeur manquante dès que nous dépassons un seuil, nous sommes dans un cas de données *NMAR* car il y a une censure.

Dans la suite, nous voyons quelques propositions de ce que nous pouvons faire et ce qu'il vaut mieux éviter de faire.

5.3.1 Données manquantes aléatoirement et suppression d'individus

L'un des avantages de la très grande dimension est qu'il n'est pas nécessaire d'utiliser toutes les données pour avoir de bonnes estimations ou de bonnes prédictions. Dans le cas de données *MCAR* et sous la condition que la proportion de données manquantes n'est pas trop élevée, faire une analyse en enlevant les individus avec des données manquantes ne biaisera pas l'estimation.

Attention au piège

Ceci n'est absolument pas à faire dans le cas de données *NMAR*. Au contraire, dans l'exemple précédent, si nous faisons une moyenne en utilisant uniquement les valeurs *plus petite qu'un certain seuil*, la moyenne sera forcément biaisée vers le bas.

Dans le cas de données de *MAR*, il est possible de faire de même en donnant plus de poids aux observations où les probabilités d'être manquantes sont plus importantes (voir par exemple les théories de ré-échantillonnage vues dans la SAÉ *SAÉ 2.02 - Estimation par échantillonnage* ou les ressources *R2.06 - Probabilités 2* et *Ressource R2.08 : Statistique inférentielle* par exemple). Cela nécessite en amont d'estimer les probabilités d'avoir une donnée manquante.

5.3.2 Ne jamais remplacer par la moyenne

Lorsque les données sont numériques et pendant longtemps, les logiciels avaient tendance à remplacer par la moyenne de la variable. À part dans de rare cas, ce choix va biaiser les résultats (quelque soit le type de variables manquantes).

Exemple

Soit $\boldsymbol{Y}_n = (Y_1, \dots, Y_n)$ un n échantillon avec une loi ayant un moment d'ordre 2 dont $n-m$ valeurs sont manquantes avec $m \in \{1, \dots, n-1\}$. Comme nous avons des variables indépendantes et identiquement

distribuées, l'ordre des valeurs est arbitraire et nous supposons que les m premières valeurs sont celles qui sont connues et les $n-m$ suivantes sont manquantes. Nous étudions le vecteur

$$\widetilde{\boldsymbol{Y}}_n = \left(Y_1, \ldots, Y_m, \underbrace{\overline{\boldsymbol{Y}}_m, \ldots, \overline{\boldsymbol{Y}}_m}_{n-m \text{ fois}}\right) \text{ où } \overline{\boldsymbol{Y}}_m = \frac{1}{m}\sum_{i=1}^{m} Y_i. \tag{5.1}$$

Par linéarité de l'espérance, nous avons que $\mathbb{E}\left[\overline{\boldsymbol{Y}}_m\right] = \mathbb{E}\left[Y_1\right]$ c'est-à-dire le moment d'ordre 1 de la loi. Ainsi, nous avons :

$$\begin{aligned}
\widetilde{\overline{\boldsymbol{Y}}}_n &= \frac{1}{n}\left(\sum_{i=1}^{m} Y_i + \sum_{i=m+1}^{n} \overline{\boldsymbol{Y}}_m\right) \\
&= \frac{m}{n}\underbrace{\frac{1}{m}\sum_{i=1}^{m} Y_i}_{\overline{\boldsymbol{Y}}_m} + \frac{n-m}{n}\overline{\boldsymbol{Y}}_m \\
&= \frac{1}{n}\left[m\overline{\boldsymbol{Y}}_m + (n-m)\overline{\boldsymbol{Y}}_m\right] = \overline{\boldsymbol{Y}}_m.
\end{aligned}$$

Ainsi, la moyenne empirique basée sur le vecteur $\widetilde{\boldsymbol{Y}}_n$ de l'équation (5.1) est presque sûrement égale à la moyenne empirique basée sur les observations présentes dans le n échantillon $\boldsymbol{Y}_n$. De plus, la moyenne empirique basée sur l'échantillon $\boldsymbol{Y}_n$ étant un estimateur non biaisé, l'estimateur basé sur le vecteur $\widetilde{\boldsymbol{Y}}_m$ l'est aussi. En revanche, si nous prenons l'estimateur empirique de la variance, nous avons :

$$\begin{aligned}
\widetilde{\sigma}_n^2 &= \frac{1}{n}\left[\sum_{i=1}^{m}\left(Y_i - \widetilde{\overline{\boldsymbol{Y}}}_n\right)^2 + \sum_{i=m+1}^{n}\left(\overline{\boldsymbol{Y}}_m - \widetilde{\overline{\boldsymbol{Y}}}_n\right)^2\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{m}\left(Y_i - \overline{\boldsymbol{Y}}_m\right)^2 + \sum_{i=m+1}^{n}\underbrace{\left(\overline{\boldsymbol{Y}}_m - \overline{\boldsymbol{Y}}_m\right)^2}_{=0}\right] \text{ car } \widetilde{\overline{\boldsymbol{Y}}}_n = \overline{\boldsymbol{Y}}_m \text{ p.s} \\
&= \frac{m}{n}\frac{1}{m}\sum_{i=1}^{m}\left(Y_i - \overline{\boldsymbol{Y}}_m\right)^2 = \underbrace{\frac{m}{n}}_{<1}\sigma_m^2
\end{aligned}$$

donc l'estimateur de la variance basé sur le vecteur $\widetilde{\boldsymbol{Y}}_n$ de l'équation (5.1) est presque sûrement strictement plus petit que l'estimateur basé uniquement sur les valeurs connues du n échantillon $\boldsymbol{Y}_n$.

5.3.3 Garder en tête l'objectif final

De manière générale, si nous souhaitons remplacer des valeurs manquantes par des valeurs estimées ou simulées, il est important de garder l'objectif en tête. Adaptons l'exemple proposé dans la thèse de Audigier (2015) (voir la figure 5.3) :

1. Simulation de 500 observations indépendantes et de même loi $\mathcal{U}\left([-5,5]\right)$.
2. Simulation de 450 observations indépendantes suivant la loi :

$$Y_i = X_i + \varepsilon_i \text{ où } \varepsilon \overset{\text{iid}}{\sim} \mathcal{N}\left(0,1\right). \tag{5.2}$$

3. Les 50 dernières données manquantes sont estimées par :
   - La moyenne $\overline{\boldsymbol{Y}}$ des valeurs des $\boldsymbol{Y}$ estimées.
   - Une projection par une estimation des paramètres suivant le modèle suivant :

$$Y_i = \alpha X_i + \beta + \varepsilon_i \text{ où } \varepsilon \overset{\text{iid}}{\sim} \mathcal{N}\left(0,\sigma^2\right) \tag{5.3}$$

avec α, β et σ^2 estimés par moindre carré (voir les ressources *R2.05 - Statistique descriptive 2*, *R4.EMS.08 - Modèle linéaire* et *R5.VCOD.08 - Renforcement informatique 2*). La projection se fait de la façon suivante :

$$Y_i = \widehat{\alpha} X_i + \widehat{\beta}. \tag{5.4}$$

- Une simulation des données suivant le modèle de l'équation (5.3) :

$$Y_i = \widehat{\alpha} X_i + \widehat{\beta} + \varepsilon_i \text{ où } \varepsilon \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \widehat{\sigma}^2\right). \tag{5.5}$$

Remarque

Notons que nous sommes dans un cas de données *MAR* car seules les données suivant Y sont manquantes mais elles le sont avec la même probabilité (indépendamment de la valeur de X associées ou de la valeur possible de Y).

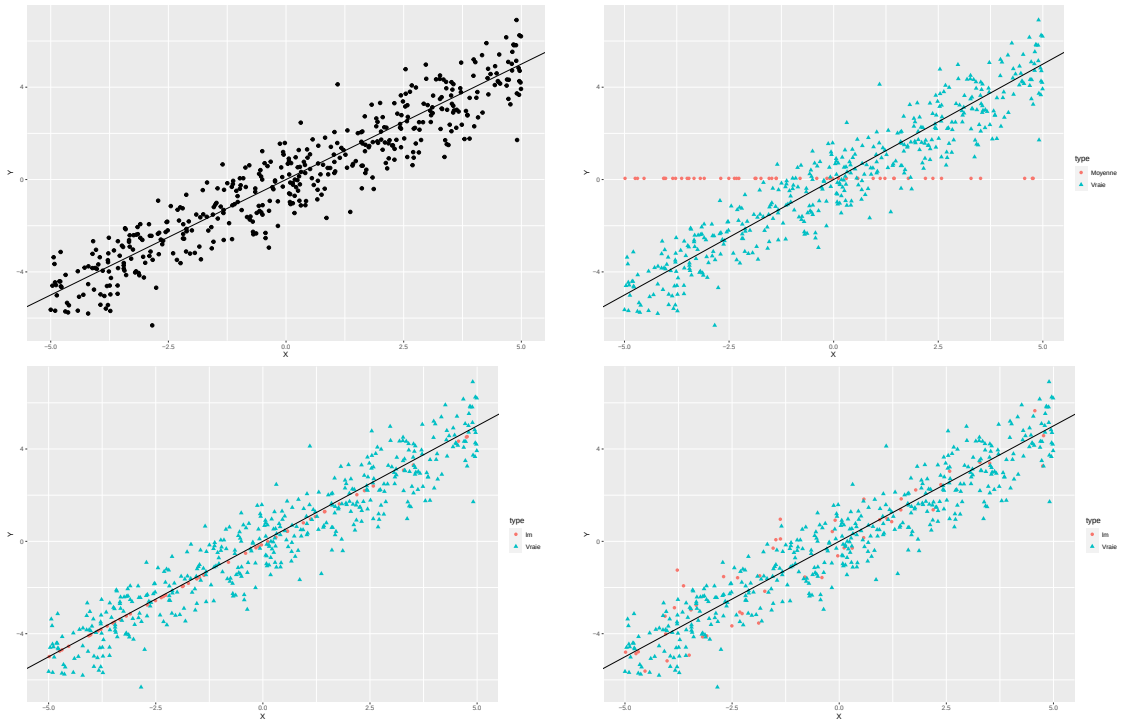


FIGURE 5.3 – Représentation des données simulées suivant le modèle de l'équation (5.2) (en haut à gauche), les points en remplaçant les données manquantes par la moyenne (en haut à droite), les points en remplaçant les données manquantes par une estimation des paramètres de l'équation (5.4) (en bas à gauche) et suivant une simulation des donnes suivant l'équation (5.5) (en bas à droite). Le trait noir correspond à la droite d'équation $y = x$.

Sur la figure 5.3, nous pouvons voir en haut à droite que le fait de remplacer par la moyenne casse la structure de dépendance entre les deux variables (les points en rouge ne semblent pas suivre la logique des triangles bleus). Si nous remplaçons les valeurs manquantes par une projection suivant le modèle (5.4) (en bas à gauche), nous voyons que la structure semble conservée mais les points sont proches de la droite d'équation $y = x$; comme pour la partie 5.3.2, si nous faisons une estimation basée sur le modèle (5.3), l'estimation de la variance sera sous estimée. Pour corriger cela, nous pouvons simuler nos données suivant le modèle (5.5) (en bas à droite de la figure) mais, dans ce cas, nous introduisons de la variabilité dans nos estimations.

Codage en R

Pour simuler les données de la figure 5.3, nous avons utilisé le code suivant :

```
1 ##############
2 # Simulation
```

```
############
n<-500
X<-runif(n)*10-5
Y<-X+rnorm(n)
Y[1:50]<-NA

library(ggplot2)
#######
# Base
#######
dat<-data.frame(X=X,Y=Y)
ggplot(dat,aes(x=X,y=Y))+geom_point(cex=2)+geom_abline(slope=1)
#######
# Completion mean
#######
Y[1:50]<-mean(Y,na.rm=TRUE)
dat<-data.frame(X=X,Y=Y,type=factor(c(rep("Moyenne",50),rep("Vraie",450))))
ggplot(dat,aes(x=X,y=Y,pch=type,col=type))+geom_point(cex=2)+geom_abline(slope=1)
#######
# Completion lm
#######
lm<-lm(Y~X,data = dat[51:500,])
Y[1:50]<-lm$coefficients[1]+lm$coefficients[2]*X[1:50]
dat<-data.frame(X=X,Y=Y,type=factor(c(rep("lm",50),rep("Vraie",450))))
ggplot(dat,aes(x=X,y=Y,pch=type,col=type))+geom_point(cex=2)+geom_abline(slope=1)
#######
# Simulation + lm
#######
lm<-lm(Y~X,data = dat[51:500,])
Y[1:50]<-lm$coefficients[1]+lm$coefficients[2]*X[1:50]+rnorm(50,sd = sqrt(mean(lm$residuals^2)))
dat<-data.frame(X=X,Y=Y,type=factor(c(rep("lm",50),rep("Vraie",450))))
ggplot(dat,aes(x=X,y=Y,pch=type,col=type))+geom_point(cex=2)+geom_abline(slope=1)
```

Ce principe peut être généralisé à tout objectif. Par exemple, dans le cas d'une ACP (Analyse en Composante Principale), s'il y a des données manquantes et que nous souhaitons conserver l'objectif de mettre en évidence des liens linéaires entre les variables, il existe des solutions proposées dans le package `missMDA` du logiciel R proposé par Josse et Husson (2012). Notamment, il existe deux techniques reprenant les mêmes idées :

- par imputation (Josse et Husson (2012)),
- par simulation (Audigier (2015)).

Ou encore, plus récemment, Audigier et Niang (2023) propose des méthodes d'imputations pour le cadre de la classification ou, pour une version française, la communication dans les *Journées de Statistique* de Audigier et Zouleya (2023) [11]. Comme nous pouvons le voir, la recherche sur ce domaine est (très) récente et il est donc nécessaire de se former régulièrement.

11. Voir l'acte de conférence sur l'url https://drive.google.com/file/d/1gStcIkGcCOD4ouFsUzOdGQAk_84K2NKX/view

Chapitre 6

Valeur

"On ne mesure la vraie valeur des choses que lorsqu'on en est privé."
Olivier Salesse

Dans cette partie, nous discuterons de stocker et/ou de n'utiliser que les variables les plus pertinentes.

6.1 Protocole intelligent

En écho avec les chapitres 2 et 3 se pose la question de l'importance des variables dans l'objectif final. Avec la montée en puissance des méthodes de stockage et d'analyse, le désir de tout conserver dans le but de voir après ce que nous analyserons est grand. Néanmoins, cela peut tout de même poser des problèmes de stockage (voir le chapitre 2) et de temps de calculs (voir le chapitre 3). A ceci s'ajoute des problèmes dans les analyses où certains algorithmes vont reproduire les biais (par exemple, il y a les recherches récentes dans la science du recrutement ; voir par exemple Brault et al. (2023)).

La (mauvaise) utilisation de la statistique à travers les âges

Dans leur article, Datta et al. (2015) ont simulé 1000 utilisateurs et utilisatrices et ont regardé comment les algorithmes, notamment ceux de Google, réagissaient sur la publicité proposée. La conclusion pour les offres de poste met en évidence que les profils masculins ont eu 1800 offres d'emploi à haut salaire contre 300 pour les profils féminins [1]. Le choix de mettre le *genre* des personnes pour mieux cibler les publicités a créé une amplification du biais.

Cette réflexion sur le stockage, en amont, des données doit aussi se placer dans le respect du *RGPD*.

6.2 Sélection des variables

À l'opposé, lorsque nous avons récupéré un certains nombres de données, il est important de se demander si nous devons conserver toutes les données dans le cadre de notre objectif. Dans la suite, nous allons principalement observer deux cas de figures :

- Dans le premier, nous regardons le cas où nous avons plus d'individus que de paramètres à estimer et nous discuterons donc du problème de sélection de modèles.
- Dans un deuxième temps, nous étudierons le cas où il y a plus de variables que d'individus (cas mentionné dans la section 1.5.3) et de comment continuer les estimations.

6.2.1 Sélection de modèle

Durant le BUT, nous nous sommes intéressés au problème de la sélection de modèle et de variables (voir par exemple les ressources *R4.EMS.08 - Modèle linéaire*, *R5.02 - Data mining*, *R5.EMS.06 - Modélisation statistique avancée* et *R5.VCOD.08 - Renforcement informatique 2* ou les *SAÉ2.06 - Analyse de données, reporting et datavisualisation*, *SAÉ4.EMS.01 - Expliquer ou prédire une variable quantitative à partir de*

1. Voir par exemple la vidéo #DATAGUEULE 84 pour une présentation vulgarisée : https://youtu.be/oJHfUv9RIY0?si=kNlHuGSOV2Yvvq08

plusieurs facteurs et *SAÉ5.03 - Mise en oeuvre d'un processus de Datamining*). Principalement, deux types de méthodes ont été abordés :

- Les tests de nullité des paramètres (voir les ressources *R3.06 - Tests d'hypothèses pour l'analyse bivariée*, *R4.EMS.08 - Modèle linéaire* et *R5.VCOD.08 - Renforcement informatique 2*). Comme nous l'avons vu en section 1.5.1, il y a un risque que ces tests rejetteront systématiquement l'hypothèse de nullité des paramètres.

- La sélection de modèle notamment avec :

  – Des critères de sélection de modèle (voir les ressources *R4.EMS.08 - Modèle linéaire* et *R5.EMS.06 - Modélisation statistique avancée* et la *SAÉ4.EMS.01 - Expliquer ou prédire une variable quantitative à partir de plusieurs facteurs*) comme le critère *AIC* (pour *Akaike Information Criterion*) ou le critère *BIC* (pour *Bayesian Information Criterion*) qui vont venir pénaliser le critère.

  – De la validation croisée (voir la ressource *R5.02 - Data mining* notamment) qui va découper l'échantillon en deux groupes pour estimer les paramètres sur une partie et tester sur l'autre partie.

Les critères de sélection de modèle n'étant pas enseignés aux deux parcours mais étant souvent présent dans les sorties de certains algorithmes, je n'insisterai que sur un point : à **chaque objectif correspond un critère**. Par exemple, nous avons :

- Si l'objectif est l'**estimation** des paramètres, notamment le nombre de groupe dans une méthode de classification comme les K-means (voir la ressource *R4.03 - Classification automatique*), il est préférable d'utiliser le **critère *BIC*** (pour *Bayesian Information Criterion*) dont la formule est :

$$BIC(\mathcal{M}) = f(\mathcal{M}) + \frac{|\mathcal{M}|}{2}\log n \tag{6.1}$$

  avec $\mathcal{M}$ le modèle tester (par exemple, pour les K-means, le modèle à 2 groupes ou 3 groupes...), $|\mathcal{M}|$ le nombre de paramètres estimés par le modèle, n le nombre d'individus et $f(\mathcal{M})$ est une fonction du modèle en lien avec l'objectif comme, par exemple, les **moindes carrés** (voir la ressource *R2.05 - Statistique descriptive 2*) ou l'opposé du maximum de vraisemblance (voir la ressource *R5.EMS.06 - Modélisation statistique avancée*).

- Si l'objectif est plutôt la **prédiction**, c'est-à-dire prédire le comportement d'une nouvelle observation (voir par exemple la ressource *R4.03 - Classification automatique* ou la *SAÉ 3.03 - Description et prévision de données temporelles*), il vaut mieux utiliser le **critère *AIC*** (pour *Akaike Information Criterion*) dont la formule est :

$$BIC(\mathcal{M}) = f(\mathcal{M}) + 2|\mathcal{M}| \tag{6.2}$$

  avec les mêmes remarques que pour le critère *BIC* de l'équation (6.1).

Attention au piège

Nous pourrions penser que les objectifs d'**estimation** et de **prédiction** sont les mêmes puisqu'il s'agit de trouver le *bon* modèle. Néanmoins, comme vous l'avez vu en ressource *R4.03 - Classification automatique*, ces objectifs ne sont pas tout à fait les mêmes. De plus, il faut se rappeler qu'une modélisation, aussi efficace soit elle, reste imparfaite et trouver le modèle *choisi* par la nature pour simuler les données est compliqué.

Remarque

Sur les équations (6.1) et (6.2), nous voyons que seules les pénalisations divergent. Pour le critère *BIC* (équation (6.1)), nous avons une pénalisation qui dépend à la fois de la complexité du modèle et du nombre d'individus tandis que pour le critère *AIC* (équation (6.2)), seule la complexité du modèle intervient. En pratique, le critère *AIC* va avoir tendance à conserver des modèles plus complexes que le critère *BIC* lorsqu'il y a un très grand nombre d'individus.

6.2.2 Sélection de variables que $p > n$

Enfin, nous avons vu dans la section 1.5.3 qu'un trop grand nombre de variables par rapport au nombre d'individus impliquait une mauvaise estimation des paramètres ; ce problème se rencontre notamment en biologie avec le séquençage des génomes permettant de les *découper* en tout petit bout pour un individu donné. Pour contrer ce problème, il existe une méthode appelée *Least Absolute Shrinkage and Selection Operator* (ou plus simplement *LASSO*) introduite par Tibshirani (1996) dont nous expliquons le principe dans un cas simple :

Définitions 7 (Méthode *LASSO*)

Étant donnés une matrice $\boldsymbol{X} \in \mathcal{M}_{n\times p}(\mathbb{R})$, un vecteur $\boldsymbol{\beta} \in \mathbb{R}^p$ **sparse**, c'est-à-dire avec un *grand* nombre de valeurs nulles, et des observations $\boldsymbol{Y}_n$ vérifiant le modèle suivant :

$$\boldsymbol{Y}_n = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{6.3}$$

avec $\boldsymbol{\varepsilon}$ un n échantillon de loi gaussienne centrée et de variance σ^2 alors la méthode *LASSO* (pour *Least Absolute Shrinkage and Selection Operator*) prend un **paramètre de régularisation** $\lambda > 0$ et cherche l'estimateur $\widehat{\boldsymbol{\beta}}_\lambda$ de $\boldsymbol{\beta}$ vérifiant :

$$\widehat{\boldsymbol{\beta}}_\lambda \in \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \|\boldsymbol{Y}_n - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\} \tag{6.4}$$

où $\|\cdot\|_2$ est la **norme euclidienne** et $\|\cdot\|_1$ est la **norme absolue** définies pour tout $\boldsymbol{y} \in \mathbb{R}^n$:

$$\|\boldsymbol{y}\|_2 = \sqrt{\sum_{i=1}^n y_i^2} \text{ et } \|\boldsymbol{y}\|_1 = \sum_{i=1}^n |y_i| . \tag{6.5}$$

Remarque

Le modèle de l'équation (6.3) est le même que le modèle linéaire vu en ressources *R4.EMS.08 - Modèle linéaire* et *R5.VCOD.08 - Renforcement informatique 2* sauf que nous faisons l'hypothèse que le vecteur $\boldsymbol{\beta}$ est sparse.

Remarque

La résolution dans l'équation (6.4) est une minimisation par les moindres carrés (voir la ressource *R2.05 : Statistique descriptive 2*) dont nous pénalisons les coordonnées du vecteur $\boldsymbol{\beta}$ qui sont trop élevées. Ainsi, nous pouvons voir ce paramètre comme un coût que nous devons payer d'autant plus élevé que λ est grand et/ou que nous avons besoin de valeurs élevées pour le vecteur $\widehat{\boldsymbol{\beta}}_\lambda$. En particulier :

- Si le paramètre λ est proche de 0 alors prendre des valeurs élevées pour $\widehat{\boldsymbol{\beta}}_\lambda$ ne coûtera presque rien et l'algorithme va donc avoir tendance à faire du sur-apprentissage en estimant les fluctuations dues au bruit.
- Si le paramètre λ tend vers l'infini alors la moindre valeur non nulle va coûter trop cher et, à partir d'un moment, le meilleur paramètre $\widehat{\boldsymbol{\beta}}_\lambda$ sera le vecteur nul.

L'un des enjeux de la modélisation *LASSO* est donc de choisir correctement le *bon* paramètre λ ; il existe différentes méthodes notamment une basée sur la validation croisée (voir la ressource *R5.02 - Data mining*) qui est généralement implémentée de base.

La modélisation *LASSO* possède plein d'extensions (Elastic-Net de Zou et Hastie (2005), Fused-lasso de Tibshirani et al. (2005) ou Group-Lasso de Yuan et Lin (2006) pour ne citer qu'elles) et une grande communauté de recherche qui travaille encore à optimiser les méthodes et l'estimation ; l'étude des méthodes *LASSO* est souvent un cours en lui même en master de statistique et en science des données. Dans ce cours, nous nous contenterons d'appliquer les fonctions comme des boîtes noires et de comprendre les sorties.

Codage en R

Pour montrer une application, nous reprenons les données `spotify` du TP1 après le traitement du TP3 donc un individu statistique est une chanson (unicité par l'id ; soient 28 356 individus) et nous avons les données originales ainsi que le nombre de fois où la chanson apparaît dans une playlist (pour un total de 458 si nous ne prenons que les variables numériques). Le nom du tableau est `Tab_final`. Le but est d'essayer d'expliquer la popularité (variable `track_popularity` allant de 0 à 100) en fonction des autres variables. Pour utiliser la procédure *LASSO*, nous devons trouver un λ cohérent. Pour cela, le package `glmnet` propose une procédure basée sur la validation croisée :

```
### Chargement
library(glmnet)
### Lambda par crossvalidation
lambda<-cv.glmnet(as.matrix(Tab_final[,-1]), Tab_final$track_popularity)
plot(lambda)
```

La sortie graphique est mise sur la figure 6.1. Nous pouvons voir que pour des λ petits (à gauche de la figure), nous avons quasiment la même erreur de prédiction jusqu'à environ $\lambda \approx e^{-3}$ puis va augmenter d'abord doucement puis plus rapidement sous forme de sigmoïde. La méthode propose de prendre $\lambda_{\min} \approx e^{-4.128335}$ qui minimise l'erreur de prédiction (ligne verticale en pointillés de gauche sur la figure 6.1) et $\lambda_{1se} \approx e^{-2.26766}$ pour 1 déviation standard (ligne verticale en pointillés de droite sur la figure 6.1). Ensuite, nous pouvons nous intéresser aux variables conservées :

```
### Estimation du modèle
lasso<-glmnet(as.matrix(Tab_final[,-1]), Tab_final$track_popularity,lambda = lambda$lambda.1se)
### Nombre de variables non nulles
mean(lasso$beta>0)
```

Pour l'estimation de λ_{1se} la moins conservatrice, nous obtenons environ 56% des variables qui sont *actives* c'est à dire qu'on peut considérer comme non nulles.

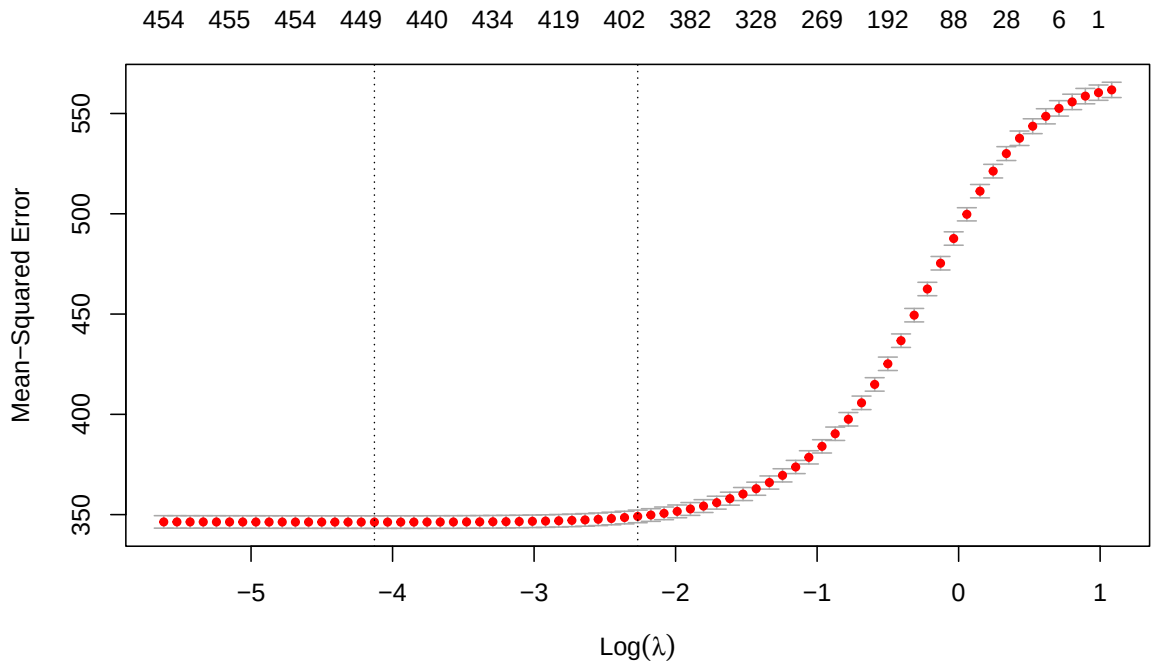


FIGURE 6.1 – Représentation de la sortie `plot(lambda)` de l'exemple de `spotify` : chaque intervalle est une estimation de l'erreur suivant la valeur de $\log(\lambda)$ et les lignes verticales en pointillés correspondent aux deux propositions de choix de λ proposés par le package `glmnet`. Les nombres au dessus correspondent aux nombres de variables actives.

Chapitre 7

Visualisation

"Le point de vue le plus simple est toujours le meilleur."
Charlie Chaplin

Enfin, le dernier V concerne la visualisation. Cette problématique est déjà compliquée lorsque nous avons trois variables et n'a pas de solution complète au delà.

7.1 Projection

La première solution est de projeter les variables. Par exemple, nous pouvons représenter des croisements de deux ou trois variables. Toutefois, les méthodes les plus adaptées sont celles vues dans la ressource *R4.02 - Méthodes factorielles*à savoir l'**Analyse en Composante Principale** (ou **ACP**) et l'**Analyse Factorielle des Correspondances** (ou **AFC**) ou l'**Analyse des Correspondances Multiples** (ou **ACM**). Ces méthodes recherchent le plan permettant au mieux de projeter les variables et individus afin de les séparer.

Attention au piège

Attention toutefois, cette projection peut quand même perturber dans la mesure où nous projetons les points. Bien que ce soit la projection qui *écarte* le mieux les points, cela ne veut pas dire que deux points proches sur le plan de projection le sont suivant toutes les dimensions [1]

7.2 Classification

Une autre représentation possible est celle qui permet de classifier les individus. Pour cela, nous pouvons penser par exemple aux méthodes vues en ressource *R4.03 - Classification automatique* comme la Classification Ascendante Hiérarchique (CAH).

7.3 Visualisation dynamique et tableau de bord

Pour contourner les problèmes de représentations, il peut être intéressant de proposer des représentations dynamiques. Pour cela, nous pouvons utiliser la librairie `shiny` apprise en ressource *R2.01 - Reporting et Datavisualisation.*

Exemple

Des entreprises se sont spécialisées dans l'exploration des grandes bases de données. A ce titre, nous pouvons par exemple citer l'entreprise *Orange Data Mining* qui propose un outil gratuit :

https://orangedatamining.com/

1. Voir par exemple la vidéo suivante d'un même nuage de points dont le changement de perspective ne donne pas les mêmes sentiments de sitance : https://youtu.be/3FAlxOMxfSw?si=6tY_VQb7OoDCws2Q

Chapitre 8

Aide mémoire en Python

Dans cette annexe, nous remettons des commandes de bases en Python nécessaires à la survie.

8.1 Manipulation pour les environnements de travail

Pour manipuler les environnements, la bibliothèque `os` permet d'interagir avec le système d'exploitation :

```
import os
```

Notamment, il est possible de savoir à tout moment où on se trouve à l'aide de la fonction `getcwd` :

```
os.getcwd()
```

Pour changer l'environnement de travail, c'est la fonction `chdir` :

```
# Si pwd est une variable str contenant le chemin
os.chdir(pwd)
```

8.2 Barre de progression

Lorsque les calculs sont longs, il peut être intéressant de savoir où en est le code. Pour cela, et si vous avez un boucle `for`, il peut être intéressant d'ajouter une barre de progression. Ceci peut se faire avec la fonction `tqdm` de la bibliothèque du même nom :

```
from tqdm import tqdm
j = 0
for i in tqdm(range(10**8)):
    j += i
```

8.3 Recherche de valeurs particulières

Pour rechercher des valeurs particulières, nous savons qu'il est possible d'utiliser l'opérateur `"=="` qui nous renvoie des réponses logiques `True` et `False`. Souvent, c'est la position qui nous intéresse. Dans ce cas, nous pouvons utiliser la fonction `flatnonzero` de la bibliothèque `numpy`. Par exemple, pour chercher les positions des valeurs manquantes d'une colonne d'un tableau `Tableau` de type `DataFrame`, nous avons le code suivant :

```
import pandas as pd
import numpy as np
np.flatnonzero(pd.isna(Tableau["Nom colonne"]))
```

8.4 Enlever une valeur

Dans Python, la valeur `None` (sans guillemets) représente une valeur manquante. En remplaçant une valeur par `None`, on peut généralement continuer à utiliser les autres fonctions sans incidences.

Bibliographie

V. Audigier. Imputation multiple par analyse factorielle : Une nouvelle méthodologie pour traiter les données manquantes. PhD thesis, Rennes, Agrocampus Ouest, 2015.

V. Audigier et N. Niang. Clustering with missing data : which equivalent for rubin's rules ? Advances in Data Analysis and Classification, 17(3) :623–657, 2023. URL https://drive.google.com/file/d/1gStcIkGcC0D4ouFsUzOdGQAk_84K2NKX/view.

V. Audigier et F. S. Zouleya. Clustering sur données incomplètes : méthodes directes ou imputation multiple ? Dans Les 54èmes Journées de Statistique, 2023.

J. Bennett et S. Lanning. The netflix prize. Dans Proceedings of KDD cup and workshop, volume 2007, page 35, 2007.

P. Besse. Analyse en composantes principales (acp). Wikistat Toulouse, 2022. URL http://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-explo-acp.

V. Brault, J. Chiquet, et C. Lévy-Leduc. Efficient block boundaries estimation in block-wise constant matrices : An application to hic data. Electron. J. Statist., 11(1) :1570–1599, 2017. ISSN 1935-7524. doi : 10.1214/17-EJS1270.

V. Brault, A. Lacroux, P. Le Gall, C. Martin-Lacroux, A. Saillet, et S. Wang. Modélisation du biais dans les recrutements : étude de l'influence d'un biais dans les données d'apprentissage de différentes procédures. Dans 54èmes Journées de Statistique de la SFdS, 2023.

V. Cisco. Cisco visual networking index : Forecast and trends, 2017–2022. White paper, 1(1), 2018.

A. Datta, M. C. Tschantz, et A. Datta. Automated experiments on ad privacy settings : A tale of opacity, choice, and discrimination. Dans Privacy Enhancing Technologies, volume 1, page 92–112, 2015.

T. J. Hatton. How have europeans grown so tall ? Oxford Economic Papers, 66(2) :349–372, 2014.

M. Hilbert et P. López. The world's technological capacity to store, communicate, and compute information. science, 332(6025) :60–65, 2011.

F. Husson, S. Lê, et J. Pagès. Analyse de données avec R. Presses universitaires de Rennes, 2016.

J. Josse et F. Husson. Handling missing values in exploratory multivariate data analysis methods. Journal de la Société Française de Statistique, 153(2) :79–99, 2012.

D. Laney et al. 3d data management : Controlling data volume, velocity and variety. META group research note, 6(70) :1, 2001.

V. Mayer-Schönberger et K. Cukier. Big data : A revolution that will transform how we live, work, and think. Houghton Mifflin Harcourt, 2013.

T. S. Project et M. Efoui-Hess. Climat : l'insoutenanble usage de la vidéo en ligne : un cas pratique pour la sobriété numérique. Shift Project, 2019. URL https://theshiftproject.org/wp-content/uploads/2019/07/2019-01.pdf.

I. Sandvine. Global internet phenomena report. North America and Latin America, 2016.

É. Schultz et M. Bussonnier. Python pour les SHS. Introduction à la programmation pour le traitement de données. Pratique de la statistique. Presses universitaires de Rennes, 2021.

C. Thomas H, E. Charles, R. Ronald L, S. Clifford, et al. Introduction to algorithms third edition, 2009.

R. Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B : Statistical Methodology, 58(1) :267–288, 1996.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, et K. Knight. Sparsity and smoothness via the fused lasso. Journal of the Royal Statistical Society Series B : Statistical Methodology, 67(1) :91–108, 2005.

M. Yuan et Y. Lin. Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society Series B : Statistical Methodology, 68(1) :49–67, 2006.

H. Zou et T. Hastie. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society Series B : Statistical Methodology, 67(2) :301–320, 2005.

Table des figures

Liste des tableaux

Index