REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Abou Bekr Belkaid de Tlemcen
Faculté de Technologie
Département de Génie Biomédical
Laboratoire de Génie Biomédical

Fouille de données

Polycopié de Travaux pratiques

Master en Informatique Biomédicale

1^{ere} Année

SETTOUTI Nesma
EL HABIB DAHO Mostafa

Couriel : nesma.settouti@gmail.com

Année Universitaire: 2017-2018

Avant-propos

Le désir ainsi que le besoin de la connaissance ont conduit au développement de systèmes et d'équipements qui peuvent générer et collecter des quantités massives de données. De nombreux domaines, en particulier ceux impliqués dans la prise de décision, traitent la problématique de l'acquisition des connaissances. nous citons en particulier le cas des données médicales et biologiques. Les progrès de la capacité de stockage et des équipements de collecte de données numériques tels que les scanners ont permis de générer des ensembles de données massives, parfois appelés bases de données, qui se mesurent en téraoctets.

Notre capacité à parcourir les données ainsi que la recherche de la donnée pertinente pour extraire la connaissance ciblée est entravée généralement par la taille et la complexité de la base de données stockée. En effet, la taille des données rend l'analyse humaine difficile dans de nombreux cas, en annulant l'effort déployé lors de la collecte. Il existe plusieurs options viables actuellement utilisées pour aider à éliminer les informations bruitées. Le processus de récupération des données utilisant différents outils est appelé l'extraction des connaissances à partir des données ECD ou (KDD : Knowledge Discovery from Data en anglais).

Le procédé de fouille de données « data mining » est l'un des maillons de la chaîne de traitement pour la découverte des connaissances à partir des données. Il est l'art d'extraire des connaissances à partir des données en faisant intervenir des méthodes et des outils issus de différents domaines de l'informatique, de la statistique ou de l'intelligence artificielle en vue de découvrir des connaissances utiles.

Ce polycopié de travaux pratiques est destiné à des étudiants de master informatique biomédicale. Les objectifs du TP ciblent en particulier la compréhension des principaux algorithmes de fouille de données : le clustering par les k-moyennes, par classification hiérarchique ascendante ; l'algorithme Apriori pour les règles d'association ; les algorithmes de classification supervisée par arbres de décision et leurs extensions aux méthodes d'ensemble (les forêts aléatoires).

Ce document est centré autour de l'utilisation des techniques d'apprentissage artificiel pour l'analyse et l'exploration de données afin d'en comprendre le sens, de déceler des relations entre des évènements, d'en déduire des modèles de comportement. L'accent étant aussi porté sur le choix des paramètres, la complexité des algorithmes et l'évalua-

tion des performances pour un bon choix de méthodes face à une étude de cas.

Tous les TPs sont réalisés avec le logiciel libre et évolutif R. R est un logiciel de statistique et de data mining, piloté en ligne de commande. Il est extensible (quasiment) à l'infini via le système des packages. Sa facilité de programmation et sa forte utilisation : en enseignement, en recherche et dans le monde de l'entreprise, en font de lui un langage omniprésent et indispensable pendant le formation des étudiants de master en informatique biomédicale.

Mme. Settouti Nesma
&
Mr. El Habib Daho Mostafa

Table des matières

0 TP 0 : Notions de bases du Langage R 1
0.1 Présentation du langage et logiciel R 1
0.2 Installation et description du Logiciel R 2
0.3 Fonctionnement de R 6
0.4 Applications : premiers pas avec R 7
0.4.1 Calcul élémentaires 7
0.4.2 Scalaires, vecteurs et matrices 7
0.4.3 Les fonctions 8
0.4.4 Structure des données 8
0.4.5 Les graphiques 9
0.4.6 Importer et exporter des fichiers de données 9
0.4.7 Éléments de programmation 11
0.5 Applications 12

1 TP1 : Pré-traitement des données : les méthodes de visualisation et de description 15
1.1 Objectif 15
1.2 Importation de la base de données 15
1.3 Manipulation des données 16
1.3.1 Aperçu des données 16
1.3.2 Transformation de variables quantitatives / qualitatives 16
1.3.3 Changement de codage des variables 17
1.3.4 Visualisation des données 17
1.3.5 Conditions et restrictions 19
1.3.6 Les changements de valeurs 20
1.3.7 Traitement des données manquantes 20
1.3.8 Tri des données 21
1.4 Réduction de dimension 21
1.4.1 La transformation de variables (feature extraction) 22
1.4.2 La sélection de variables (Feature selection) 27
1.5 Travail demandé 29

2 TP2 : Les méthodes de structuration et de classification en apprentissage non supervisé 31
2.1 Objectif 31
2.2 L'algorithme des K-moyennes 31
2.3 Classification hiérarchique 36
2.3.1 Le regroupement agglomératif 36
2.3.2 Le regroupement divisif 36

2.3.3 Le clustering hiérarchique agglomératif . . . 37
2.3.4 Le clustering hiérarchique divisif . . . 38
2.3.5 Analyse des dendrogrammes . . . 39
2.4 Travail demandé . . . 41

3 TP3 : Les méthodes de structuration et de classification en apprentissage supervisé . . . 43
3.1 Objectif . . . 43
3.2 L'arbre de décision CART (Classification And Regression Tree) . . . 44
3.2.1 Echantillonnage : Apprentissage vs Test . . . 44
3.2.2 Construction de l'arbre de decision . . . 44
3.2.3 Élagage de l'arbre de décision . . . 46
3.2.4 Évaluation des performances . . . 47
3.2.5 Règles de classification . . . 48
3.3 Forêts aléatoires . . . 49
3.3.1 Principe de Boostrapping . . . 49
3.3.2 Principe du Bagging . . . 49
3.3.3 Construction de la forêt aléatoire . . . 49
3.3.4 La mesure des variables d'importance . . . 52
3.3.5 La forêt aléatoire pour le traitement des données manquantes . . . 53
3.4 Travail demandé . . . 54

4 TP4 : Les méthodes descriptives et prédictives . . . 55
4.1 Objectif . . . 55
4.2 Les règles d'association . . . 56
4.2.1 L'Algorithme Apriori . . . 56
4.2.2 Construction des règles d'association . . . 57
4.3 Régression linéaire simple et multiple . . . 60
4.3.1 Préparation de la base de données . . . 61
4.3.2 Régression linéaire simple . . . 61
4.3.3 La régression linéaire multiple . . . 63
4.4 Travail demandé . . . 64
4.4.1 Exercice d'application des règles d'association . . . 64
4.4.2 Exercice d'application de la régression linéaire simple et multiple . . . 64

Conclusion . . . 67

ANNEXE A : Résumé des principales fonctions R . . . 69

ANNEXE B : Le jeu de données «satisfaction_hopital» . . . 73

ANNEXE C : Le jeu de données « Heart disease » . . . 75

ANNEXE D : Le jeu de données « Lenses » . . . 77

ANNEXE E : Le jeu de données « Poids de naissance » . . . 79

Références Bibliographiques . . . 81

0

TP 0 : Notions de bases du Langage R

Introduction

Lors du processus d'apprentissage, il est essentiel que le langage de programmation ne soit pas une entrave aux étudiants. Ainsi pouvoir se focaliser pleinement sur ce que l'on doit apprendre sans devoir se concentrer pour déchiffrer un langage pas familier est un atout indéniable. Dès lors, il était important de proposer aux étudiants un outil dans l'air du temps permettant d'apprendre efficacement à maîtriser les méthodes de fouille de données et leurs subtilités.

Le but du présent document de travail pratique est de fournir un point de départ pour les novices au logiciel statistique et au langage informatique R. Les possibilités offertes par R étant très vastes, il est utile pour l'étudiant Master d'assimiler certaines notions et concepts pour évoluer plus aisément par la suite.

Ce TP d'initiation au logiciel R est largement inspiré d'un manuel produit par Emmanuel Paradis [15]. Pour une documentation plus complète, vous pourrez vous référer aux différentes informations que vous trouverez dans l'annexe A de ce présent document.

0.1 Présentation du langage et logiciel R

R est à la fois un logiciel de statistique et aussi un langage de Programmation. Il possède de nombreuses fonctionnalités pour la modélisation, le reporting et la visualisation de données. Il s'agit avant tout d'un langage de programmation inspiré du langage S développé dans les années 80 (Becker & Chambers, [2, 3]). Au courant des années 1990, Ross Ihaka et Robert Gentleman [11] créent le R au département de Statistiques de l'Université d'Auckland, Nouvelle Zélande.

Depuis 1997, une vingtaine de développeurs forment l'équipe de développement de R (R Development Core team [16]). Les membres de cette équipe ont les droits d'écriture sur le code source. Le logiciel R fonctionne initialement en ligne de commande, mais des interfaces permettent désormais une utilisation plus conviviale. C'est un langage interprété et non compilé c'est-à-dire que les commandes tapées au clavier sont directement exécutées sans qu'il soit besoin de construire un programme complet comme cela est le cas pour la plupart des langages informatiques (C, Fortran, Pascal, . . .). Toutefois, ce n'est pas seulement un « autre » environnement statistique (comme SPSS ou SAS, par exemple), mais aussi un langage de programmation complet et autonome.

Logiciel gratuit et open source, le développement de R s'appuie sur une large communauté. Le langage R est l'un des programmes d'analyse statistique de référence auprès de la communauté scientifique et universitaire. Parmi les caractéristiques particulièrement intéressantes du R, nous notons :

- Une large collection intégrée et cohérente d'outils d'analyse statistique ;
- Un système performant de stockage et de manipulation des données ;
- Un large éventail d'outils graphiques particulièrement flexibles ;
- Il est basé sur la notion de vecteur, ce qui simplifie les calculs mathématiques et réduit considérablement le recours aux structures itératives (boucles for, while, etc.) ;
- Il ne nécessite pas de typage ni de déclaration obligatoire des variables ;
- La plupart des méthodes avancées de statistique sont aussi disponibles au travers de modules externes appelés packages. Ceux-ci sont faciles à installer directement à partir d'un menu du logiciel.

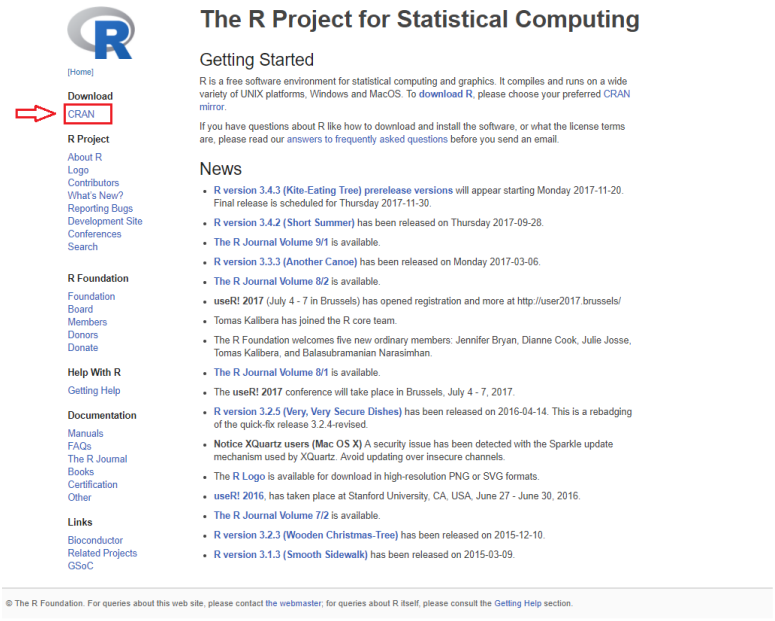
0.2 Installation et description du Logiciel R

Un point fort de R réside dans le fait que ce logiciel est distribué librement. Son installation peut être mise en œuvre à partir du site internet du *Comprehensive R Archive Network (CRAN)* [1] qui d'une part met à disposition les exécutables et d'autres part donne des informations relatives à la procédure d'installation. Il évolue très rapidement et environ tous les six mois une nouvelle version du langage est proposée au public.

Procédure d'installation

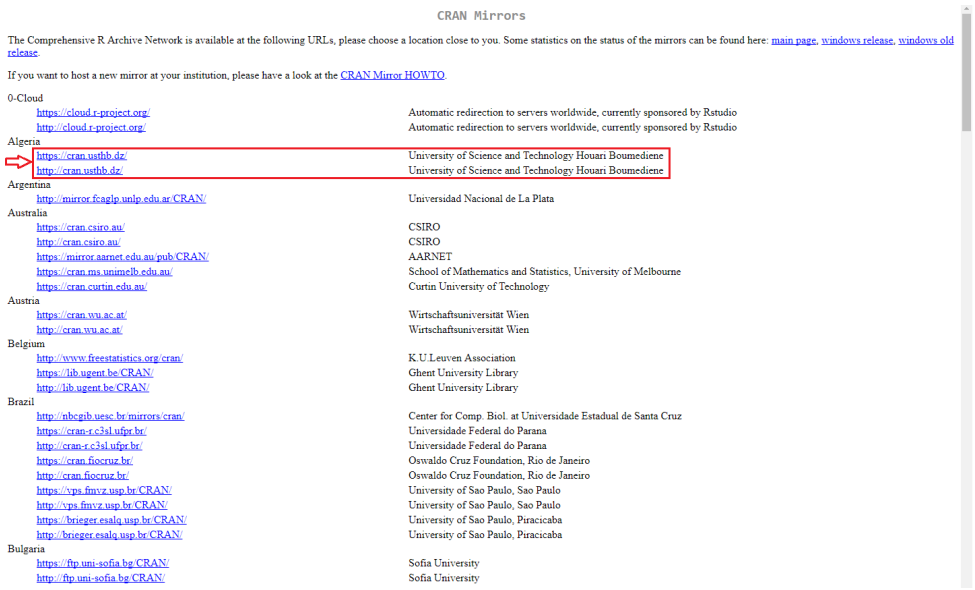
Vous pouvez télécharger la version la plus à jour de R comme suit :

- Allez sur le site `http://www.r-project.org/` ;



1. https ://cran.r-project.org/

– Cliquez sur « CRAN » dans le menu à gauche ;
– Sélectionnez le site miroir du CRAN le plus près de chez vous ; à savoir pour notre cas, ça sera l'University of Science and Technology Houari Boumediene Alger, Algérie



CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

If you want to host a new mirror at your institution, please have a look at the CRAN Mirror HOWTO.

0-Cloud	
https://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
http://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
Algeria	
https://cran.usthb.dz/	University of Science and Technology Houari Boumediene
http://cran.usthb.dz/	University of Science and Technology Houari Boumediene
Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	
https://cran.csiro.au/	CSIRO
http://cran.csiro.au/	CSIRO
https://mirror.aarnet.edu.au/pub/CRAN/	AARNET
https://cran.ms.unimelb.edu.au/	School of Mathematics and Statistics, University of Melbourne
https://cran.curtin.edu.au/	Curtin University of Technology
Austria	
https://cran.wu.ac.at/	Wirtschaftsuniversität Wien
http://cran.wu.ac.at/	Wirtschaftsuniversität Wien
Belgium	
http://www.freestatistics.org/cran/	K.U.Leuven Association
https://lib.ugent.be/CRAN/	Ghent University Library
http://lib.ugent.be/CRAN/	Ghent University Library
Brazil	
http://nbcgib.uesc.br/mirrors/cran/	Center for Comp. Biol. at Universidade Estadual de Santa Cruz
https://cran-r.c3sl.ufpr.br/	Universidade Federal do Parana
http://cran-r.c3sl.ufpr.br/	Universidade Federal do Parana
https://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
http://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
https://vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
http://vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
https://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba
http://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba
Bulgaria	
https://ftp.uni-sofia.bg/CRAN/	Sofia University
http://ftp.uni-sofia.bg/CRAN/	Sofia University

– Dans la case intitulée « Download and Install R », cliquez sur le lien correspondant à votre système d'exploitation. R est disponible pour Microsoft Windows, Macintosh et de nombreux systèmes de type Unix. Dans notre cas, nous sélectionnons le système d'exploitation Microsoft Windows.

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Thursday 2017-09-28, Short Summer) R-3.4.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

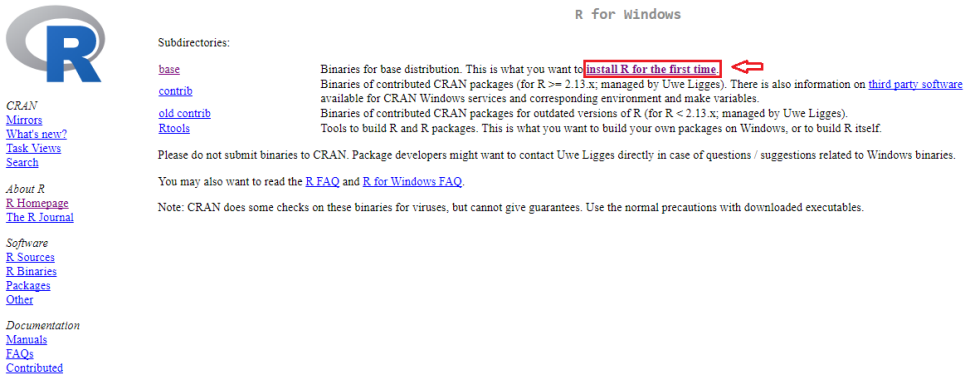
Submitting to CRAN

To "submit" a package to CRAN, check that your submission meets the CRAN Repository Policy and then use the web form.

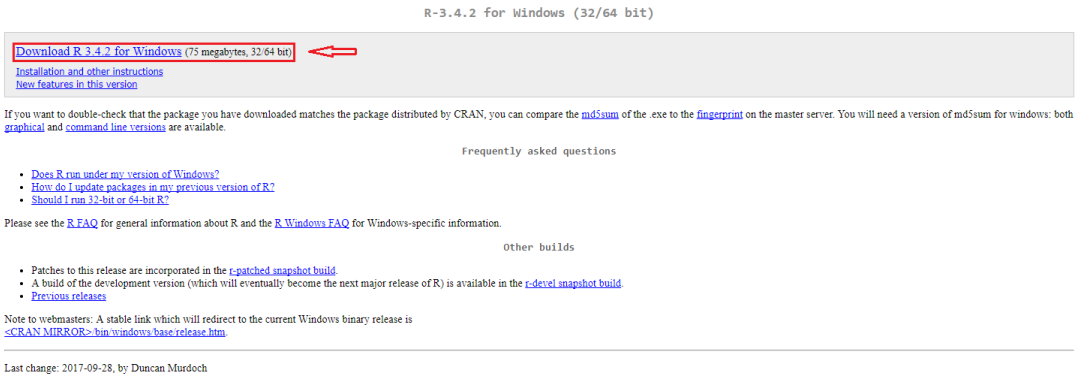
If this fails, upload to ftp://CRAN.R-project.org/incoming/ and send an email to CRAN-submissions@R-project.org following the policy. Please do not attach submissions to emails, because this will clutter up the mailboxes of half a dozen people.

Note that we generally do not accept submissions of precompiled binaries due to security reasons. All binary distribution listed above are compiled by selected maintainers, who are in charge for all binaries of their platform, respectively.

– Afin d'installer R, il nous faudra la version basique, acceder au lien de « install R for the first time »



– Cliquez sur le lien de téléchargement pour la dernière version disponible pour Windows (32/64 bits).



Il existe plusieurs interfaces graphiques « GUI : Graphical User Interface », qui permettent d'accéder à une partie des fonctions du langage R.

– **RGUI** l'interface graphique installée par défaut sous Windows (Figure0.1).
– **JGR** une interface graphique programmée en Java pour R.
– **Rattle** une interface graphique pour le data mining utilisant R.
– **Statistical Lab**
– **RExcel** pour exécuter les fonctionnalités de R et de R Commander à partir de Microsoft Excel.
– **rggobi** une interface pour le logiciel GGobi spécialisé dans la visualisation de données multidimensionnelles.

Dans ces travaux pratiques dédiés à l'application des techniques de fouille de données, nous allons tout d'abord présenter l'interface graphique classique RGUI sous Windows. Elle facilite certaines opérations tel que l'installation de packages externes, mais elle offre autrement peu de fonctionnalités additionnelles pour l'édition de code R. Pour éluder cet inconvénient, il existe des programmes qui facilitent l'écriture des instructions et

des programmes en langage R. Ces programmes se regroupent en deux grandes catégories :

- Les éditeurs de texte comme le bloc-note, Wordpad, Notepad ++ ou Word.
- Les environnements de programmation (IDEs : Integrated development environments), comme Tinn-R, Emacs (Emacs Speaks Statistics), Jedit, Kate, WinEdt (R Package RWinEdt) ou Vim.

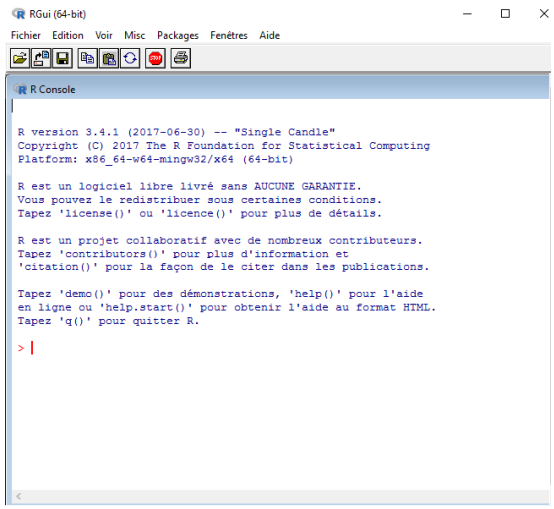


Fig. 0.1. Console R sous Microsoft Windows 64 bits version 3.4.1 datée du 30 Juin 2017.

Tableau 0.1. Menu de la console RGUI

Le menu	**Fonctionnalité**
Le menu Fichier	contient les outils nécessaires à la gestion de l'espace de travail, tels que la sélection du répertoire par défaut, le chargement de fichiers sources externes, la sauvegarde et le chargement d'historiques des commandes exécutées, etc.
Le menu Edition	contient les habituelles commandes de copier-coller, ainsi que la boîte de dialogue autorisant la personnalisation de l'apparence de l'interface.
Le menu Voir	permet d'afficher ou de masquer la barre d'outils et la barre de statut.
Le menu Misc	traite de la gestion des objets en mémoire et permet d'arrêter un calcul ou des calculs en cours de traitement.
Le menu Packages	automatise la gestion et le suivi des bibliothèques de fonctions, permettant leur installation et leur mise à jour de manière transparente depuis l'un des miroirs du CRAN (Comprehensive R Archive Network) http ://cran.r-project.org/.
le menu Fenêtres et le menu Aide	assument des fonctions similaires à celles qu'ils occupent dans les autres applications Windows, à savoir la définition spatiale des fenêtres et l'accès à l'aide en ligne et aux manuels de références du langage R.

RGUI fonctionne avec plusieurs fenêtres sous Windows :

- La fenêtre R Console est la fenêtre principale où sont réalisées par défaut les entrées de commandes et les sorties de résultats en mode texte. Le menu de la console RGUI est résumé dans le tableau (Table 0.1).
- Les fenêtres facultatives, telles que les fenêtres graphiques, les fenêtres d'informations (historique des commandes, aide, visualisation de fichier, etc), toutes appelées par des commandes spécifiques via la fenêtre R Console.

0.3 Fonctionnement de R

La syntaxe associée à R est relativement simple même si quelques règles sont à connaître :

- Le symbole « > » qui apparaît en début de ligne montre que R est prêt à être utilisé.
- Le symbole « # » indique à R de ne pas évaluer le code qui suit, jusqu'à la fin de la ligne. Il permet donc de placer des lignes en commentaires.
- Lorsqu'on fournit à R une commande incomplète, celui-ci nous propose de la compléter en nous présentant une invite de commande spéciale utilisant le signe «+».
- L'appel à une fonction se fait avec ses paramètres entre parenthèses, l'absence de parenthèses provoque l'affichage du code de la fonction.
- À noter que les espaces autour des opérateurs n'ont pas d'importance lorsque l'on saisit les commandes dans R.
- Le caractère « **;** » sépare deux commandes qui seront exécutées séquentiellement.
- Afin de connaître avec précision les options d'une fonction, il suffit de consulter l'aide en ligne accessible grâce à la commande **help(fonction)** ou **?fonction**.
- L'ensemble des commandes que vous passez est gardée en mémoire dans un fichier .history
- Tous les « objets » (variables, fonctions. . .) crées sont stockés dans un fichier générique nommé **.Rdata**. Mais il est préférable de stocker vos données dans des fichiers propres (Ex : tp.Rdata)
- Il est recommandé de créer un répertoire de travail dédié à R sur votre disque dur. Puis spécifiez-le sous R en passant par **File/Change dir.**

Quand R est utilisé, les variables, les données, les fonctions, les résultats, etc, sont stockés dans la mémoire de l'ordinateur sous forme d'objets qui ont chacun un nom. L'utilisateur va agir sur ces objets avec des opérateurs (arithmétiques, logiques, de comparaison, . . .) et des fonctions (qui sont-elles mêmes des objets).

De nombreuses fonctions sont déjà stockées dans une bibliothèque localisée sur le disque dans le répertoire *R_HOME/library* (R_HOME désignant le répertoire où R est installé). Ce répertoire contient des "packages" de fonctions, eux-mêmes présents sur le disque sous forme de répertoire. Le package **base** constitue le cœur du logiciel et contient, comme son nom l'indique, les fonctions de base. Chaque package a un répertoire nommé R avec un fichier qui a pour nom celui du package (par exemple, pour base, ce sera le fichier R_HOME/library/base/R/base). Ce fichier contient les fonctions du package.

Si l'on souhaite utiliser des fonctions appartenant à un autre package, il suffit de charger ce dernier à l'aide de la commande Library. Par exemple **library(rpart)** permet de charger les fonctions dédiées à l'arbre de decision CART (Classification And Regression Tree).

0.4 Applications : premiers pas avec R

0.4.1 Calcul élémentaires

R permet de faire les opérations de calcul élémentaire :

```
R> 2 + 3
R> 2 * 4
R> 3 - 8
R> 2.1 - 6 * 2.5
R> 3 * 2 - 5 * (2-4) / 6.02
R> 2 ^ 2
R> ((1+sqrt(5))/2)^2
```

Le résultat peut être stocké dans une variable $R > a = 2 + 3$ gardée en mémoire (a apparaît alors dans la fenêtre espace de travail), et qui peut être ré-utilisée par la suite :

```
R> nombredor = sqrt(a).
```

Toutes les variables créées sont stockées dans la mémoire de R. On peut obtenir la liste des objets stockés par la fonction **ls (liste)** :

```
R> ls()
```

Il est possible aussi d'effacer une variable avec la fonction **rm (remove)** :

```
R> rm()
R> rm(list=ls())
```

0.4.2 Scalaires, vecteurs et matrices

La variable a définie précédemment est un scalaire. R gère également des vecteurs et matrices. Un des avantages de R est qu'un grand nombre d'opérations et de fonctions sont applicables directement sur des vecteurs (voir sur des matrices). Ainsi, le recours au boucle de type for peut être évitée, et doit généralement l'être pour des raisons de rapidité d'exécution.

Pour créer un vecteur, il est possible d'utiliser la fonction de concaténation **c** :

```
R> x = c(7,8,9)
R> y = 1:3
R> z = rep(x,y)
```

Il y a d'autres manières de créer des vecteurs par la fonction **seq (séquence)** :

```
R> seq(from=1, to=2, by=0.25)
R> seq(from=1, to=2, by=0.25)
```

La commande **matrix** permet de créer une matrice

```
R> M = matrix(z,2,3)
```

Ce qui peut également être fait en concaténant des vecteurs en ligne **(rbind)** ou en colonne **(cbind)** :

```
R> M = cbind(x,y)
R> M = rbind(x,y)
```

Les tableaux à plus de 2 dimensions, appelés **array** en R, sont également utilisables :

```
R> T = array(0,dim=c(2,3,4))
```

0.4.3 Les fonctions

R permet de faire des calculs plus élaborés. Il utilise pour cela des fonctions. Il dispose d'un grand nombre de fonctions prédéfinies, utilisables en appelant la fonction par son nom suivi de ses arguments entre parenthèses :

```
R> mean(x)
R> rnorm(10)
R> abs(-4)
R> log(4)
R> exp(1)
R> round(3.1415)
```

R possède aussi en mémoire la valeur de quelques constantes mathématiques :

```
R> pi
R> cos (2 * pi)
```

0.4.4 Structure des données

R gère également des listes, ainsi qu'une structure spécifique à R : le data frame

Listes

Une liste est une combinaison de structures de données de natures potentiellement différentes :

```
R> L=list(elt1=c(1,2,3),elt2=matrix(rnorm(9),3,3),elt3='tutu',
elt4=seq(1,4,by=0.5))
```

Les éléments de la liste sont alors accessibles par un $, et les noms des éléments par la commande **names** :

```
R> L$elt4
R> names(L)
```

Data frames

En général, les données d'une étude statistique sont présentées sous forme matricielle. Les lignes représentent les individus observés et les colonnes les variables étudiées. Le logiciel R, pour ses applications les plus évoluées, fonctionne avec des structures de données appelées data frame. Il est possible de créer une data frame en transformant une matrice numérique lorsque toutes les données sont numériques. Nous appliquons :

```
R> mat = matrix(1:9, ncol = 3)
R> as.data.frame(mat)
```

On peut aussi créer une data frame en concaténant des objets de différentes natures :

```
R> age = c(24, 26, 22)
R> sexe = c("H", "H", "F")
R> Healthy = c(TRUE, FALSE, TRUE)
R> enquete = data.frame(age, sexe, Healthy)
R> enquete
```

0.4.5 Les graphiques

R permet de créer un grand nombre de graphiques, qui seront introduits au fur et à mesure des cours de Statistique. Nous présentons ici deux premières fonctions **plot()** et **hist()**. La fonction **plot()** permet de représenter un nuage de points (Figure 0.2) :

```
R> x=rnorm(20);y=2*x+1+rnorm(20,0,0.1)
R> plot(x,y,type='p',xlab='x',ylab='y',main='Nuage de points',col=2)
```

La fonction **hist** permet de représenter un histogramme (Figure0.2) :

```
R> hist(rnorm(1000),breaks=20,main='Histogramme')
```

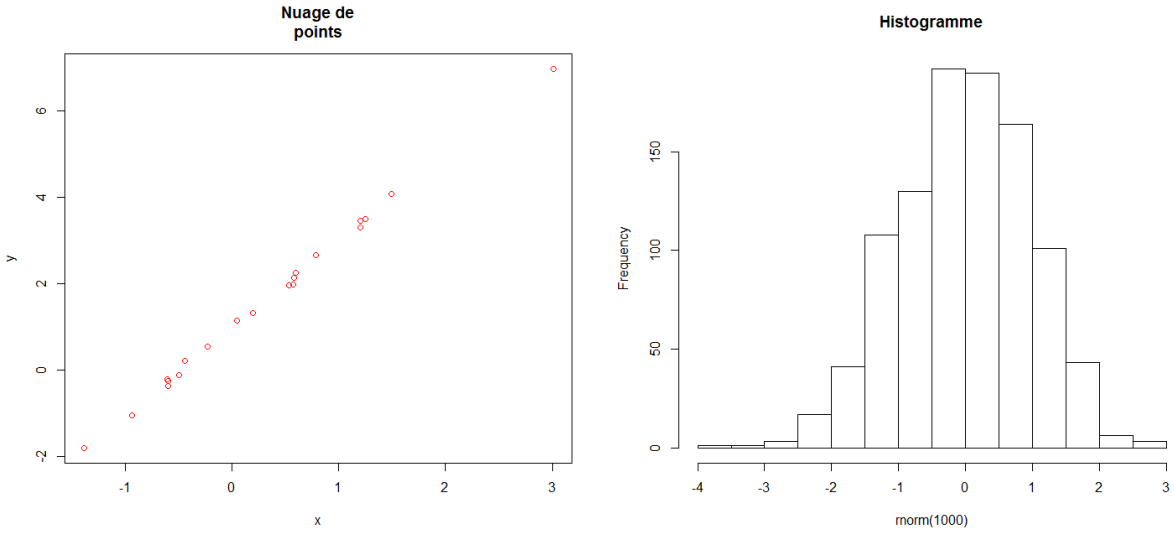


Fig. 0.2. Nuage de points (gauche) avec la fonction plot et histogramme avec la fonction hist

0.4.6 Importer et exporter des fichiers de données

Il y a plusieurs façons d'importer et d'exporter des fichiers de données dans R. Les principales sont les fonctions **write.table** et **read.table** qui permettent respectivement d'exporter dans un fichier texte un data frame et d'importer un fichier texte (de type individus en ligne et variables en colonnes) dans un data frame.

```
R> read.table(file, header=FALSE, sep="",quote="\"'", dec=".",
row.names=,col.names=,as.is=FALSE, na.strings="NA", skip=0,
check.names=TRUE, strip.white=FALSE)
```

file	le nom du fichier (entre ""), éventuellement avec son chemin d'accès (le symbole est interdit et doit être remplacé par /, même sous Windows)
header	une valeur logique (FALSE ou TRUE) indiquant si le fichier contient les noms des variables sur la 1ère ligne

sep	le séparateur de champ dans le fichier, par exemple sep="\t" si c'est une tabulation
quote	les caractères utilisés pour citer les variables de mode caractère
dec	le caractère utilisé pour les décimales
row.names	un vecteur contenant les noms des lignes qui peut être un vecteur de mode caractère, ou le numéro (ou le nom) d'une variable du fichier (par défaut : 1, 2, 3, ...)
col.names	un vecteur contenant les noms des variables (par défaut : V1, V2, V3, ...)
as.is	contrôle la conversion des variables caractères en factor (si FALSE) ou les conserve en caractères (TRUE)
na.strings	indique la valeur des données manquantes (sera converti en NA)
skip	le nombre de lignes à sauter avant de commencer la lecture des données
check.names	si TRUE, vérifie que les noms des variables sont valides pour R
strip.white (conditionnel à sep)	si TRUE, scan efface les espaces (= blancs) avant et après les variables de mode caractère

```
R > write.table(x, file, append=FALSE, quote=TRUE,
sep=" ", eol="\n", na="NA", dec=".", row.names=TRUE,
col.names=TRUE)
```

sep	le séparateur de champ dans le fichier
col.names	un nombre indiquant si les noms des colonnes doivent être écrits dans le fichier row.names idem pour les noms des lignes
quote	un nombre ou un vecteur numérique ; si TRUE, les variables de mode caractère sont entourées de guillemets "" ; si un vecteur est numérique, ses éléments donnent les indices des colonnes avec guillemets. Dans les deux cas, les noms des lignes et des colonnes sont aussi avec guillemets s'ils sont écrits.
dec	le caractère utilisé pour les décimales
na	indique le caractère utilisé pour les données manquantes
eol	le caractère imprimé à la fin de chaque ligne ("\n" correspond à un retour-charriot)

0.4.7 Éléments de programmation

Condition if

La syntaxe pour la condition **if** est la suivante (la condition **else** pouvant être omise) :

```
R> # Exemple 1
R> x=3
R> if(x == 2)
      print("Hello")
      else
      print("x est différent de 2")

R> # Exemple 2
R> w=2
R> if (w>3){
R> res=2
R> }else{
R> res=4
R> }
R> print(res)
```

Boucle for

Une boucle for a la syntaxe suivante

```
R># Exemple 1
R> vec=c()
R> for (i in 1:10){
R> vec=c(vec,i)
R> cat('iteration numero: ',i,'/n')
R> }

R># Exemple 2
R> x <- 1:5
R> y <- numeric(length(x))
R> for (i in 1:length(x)) if (x[i] == b) y[i] <- 0
                                        else y[i] <- 1
```

Boucle while

Quand on veut répéter un calcul tant qu'une condition est satisfaite, on utilise la fonction **while()**, avec la syntaxe suivante :

```
R> x <- 100
R> while(x/3 > 1){
R> x <- x/3
R> }
```

Écrire ses propres fonctions

Un des grands intérêts du logiciel R est qu'il est possible de créer ses propres fonctions. Voici ci-dessous un exemple permettant de présenter la syntaxe. Les arguments de la fonction sont donnés après l'instance **function** ; une valeur par défaut à un argument peut être donnée en indiquant cette valeur lorsque les arguments sont définis (par exemple arg2=0 indique que l'argument arg2 aura la valeur nulle par défaut). Le résultat de la fonction peut être de différente nature (scalaire, vecteur, matrice, liste...).

```
R> mafonction <- function(arg1,arg2=0,arg3=1){
R> tmp=1/sqrt(2*pi*arg3) * exp(-1/2 * ((arg1-arg2)/(sqrt(arg3)))^2)
R> return(res=list(argument1=arg1,densite=tmp))
R> }
R> x=seq(-3,5,0.01)
R> y=mafonction(x,arg2=1)
R> plot(x,y$densite,type='l',col=3)
```

Remarque : L'annexe A passe en revue les quelques commandes essentielles à connaître pour commencer à travailler sous R. L'ouvrage de Goulet. (2014) [9] constitue une excellente référence pour l'apprentissage plus poussé du langage et logiciel R.

0.5 Applications

Organisation du travail

L'expérience prouve que la meilleure stratégie est de créer un répertoire (dossier) pour chaque travail pratique, et d'y disposer :

1. les fichiers de données brutes ;
2. le fichier script contenant les commandes R ;
3. le workspace et le (s) fichier(s) résultats (textes et graphiques).

Exercice 1

1. Soit x un vecteur contenant les valeurs d'un échantillon :

   ```
   R> x = [ 1 18 2 1 5 2 6 1 12 3 13 8 20 1 5 7 7 4 14 10]
   ```

   Écrire une expression R permettant d'extraire les éléments suivants.

   a) Le deuxième élément de l'échantillon.

   b) Les cinq premiers éléments de l'échantillon.

   c) Les éléments strictement supérieurs à 14.

   d) Tous les éléments sauf les éléments en positions 6, 10 et 12.

2. Soit x une matrice $10 * 7$ obtenue aléatoirement avec :

   ```
   R> x <- matrix(sample(1:100, 70), 7, 10)
   ```

   Écrire des expressions R permettant d'obtenir les éléments de la matrice demandés ci-dessous.

   a) L'élément (4 ; 3).

   b) Le contenu de la sixième ligne.

   c) Les première et quatrième colonnes (simultanément).

   d) Les lignes dont le premier élément est supérieur à 50.

Exercice 2 : Manipulation de matrices

1. Créer la matrice suivante :

$$A = \begin{pmatrix} -3 & 5 & 6 \\ -1 & 2 & 2 \\ 1 & -1 & -1 \end{pmatrix}$$

2. Afficher la dimension de A, son nombre de colonnes, son nombre de lignes et sa longueur ;
3. Extraire la seconde colonne de A, puis la première ligne ;
4. Extraire l'élément en troisième position à la première ligne ;
5. Extraire la sous-matrice de dimension $2 * 2$ du coin inférieur de A, c'est-à-dire

$$\begin{pmatrix} 2 & 2 \\ -1 & -1 \end{pmatrix}$$

6. Calculer la somme des colonnes puis des lignes de A ;
7. Afficher la diagonale de A
8. Rajouter le vecteur droite de la matrice A et stocker le résultat dans un objet appelé B ;
9. Retirer le quatrième vecteur de B ;
10. Retirer la première et la troisième ligne de B ;
11. Ajouter le scalaire 10 à A ;
12. Ajouter la matrice identité I_3 à A ;
13. Diviser tous les éléments de la matrice A par 2 ;
14. Multiplier la matrice A par le vecteur $[1 \quad 2 \quad 3]^T$
15. Afficher la transposée de A ;
16. Effectuer le produit avec transposition $A^T A$.

Exercice 3 : Fonction de Ricker

Considérons que l'on veuille étudier le comportement du modèle non linéaire de Ricker défini par :

$$N_{t+1} = N_t exp[r(1 - \frac{N_t}{K})]$$

Ce modèle est très utilisé en dynamique des populations, en particulier de poissons. On voudra à l'aide d'une fonction simuler ce modèle en fonction du taux de croissance r et de l'effectif initial de la population N_0 (la capacité du milieu K est couramment prise égale à 1 et cette valeur sera prise par défaut) ; les résultats seront affichés sous forme de graphique montrant les changements d'effectifs au cours du temps. On ajoutera une option qui permettra de réduire l'affichage des résultats aux dernières générations (par défaut tous les résultats seront affichés).

Mettre en place le programme et la fonction qui permettra de faire cette analyse numérique du modèle de Ricker.

1

TP1 : Pré-traitement des données : les méthodes de visualisation et de description

Introduction

Dans le processus d'extraction et gestion de données, la phase de pré-traitement est relativement importante, cette étape conditionne la qualité des modèles établis en fouille de données et permet de faire émerger la connaissance contenue au sein des données.

Le pré-traitement des données consiste à traiter les données bruitées, soit en les supprimant, soit en les modifiant de manière à en tirer le meilleur profit.

Le pré-traitement permet aussi de réduite et à écarter définitivement le bruit pour accélérer les calculs et représenter les données sous un format optimal pour l'exploration. Le pré-traitement concerne aussi l'élimination de la redondance. Une méthode largement utilisée dans ce contexte, est l'analyse en composantes principales (ACP). Une autre méthode de réduction est celle de la sélection et suppression des attributs les moins pertinents.

Plusieurs techniques de visualisation des données telles que les courbes, les diagrammes, les graphes,... etc, peuvent aider à la sélection et le nettoyage des données. Une fois les données collectées, nettoyées et pré-traitées, ces derniers pourront améliorer les résultats de la fouille de données dans le processus global d'extraction de connaissances.

1.1 Objectif

Lors de la collecte de données, ces dernières peuvent être omises à cause des erreurs de saisi ou à causes des erreurs dues à l'imprécision, des données peuvent être incohérentes ou atypiques c.-à-d. qui sortent des intervalles permis. Dans ce cas, nous sommes obligé de passer par une phase de pré-traitement des données.

Le but de ce TP est de traiter les données erronées, manquantes ou inconsistantes qui peuvent être présents lors de leurs collecte. Nous utilisons des méthodes de sélection et de transformation pour rendre les données exploitables pour des traitements futurs.

1.2 Importation de la base de données

Pour les lectures et écritures dans un répertoire, R utilise le répertoire de travail.

– **getwd() :**permet de connaître ce répertoire

- **setwd() :**permet de modifier le répertoire de travail
  R>setwd(''c :/data'')

R peut lire des données stockées dans un fichier texte (ascii) avec **read.table()**, en conservant sa structure éventuelle, la sortie est un data.frame les arguments de cette fonction sont :

- *file*= nom du fichier
- *sep*= séparateur (espace par défaut)
- *header*= booléen (=TRUE si le nom des colonnes est en tête, FALSE sinon)

Exemple : pour importer dans l'objet F les données situées dans le fichier « fichier.txt » :

```
R>F=read.table("C:\\ArR\\fichier.txt", sep="\t",header=TRUE)
```

Plusieurs variantes de données peuvent être lu par R, les fonctions : **read.csv(), read.delim(), read.fwf()**... sont utilisées pour des fichiers pour les formats comme : Excell, SAS, SPSS , bases de données SQL...)

1.3 Manipulation des données

Nous utilisons dans ce TP comme exemple d'étude, le jeu de données *« satisfaction_hopital.csv »* c'est une enquête de satisfactions sur plusieurs hôpitaux de la région parisienne, récupéré lors d'un cours de FUN [1] (France Unité Numérique). Cette collecte a été réalisée afin d'évaluer la qualité de relation et la quantité d'information reçue par le patient lors de son séjour à l'hôpital. (Voir l'ANNEXE B pour le descriptif de la base).

1.3.1 Aperçu des données

Pour avoir un aperçu sur la base de données, utilisons la fonction **summary()** :

```
R> base= read.csv ("C:\\...\\Documents\\satisfaction_hopital.csv",
sep=",", header=TRUE)
R>summary(base)
```

Si nous nous intéressons à la variable *âge* :

```
R>summary(base$age)
```

1.3.2 Transformation de variables quantitatives / qualitatives

La fonction **factor()** transforme les valeurs numériques ordonnées en niveaux (levels). Exemple, pour créer une nouvelle variable *qualitative service.c* á partir de la variable *quantitative service* :

```
R> base$service.c <- factor(base$service)
```

Pour comparer les deux variables, nous utilisons la fonction **summary()** :

```
R> summary(base$service)
R> summary(base$service.c)
```

1. https ://www.fun-mooc.fr/

Nous pouvons par la suite changer les labels des niveaux avec la fonction **levels()** :

```
R> levels(base$service.c)< c("S1","S2","S3","S4","S5","S6","S7","S8")
R> summary(base$service.c)
```

1.3.3 Changement de codage des variables

Pour coder la variable *profession* du codage numérique à qualitatif :

```
R> base$profession.c <-factor (base$profession,labels =c("agriculteur",
"artisan","cadre","intermédiaire","employé", "ouvrier","sans emploi",
"autre"))
R> summary(base$profession.c)
```

Pour coder les sexes dans le jeu de données « *satisfaction_hopital* » :

```
R> base$sexe.c <- factor(base$sexe,labels=c("homme","femme"))
R> summary(base$sexe.c)
```

1.3.4 Visualisation des données

Afin d'obtenir un visuel global de la base de données, il est possible de regrouper les histogrammes de toutes les variables (Figure1.1).

```
R> layout(matrix(c(1:9),3,3))
R> for(i in 1:9) {hist(base[,i],main=names(base)[i],xlab="")}
R> layout(1)
```

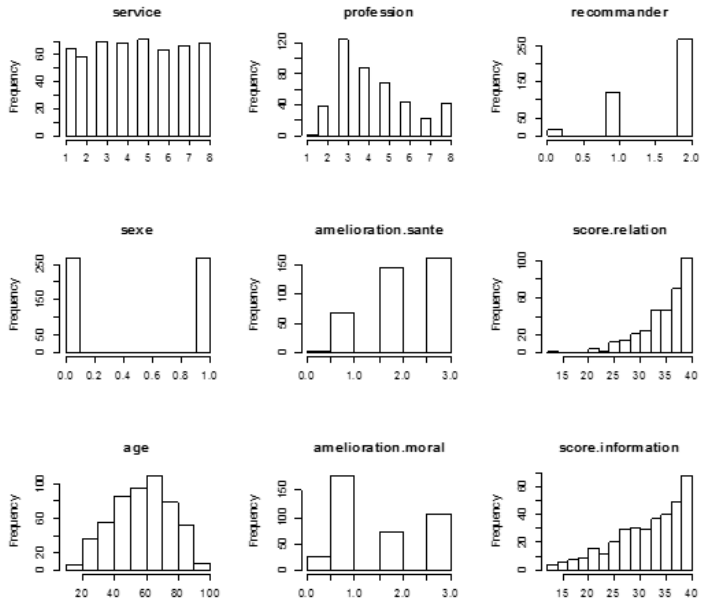


Fig. 1.1. Représentation graphique des histogrammes de toutes les variables

Pour comparer deux diagrammes en fonction d'un paramètre. Étudions la répartition des âges selon le sexe par les boites à moustaches **boxplot()** (Figure1.2) :

```
R> boxplot(base$age~base$sexe)
R> #un graphique plus élaboré et soigné
R>boxplot(base$age~base$sexe,names=c("Homme","Femme"),col=c("lightblue",
"hotpink"),main="Répartition des âges par sexe")
```

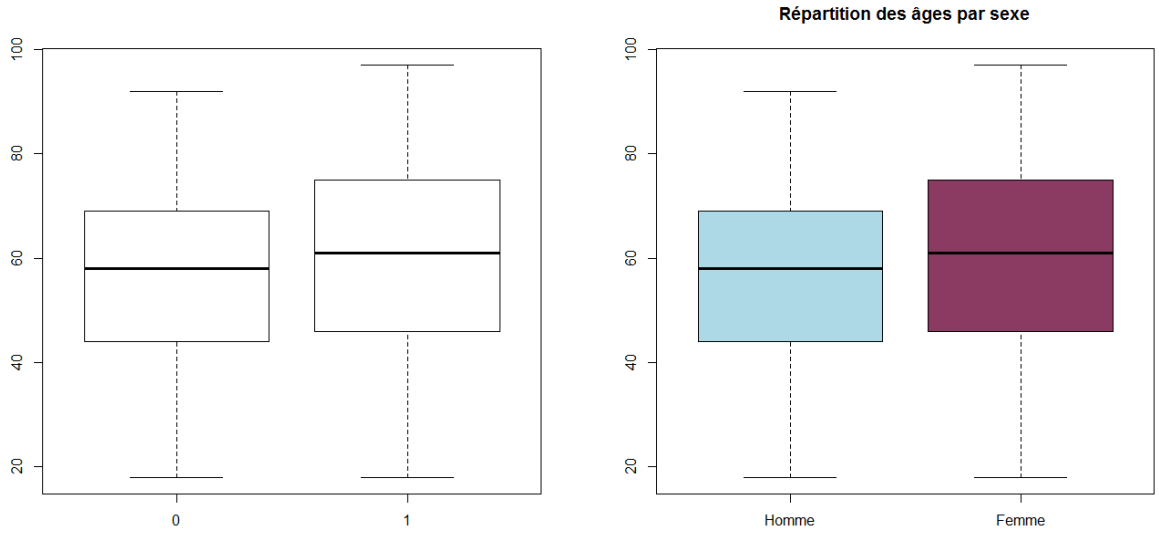


Fig. 1.2. Les boites à moustaches en fonction d'une variable.

Afin de réaliser les diagrammes croisés en bâtonnets, la commande **barplot()** prend en argument un tableau ; de ce fait, nous commençons par préparer les variables qualitatives en tableaux.

```
R> base$sexe.b<-factor(base$sexe,labels=c("H","F"))
R> tab.sexe<-table(base$sexe.b)
R> base$service.c<-factor(base$service)
R> tab.service<-table(base$service.c)
R> # Diagramme en bâtons par défaut
R> barplot(tab.sexe)
R> # Diagramme en bâtons amélioré
R> barplot(tab.service,main="Répartition des services",
xlab="Numéro des services",ylab="Effectifs",ylim=c(0,100),col=" burlywood")
R> tab.croise<-table(base$sexe.b,base$service.c,deparse.level=2)
R> barplot(tab.croise)
R> barplot(tab.croise, beside=TRUE,legend.text =TRUE,col= c("lightblue",
"hotpink4"),ylim= c(0,100), main="Répartition des services selon le sexe",
xlab="Numéro des services", ylab="Effectifs")
```

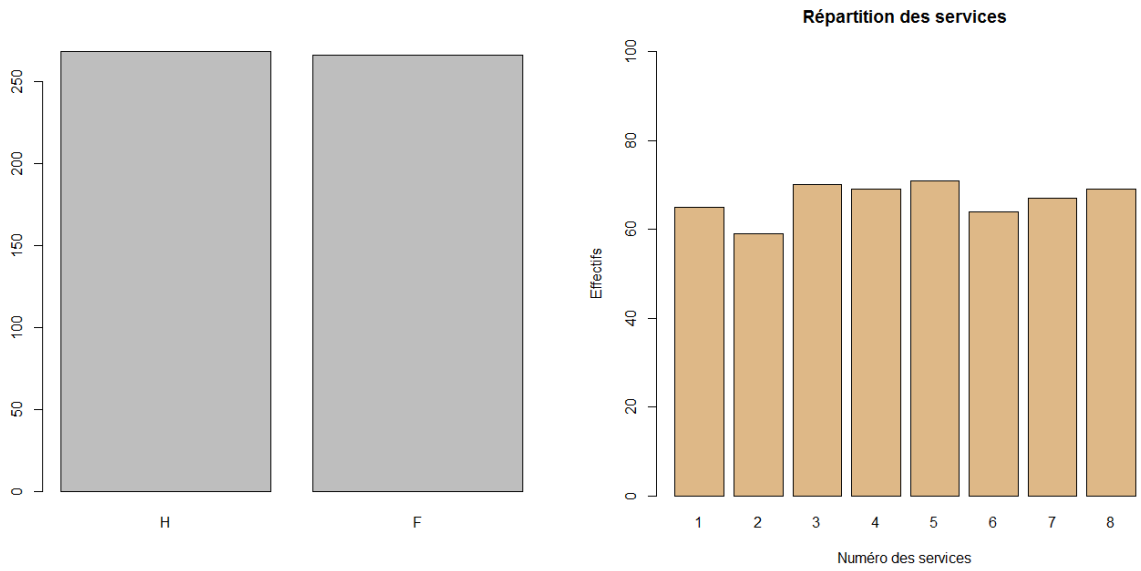


Fig. 1.3. Les diagrammes en bâtons

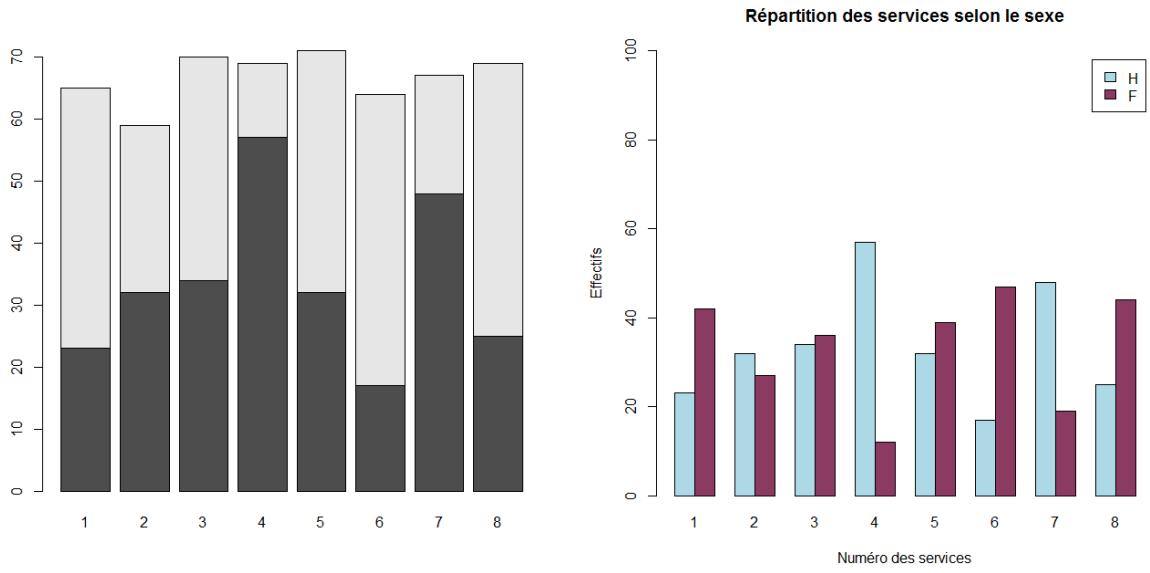


Fig. 1.4. Les diagrammes en bâtonnets croisés

1.3.5 Conditions et restrictions

Afin d'avoir une meilleure exploration des données sur lesquelles nous travaillons, il est possible d'effectuer des conditions et des restrictions qui permettrons le nettoyage des données.

```
R> base[base$age<20,]
```

Les conditions complexes avec **ET (&) et OU (|)**

```
R> base[base$age<=30 & base$sexe.b=="H",c("age","sexe.b","profession")]
R> base[base$age<=28 | base$age>=45,c("age","sexe.b")]
```

La fonction **table()** permet d'effectuer des comptages, avec tris à plat ou croisés.

```
R> C=base[base$age<=28 | base$age>=45,c("age","sexe.b")]
R> nrow(C)
R> ncol(C)
R> table(C$sexe.b)
R> E=table(base$sexe.b,base$profession.c)
R> print(E)
```

Il est possible d'effectuer des calculs récapulatifs

```
R> # Nbre d'agriculteurs dans la proportion H/F
R> print(sum(E(,2)]))
R> #Proportion de personnes sans emploi parmi les H et F
R> print(sum(E[,7]))
```

L'outil **tapply()** permet le calcul conditionnel, il renvoi un objet table, exploitable également.

```
R> # age moyen selon le sexe
R> tapply(X=base$age,INDEX=base$sexe.b,mean)
R> # age moyen selon le sexe et la profession
R> tapply(X=base$age,INDEX=list(base$sexe.b,base$profession.c),mean)
```

1.3.6 Les changements de valeurs

L'instruction **ifelse()** permet d'attribuer deux valeurs, en fonction du résultat du test. Exemple, pour transformer la variable *recommander* en variable binaire *recommander.b*, la valeur vaudra 1 et recommander vaut 2 et 0 sinon :

```
R> base$recommander.b<-ifelse(base$recommander==2,1,0)
R>table(base$recommander, base$recommander.b)
```

1.3.7 Traitement des données manquantes

Dans R, les valeurs manquantes sont représentées par le symbole **NA** (not available). Les valeurs impossibles (par exemple, la division par zéro) sont représentées par le symbole **NaN** (not a number). Contrairement à d'autres logiciels, R utilise le même symbole pour les données personnelles et numériques.

Test des valeurs manquantes

La fonction **is.na()** produit un booléen qui vaut TRUE si son paramètre est une valeur NA.

```
R> is.na(base$age)
```

Ce vecteur pourra être utilisé si vous souhaitez, par exemple, remplacez les valeurs manquantes par la valeur la plus fréquente dans le cas d'attributs qualitatifs ou la valeur moyenne de l'attribut dans le cas d'attributs numériques ou de faire des traitements qui ne s'appliquent qu'aux valeurs manquantes ou qu'aux valeurs présentes.

Suppression d'une variable

exemple nous voudrions supprimer la variable *sexe.c*

```
R> base$sexe.c<-NULL
R> base$sexe.c
```

Exclusion des valeurs manquantes de l'analyse

Les fonctions arithmétiques sur les valeurs manquantes produisent des valeurs manquantes.

```
R> mean(base$age) # nous donne NA
R> mean(base$age, na.rm=TRUE) # nous donne 58.21
```

La fonction **complete.cases()** renvoie un vecteur logique indiquant quels cas sont complets.

```
R> base[!complete.cases(base),]
```

La fonction **na.omit()** renvoie l'objet avec la suppression en listes des valeurs manquantes.

```
R> newdata <- na.omit(base)
R> summary(newdata)
```

La plupart des fonctions de modélisation dans R offrent des options pour traiter les valeurs manquantes. Il est possible d'aller au-delà de la suppression par liste des valeurs manquantes et d'utiliser des méthodes plus poussées telle que l'imputation multiple.

1.3.8 Tri des données

Si nous voulons afficher les 6 premières valeurs de la variable *age*, il suffira d'utiliser la fonction **head()**.

```
R> head(base$age)
```

Pour le tri des données deux fonctions principales peuvent être utilisées : **sort()** et **order()**.

- **sort()** permet de trier les éléments d'un vecteur, par défaut le tri est fait de manière décroissant.
- **order()** permet d'ordonner une data frame

```
R> age2=sort(base$age)
R> head(age2)
R> head(order(base$age))
R> head(base[order(base$age),])
R> head(base[order(base$age,base$score.information),])
```

1.4 Réduction de dimension

Une réduction de dimension consiste à obtenir une représentation des N données dans un espace de taille k, avec $k << p$ (p taille de l'espace originel). Cette diminution du nombre de variables décrivant les données peut avoir plusieurs objectifs :

1. Réduire le volume de données à traiter, tout en conservant au mieux «l'information utile».

2. Améliorer le rapport signal / bruit en supprimant des variables non pertinentes.
3. Améliorer la « lisibilité » des données en mettant en évidence des relations entre variables ou groupes de variables.
4. Atténuer la « malédiction de la dimension », qui désigne les cas où nous sommes en présence d'un nombre de variables nettement plus important que celui des observations.

Les méthodes de réduction de dimension sont regroupées suivant les deux approches suivantes : la transformation de variables et la sélection de variables.

1.4.1 La transformation de variables (feature extraction)

Cette approche consiste à construire de nouvelles variables à partir des variables initiales. Les nouvelles variables sont obtenues par des méthodes de type :
- Linéaires : trouver un sous-espace linéaire de dimension k dans l'espace initial R^p.
- Non linéaires : trouver un sous-espace non linéaire de dimension k dans l'espace initial.

Il est important de noter que dans cette approche, si les variables initiales ont une signification précise, les nouvelles variables sont rarement interprétables.

La transformation linéaire étant la plus utilisée, les trois méthodes factorielles linéaires les plus communément appliquées sont :

1. L'analyse en composantes principales (ACP), méthode à caractère exploratoire, adaptée à des données décrites par des variables quantitatives.
2. L'analyse factorielle discriminante (AFD), méthode à caractère exploratoire et décisionnel, adaptée à des données décrites par des variables quantitatives et appartenant à plusieurs classes.
3. L'analyse des correspondances multiples (ACM), méthode à caractère exploratoire, adaptée à des données décrites par des variables nominales.

Dans ce TP, nous appliquons L'ACP, ou Analyse en Composantes Principales qui est la méthode type pour l'exploration de données, elle consiste à réduire la dimensionnalité du problème pour en extraire l'essentiel.

Autrement dit, l'ACP cherche à définir k nouvelles variables combinaisons linéaires des p variables initiales qui feront perdre le moins d'information possible. Ces variables seront appelées «composantes principales », les axes qu'elles déterminent : «axes principaux» les formes linéaires associées : «facteurs principaux».

L'analyse en composantes principales permet de faire une synthèse de l'ensemble du tableau afin de :

- Synthétiser les liaisons entre variables (cercle des corrélations), définir les variables qui vont dans le même sens, dans un sens opposé, indépendantes ...
- Représenter dans un plan les individus afin de déterminer les individus proches ou éloignés, les regrouper en classe homogène, ... On parle de topologie des individus.
- Construire de nouvelles variables, appelées composantes principales, non corrélées et qui permettent de synthétiser l'information.

Ainsi, au lieu d'analyser le tableau à travers p variables, l'ACP permettra de limiter l'étude à quelques variables synthétiques (les composantes principales). La difficulté sera de donner un sens à ces variables et de proposer une analyse de ces résultats.

L'analyse en composantes principales sous R

La fonction **prcomp()** permet le calcul des valeurs propres de l'ACP, mais ne permet pas de réaliser efficacement et de manière élégante des ACP. En revanche, trois packages très complets écrits par des mathématiciens français contiennent toutes les options nécessaires :

- le package FactoMineR [2] (adossé à l'université de Rennes) ;
- le package ade4 [3] (adossé à l'université de Lyon) ;
- le package amap [4] (adossé à l'université de Toulouse).

Ce document décrira l'utilisation d'ade4 [5], qui est probablement le plus couramment utilisé. D'autre part, FactoMineR présente une interface graphique assez intuitive et en constante amélioration qui facilite l'application de l'ACP. De ce fait, il est recommandé aux étudiants de visiter le site web de FactoMineR [6] et amap [7] qui contiennent de nombreux détails et des tutoriels vidéo pour une réalisation simple d'une ACP.

```
R> #lecture et préparation des données
R> base= read.csv ("C:\\...\\Documents\\satisfaction_hopital.csv",
sep=",", header=TRUE)
R> #suppression des données manquantes
R> newdata<-na.omit(base)
```

L'ACP est appliquée que sur les variables quantitatives du jeu de données. Les variables qualitatives sont placées dans les variables supplémentaires, elles ne participent pas à la construction de l'ACP, mais elles apporteront de l'information supplémentaire par la suite de cette analyse.

```
R>#le calcul de l'ACP.
R> library(ade4)
R> acp<- dudi.pca(newdata, center=T, scale=T, nf=5)
R> print(acp)
R> print(summary(acp))
```

Avec :

- Nf : nombre de dimensions conservées dans les résultats finaux.
- Center : une valeur logique spécifiant si les variables doivent être déplacées pour être zéro centrées.
- Scale : une valeur logique. Si TRUE, les données sont mises à l'échelle de la variance unitaire avant l'analyse. Cette standardisation d'échelle évite que certaines variables deviennent dominantes simplement en raison de leurs grandes unités de mesure.

L'éboulis des valeurs propres

nous permet de connaître le pourcentage d'information (i.e. de variance, ou encore d'inertie) porté par chaque axe (ou composante principale) de l'ACP.

2. https ://cran.r-project.org/web/packages/FactoMineR/index.html
3. https ://cran.r-project.org/web/packages/ade4/index.html
4. https ://cran.r-project.org/web/packages/amap/index.html
5. http ://pbil.univ-lyon1.fr/ADE-4
6. http ://factominer.free.fr/.
7. http ://mulcyber.toulouse.inra.fr/projects/amap/

```
R> #l'éboulis des valeurs propres en %
R> inertie<-acp$eig/sum(acp$eig)*100
R> barplot(inertie,ylab="% d'inertie",
names.arg=round(inertie,2))
R> title("Eboulis des valeurs propres en %")
```

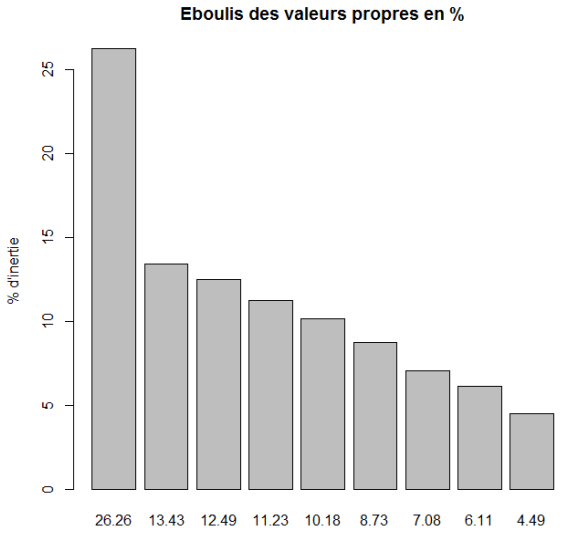


Fig. 1.5. Le diagramme des valeurs propres.

La matrice de corrélation

permet d'identifier rapidement quelques paires de variables fortement corrélées, et quelques paires de variables quasiment décorrélées.

```
R> head(acp$co)
```

Le cercle des corrélations

il est à noter que l'interprétation du cercle de corrélation se base sur les points suivants :

- plus une variable possède une qualité de représentation élevée dans l'ACP, plus sa flèche est longue ;
- plus deux variables sont corrélées, plus leurs flèches pointent dans la même direction (dans le cercle de corrélation, le coefficient de corrélation est symbolisé par les angles géométriques entre les flèches) ;
- plus une variable est proche d'un axe principal de l'ACP, plus elle est liée à lui. Cette dernière règle permet généralement de donner un sens concret aux axes de l'ACP.

```
R> s.corcircle(acp$co)
R> title("Le cercle des corrélations")
```

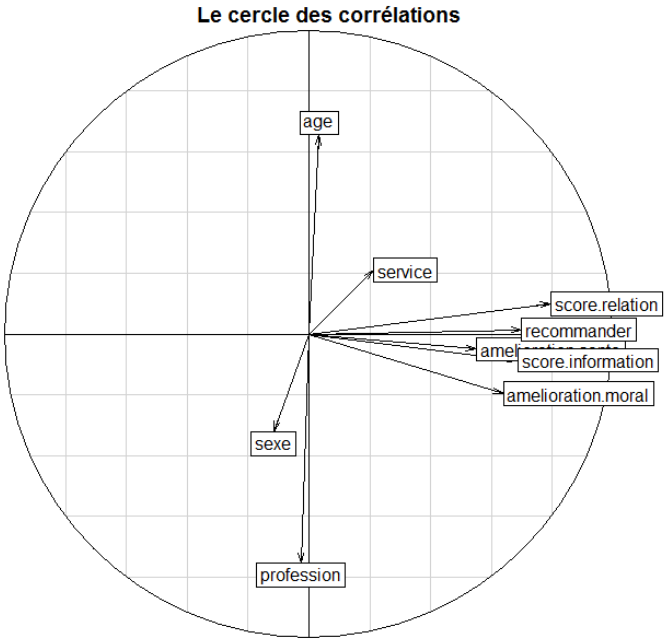


Fig. 1.6. La représentation graphique du cercle des correlations

Le nuage d'individus

Il soustrait des informations concernant les composantes principales choisies, par l'analyse de la représentation des individus. En effet, un individu mal représenté se situe généralement près du centre du repère, et sa spécificité est mal prise en compte par l'ACP, pour les composantes principales considérées.

```
R> s.label(acp$li, xax = 1, yax = 2)
```

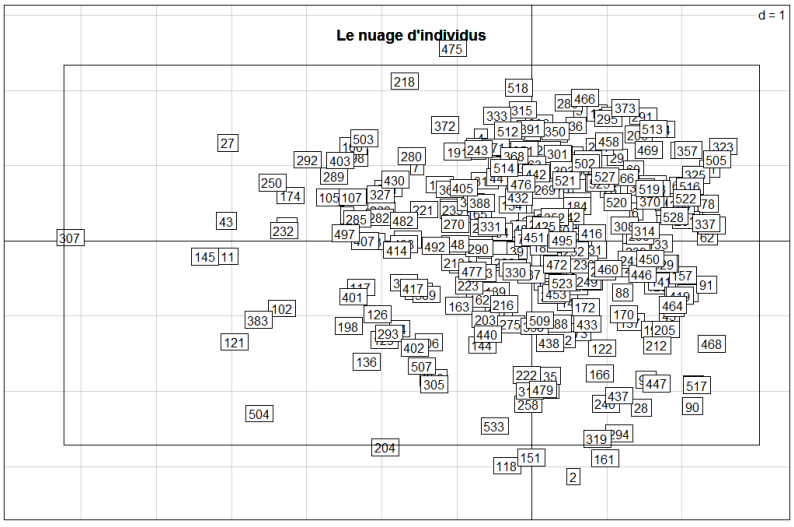


Fig. 1.7. La représentation graphique du nuage de points des individus.

Les coordonnées carrées des variables COSINUS

C'est un indicateur de la qualité de la représentation des variables sur les composantes principales. Permet de déterminer les variables qui pèsent le plus dans la définition d'une composante.

```
R> inertia <- inertia.dudi(acp, row.inertia=TRUE,col.inertia = TRUE)
R> var.cos2 <- abs(inertia$col.rel/10000)
R> head(var.cos2)
R> # ou bien plus simplement
R> head(acp$co^2)
```

- Un cos^2 élevé indique une bonne représentation de la variable sur la composante principale. Dans ce cas, la variable est positionnée à proximité de la circonférence du cercle de corrélation.
- Un cos^2 faible indique que la variable n'est pas parfaitement représentée par les composantes principales. Dans ce cas, la variable est proche du centre du cercle.

La contribution des variables aux composantes principales

Les contributions peuvent être imprimées en% comme suit :

```
R> var.contrib <- inertia$col.abs/100
R> head(var.contrib)
```

Les variables les plus importantes pour une composante principale donnée peuvent être visualisées à l'aide de la fonction **fviz_pca_contrib()** du package **factoextra** [8] :

```
R> library(factoextra)
R> fviz_pca_contrib(acp, choice = "var", axes = 1)
R> fviz_pca_contrib(acp, choice = "var", axes = 2)
```

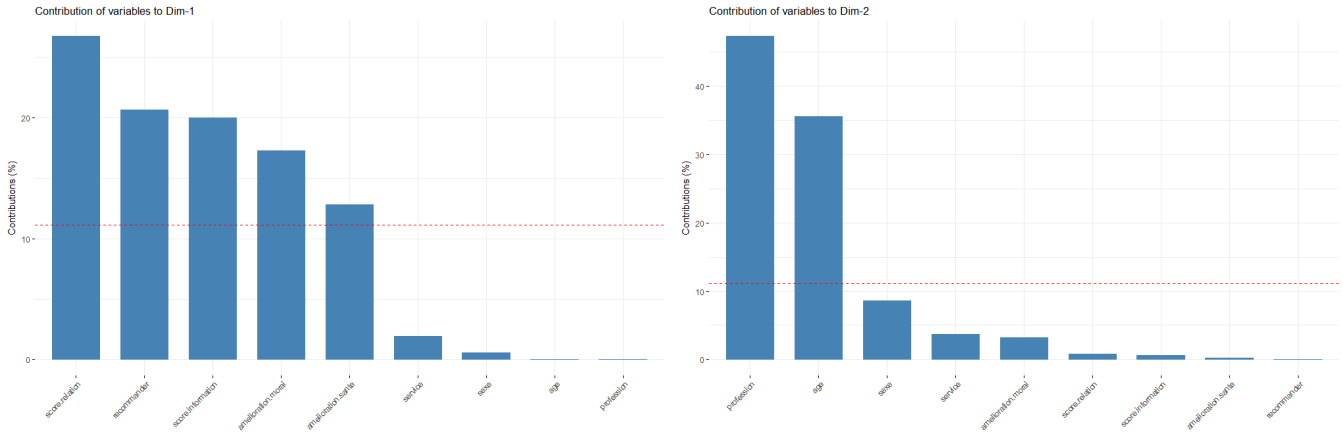


Fig. 1.8. Le diagramme de contribution des variables pour les composantes principales une et deux.

Si la contribution des variables est uniforme, la valeur attendue sera 1 / longueur (variables) = 1/10 = 10%.

8. https ://cran.r-project.org/web/packages/factoextra/index.html

La ligne pointillée rouge sur le graphique ci-dessus indique la contribution moyenne attendue. Pour une composante donnée, une variable avec une contribution supérieure à cette coupure pourrait être considérée comme importante pour contribuer à la composante.

```
R>#Classement qualité de la représentation des variables
dans toutes les composantes principales
R>fviz_cos2(acp, choice="var", axes = 1:5)
```

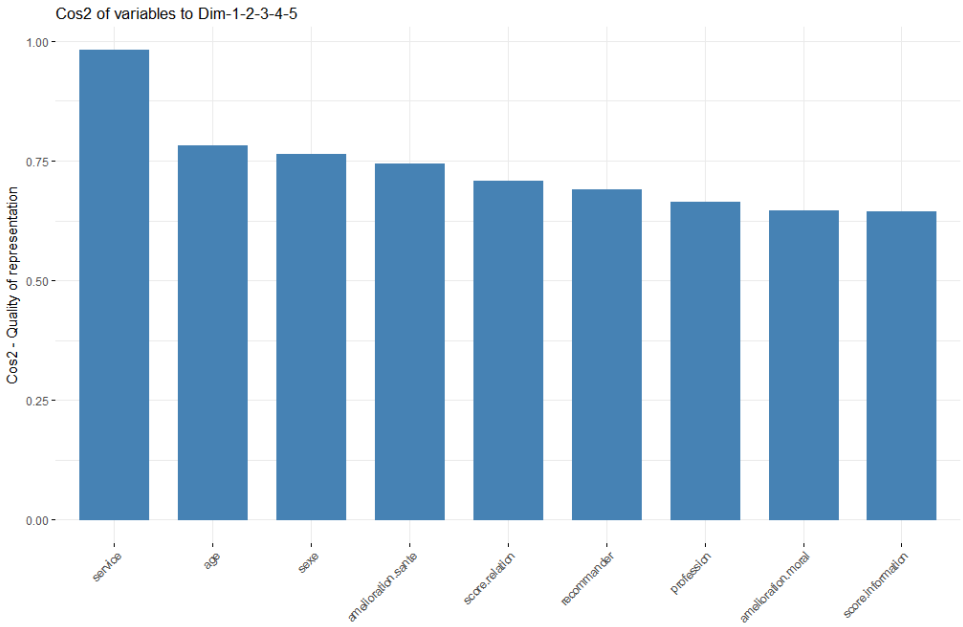


Fig. 1.9. Les diagrammes représentant la qualité de la représentation des variables dans toutes les 5 composantes principales

1.4.2 La sélection de variables (Feature selection)

Il existe deux approches principales en matière de sélection variable : les méthodes par régression (en anglais : all possible regressions) et les méthodes automatiques.

L'approche par régression

Cette approche tient en compte tous les sous-ensembles possibles du groupe de variables explicatives et trouve le modèle qui correspond le mieux aux données selon certains critères (par exemple, R2 ajusté, AIC et BIC). Ces critères assignent des scores à chaque modèle et nous permettent de choisir le modèle avec le meilleur score.

La fonction **regsubsets()** dans le package **leaps**[9] peut être utilisée pour la sélection de sous-groupe de régression. Par la suite, nous pouvons voir les modèles classés selon différents critères de notation en traçant les résultats des **regsubsets()**.

```
R>library(leaps)
R> leaps=regsubsets(recommander~.,data=base,nbest=9)
```

9. https ://cran.r-project.org/web/packages/leaps/index.html

Pour afficher les modèles classés selon les critères R2 ajusté et BIC, respectivement, nous utilisons les instructions :

```
R> plot(leaps, scale="adjr2")
R> plot(leaps, scale="bic")
```

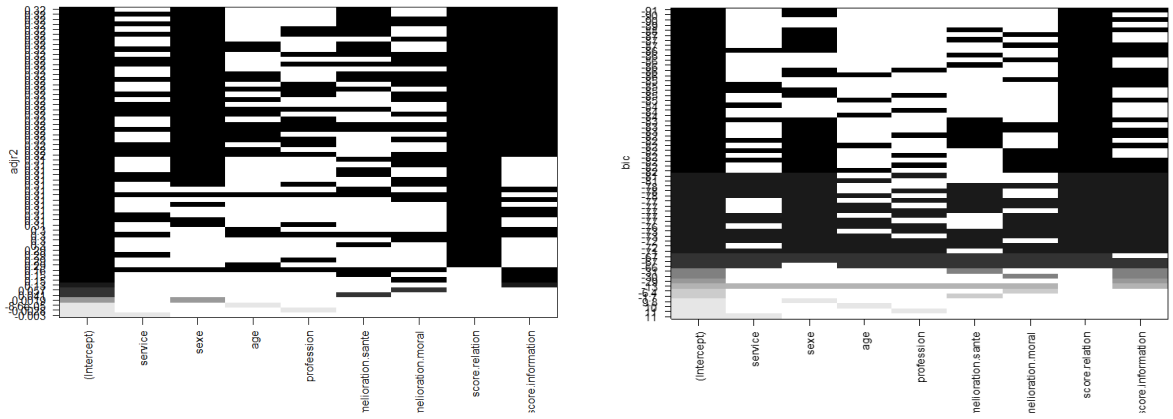


Fig. 1.10. Les résultats de la méthode Best subset regression par le critère de R2 ajusté et BIC.

Dans la Figure1.10, le noir indique qu'une variable est incluse dans le modèle, tandis que le blanc indique qu'ils ne le sont pas. Le modèle contenant toutes les variables minimise les critères R2 ajusté (à gauche), tandis que le modèle incluant les variables « sexe, score d'information et amélioration santé » minimise le BIC (à droite).

Les méthodes automatiques sont utiles lorsque le nombre de variables explicatives est important et qu'il n'est pas possible d'adapter tous les modèles possibles. Dans ce cas, il est plus efficace d'utiliser un algorithme de recherche (par exemple, la sélection de l'avant, l'élimination vers l'arrière et la régression par étapes) pour trouver le meilleur modèle.

La fonction **step()** peut être utilisée pour réaliser une sélection de variable. Pour effectuer une sélection en avant (Forward), nous devons commencer par spécifier un modèle de départ et la gamme de modèles que nous voulons examiner dans la recherche.

```
R> null=lm(recommander~1, data=newdata)
R> full=lm(recommander~., data=newdata)
R> step(null, scope=list(lower=null, upper=full),direction="forward")
```

Nous pouvons effectuer une élimination vers l'arrière (backward) sur le même ensemble de données en utilisant la commande :

```
R > step(full, data=newdata, direction="backward")
```

Et stepwise regression :

```
R > step(null, scope = list(upper=full),data=base,direction="both")
```

Les deux algorithmes donnent lieu à des résultats équivalents à la procédure de sélection vers l'avant en donnant comme meilleur modèle celui incluant les variables score.relation + sexe + score.information + amelioration.sante.

1.5 Travail demandé

Après le tirage au sort réalisé au sein du laboratoire, on désignera pour chaque binôme/étudiant une base de données sur laquelle il va travailler tout au long des travaux pratiques de fouille de données. Aussi, dans ce premier TP, il est demandé de réaliser les pré-traitements nécessaires pour la préparation de la BDD.

1. Présenter la base de données attribuée,
2. Procéder au chargement de la base de données,
3. Réaliser un aperçu des données, effectuer des transformations/codage/traitement des données manquantes si nécessaires.
4. Traiter les données manquantes à l'aide du package **MLR** [10] et de la fonction **impute()** pour appliquer la méthode d'imputation médiane / mode ; c'est-à-dire que les valeurs manquantes dans la variable entière seront imputées avec la médiane et les variables factorielles seront imputées avec le mode (valeur la plus fréquente).
5. Effectuer une réduction de dimension de la base par l'application de l'Analyse en composantes principales.
   a) Analyser le nuage des variables (cercle des corrélations) : Quelles sont les variables qui semblent bien représentées ? Quelles sont les variables moins bien représentées ? Quelles sont les variables paraissant assez fortement corrélées à chacun des axes ? Tenter d'expliquer qualitativement l'information portée par chacun des deux premiers axes de l'ACP.
   b) Si vous êtes en présence de données qualitative, commenter le positionnement de la variable supplémentaire. Que peut-on en déduire qualitativement ?
   c) Analyser le nuage d'individus : Quel est son aspect général ? Des groupes semblent-ils se former ? Peut-on identifier des individus mal représentés ? y a t-il des individus qui paraissent proches sur le plan factoriel (1,2) mais qui n'étaient pas forcément très proches dans le nuage initial ?
   d) Quelles sont les variables qui contribuent pour la composition de chaque composante principale ?
   e) Si vous pouvez déduire les variables les plus pertinentes, donnez le classement de leur importance.
6. Réaliser une sélection de variables par les approches par régression et celles automatiques.
7. Analyser les résultats de chaque approche, que remarquez-vous ?
8. Faites une synthèse générale des pré-traitements qui ont été nécessaire pour la préparation de la base de données.

Conseils pour le compte-rendu

Il est demandé de suivre les étapes suivantes pour résoudre chaque étape demandée :

1. Lire attentivement les différentes phases du TP.
2. Expliquer l'approche suivie pour la résolution du problème.
3. Commenter le listing du programme utilisé.
4. Illustrer, graphiquement ou à l'aide d'un tableau, les résultats obtenus.
5. Expliquer les résultats obtenus. Cette étape est bien entendue importante, et elle est souvent négligée.
6. Nous n'exigeons pas obligatoirement un commentaire très long, mais au moins quelques remarques pertinentes.

10. https ://cran.r-project.org/web/packages/mlr/index.html

2

TP2 : Les méthodes de structuration et de classification en apprentissage non supervisé

Introduction

Relativement lié à la reconnaissance des formes, l'apprentissage non supervisé consiste à analyser les données et à rechercher des modèles. C'est un outil extrêmement puissant pour identifier la structure des données.

Le clustering ou partitionnement de données en français, est une méthode de classification non supervisée où les classes des partitions ne sont pas prédéfinies, c'est-à-dire non inclues dans le jeu de données d'origine.

Le principe du clustering est de maximiser l'inertie interclasse ou, ce qui revient au même, de minimiser l'inertie intraclasse, sous la contrainte d'obtenir un nombre de clusters compatible avec l'objectif qu'on s'est donné. Sans cette contrainte, on obtiendrait autant de classes que d'individus.

Il existe deux grands types du clustering :

- **Le clustering non-hiérarchique** de division («top-down») : on décompose l'ensemble d'individus en K groupes, les algorithmes de ce type peuvent aussi être utilisés comme algorithmes de division dans le clustering hiérarchique.
- **Le clustering hiérarchique** d'agglomération («bottom-up») : on décompose l'ensemble d'individus en une arborescence de groupes.

2.1 Objectif

Ce travail pratique se concentre sur la façon dont vous pouvez utiliser les approches d'apprentissage non supervisées pour trouver une structure dans les données non labellisées. Le but de ce TP est d'expérimenter les différents algorithmes de clustering afin d'acquérir des notions sur : le calcul de distance, la classification par l'algorithme k-means, la classification ascendante hiérarchique et choix de distances entre classes, la construction du dendrogramme, choix du nombre de classes.

2.2 L'algorithme des K-moyennes

Les K-moyennes ou K-Means (MacQueen [13]) est un algorithme d'apprentissage non supervisé qui tente de regrouper les données en fonction de leur similitude. Dans l'algorithme des K-moyennes, nous devons spécifier le nombre de clusters dans lesquels nous

voulons que les données soient regroupées. L'algorithme attribue de manière aléatoire chaque observation à un cluster et trouve le centroïde de chaque clustering. Ensuite, l'algorithme itère en deux étapes :

- Réaffectez les points de données au groupe dont le centre de direction est le plus proche.
- Calculez le nouveau centroïde de chaque grappe.

Ces deux étapes sont répétées jusqu'à ce que la variation interne du cluster ne soit plus réduite (voir figure Fig.2.1). La variation de cluster est calculée comme la somme de la distance euclidienne entre les points de données et leurs centres de concentration des clusters respectifs.

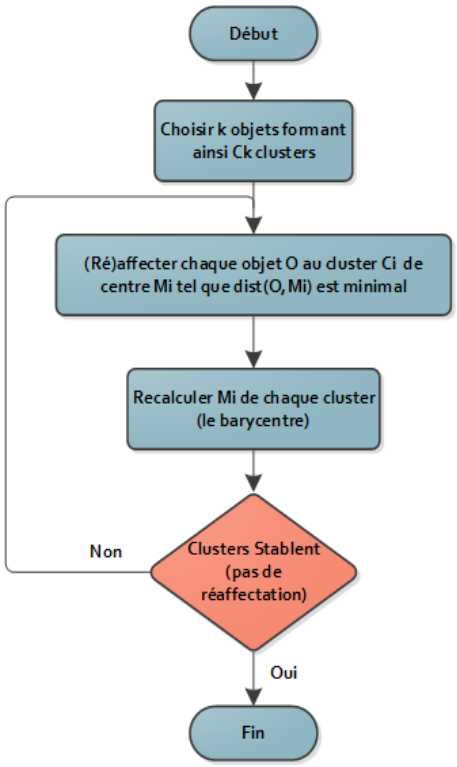


Fig. 2.1. Organigramme du principe de fonctionnement de l'algorithme K-moyennes

Préalablement à l'étape de partitionnement des données, il est indispensable de réaliser un pré-traitement de la base, par l'élimination ou traitement des données manquantes, codage, la transformation des variables qualitatives, réduction des données, etc.

```
R > base= read.csv ("C:\\...\\Documents\\satisfaction_hopital.csv",
sep=",",header=TRUE)
R > summary(base)
R > # suppression des données manquantes
R> newdata <- na.omit(base)
R> # la mise à l'échelle/standardisation des données
R> newdata <- scale(newdata)
R> head(newdata)
R> # Cas d'une base labellisée, suppression des classes
R> newdata2 =newdata
R> newdata2$class <-NULL
```

Avant d'aborder l'implémentation de l'algorithme des k-moyennes sous R, nous devons répondre à une question fondamentale qui est *« comment choisir le nombre de clusters (K) ? »* Le principe du partitionnement des données consiste à faire en sorte que les

groupes soient regroupés de manière homogène dans les clusters et de manière distincte des autres groupes.

Il n'existe aucune formule mathématique qui peut nous donner directement une réponse au choix de "K", mais c'est un processus itératif où nous devons exécuter plusieurs itérations avec différentes valeurs de "K" et choisir celles qui répondent le mieux à notre objectif.

```
R> wss <- (nrow(newdata)-1)*sum(apply(newdata,2,var))
R> for (i in 2:20)
 wss[i] <- sum(kmeans(newdata,centers=i)$withinss)
R> plot(1:20, wss, type="b", xlab="Number of Clusters",
ylab="Within groups sum of squares")
```

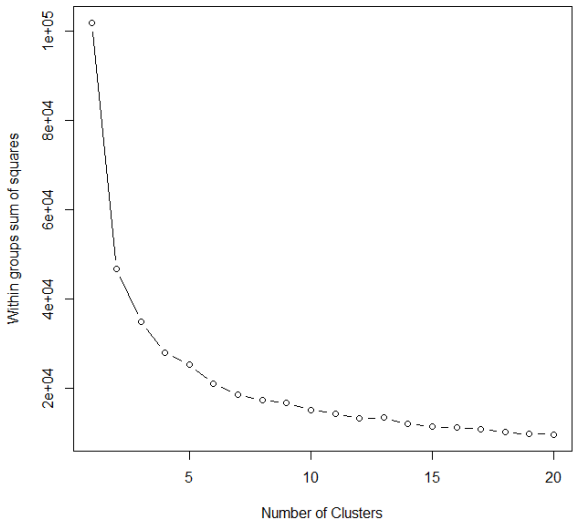


Fig. 2.2. Choix du nombre de cluster

De la Figure 2.2, nous pouvons conclure que si nous conservons un nombre de clusters = 5, nous devrions pouvoir obtenir de bons clusters avec une bonne homogénéité en eux-mêmes.

Il existe la fonction **NbClust()** du package du même nom [1], qui offre à l'utilisateur le meilleur schéma de regroupement parmi les différents résultats.

```
R> library(NbClust)
R> nc <- NbClust(newdata, min.nc=2, max.nc=20, method="kmeans")
```

1. https ://cran.r-project.org/web/packages/NbClust/index.html

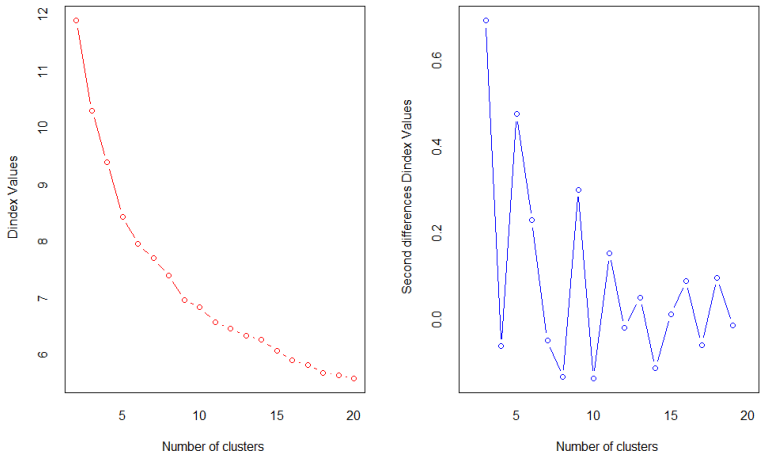


Fig. 2.3. Représentation graphique pour déterminer le nombre de clusters par l'indice D (gauche) et Hubert (droite).

```
R> table(nc$Best.n[1,])
R> barplot(table(nc$Best.n[1,]), xlab="Numer of Clusters",
 ylab="Number of Criteria",main="Number of Clusters Chosen by 20 Criteria")
```

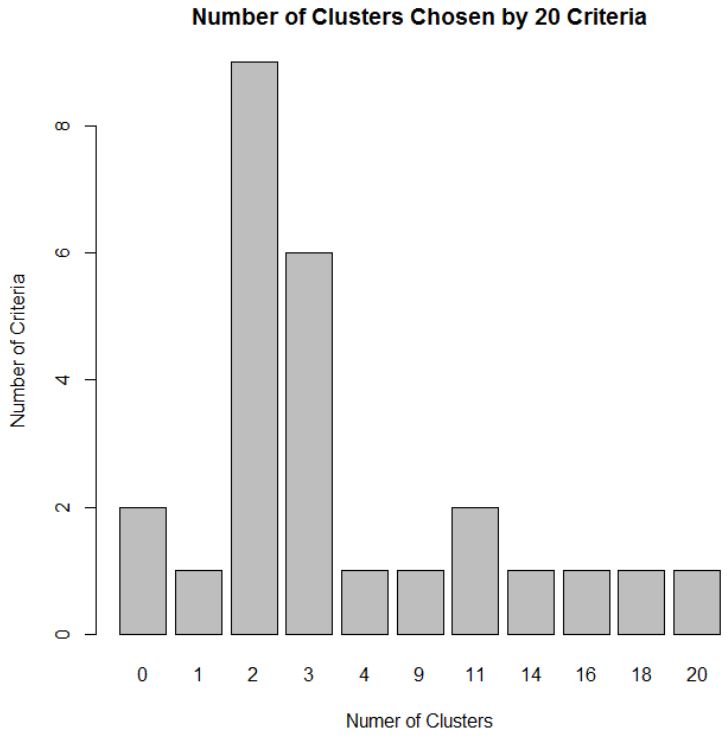


Fig. 2.4. Le nombre recommandé de cluster utilisant les 20 critères fournis par le package NbClust.

Dans notre cas applicatif, nous fixons la taille du cluster à "2" et appelons la fonction **kmeans()** pour réaliser le partitionnement en clusters.

```
R> # K-Means Cluster Analysis
R> fit <- kmeans(newdata, 2) # 2 clusters solution
R> # obtenir les moyennes des clusters
R> aggregate(newdata,by=list(fit$cluster),FUN=mean)
R> # Ajouter l'affectation à chaque cluster
R> mydata <- data.frame(newdata, fit$cluster)
R> library(cluster)
R> clusplot(mydata, fit$cluster, color=TRUE,shade=TRUE, labels=2, lines=0)
```

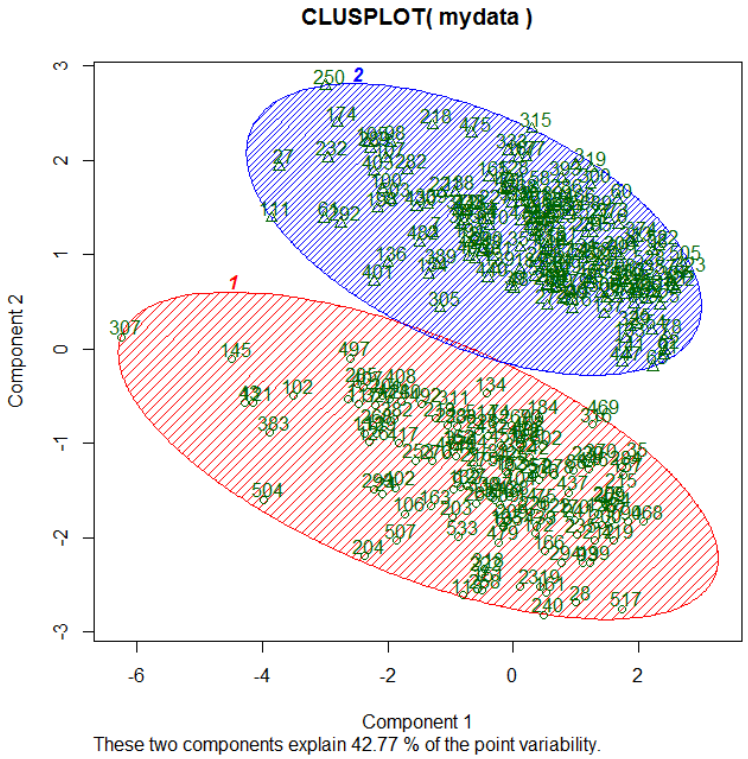


Fig. 2.5. Representation des clusters

La fonction **cluster.stats()** dans le package **fpc** [2] fournit un mécanisme pour comparer la similarité de deux solutions de cluster en utilisant une variété de critères de validation comme : le coefficient gamma d'Hubert, l'indice de Dunn et l'indice de rand ajusté.

```
R> library(fpc)
cluster.stats(d, fit1$cluster, fit2$cluster)
```

Où *d* est une matrice de distance parmi les objets,

2. https ://cran.r-project.org/web/packages/fpc/index.html

```
# distance matrix
R> d <- dist(mydata, method = "euclidean")
```

fit1$cluster, fit2$cluster : sont des vecteurs entiers contenant les résultats de classification de deux groupes différents des mêmes données.

Il est aussi possible de réaliser un tableau croisé des résultats lorsqu'on travaille sur une base labellisée au préalable.

```
R> ct.km <- table(newdata$class, fit$cluster)
```

D'une autre manière, nous pouvons quantifier l'accord entre le vecteur classes et cluster, en utilisant un index de rand ajusté fourni par le package **flexclust** [3].

```
R> library(flexclust)
R> randIndex(ct.km)
```

L'indice Rand ajusté fournit une mesure de l'accord entre deux partitions. Il varie de -1 (pas d'accord) à 1 (accord parfait). A partir de l'application des k-moyennes sur notre exemple de données, nous pouvons dire que cette méthode est l'une des premières et les plus basiques techniques de regroupement. Cependant, cette technique n'est pas seulement puissante, mais elle nous renseigne également sur l'importance de comprendre les données dans un apprentissage non supervisé. Si l'une des hypothèses n'est pas respectée, les clusters ne sont pas correctement formés. De même, pour le choix de la valeur 'k', l'utilisation de valeurs incorrectes ou arbitraires peut conduire à des clusters incorrects. D'une certaine façon, le k-means repose également sur le bon choix de k pour regrouper les données avec précision.

2.3 Classification hiérarchique

Le regroupement hiérarchique peut être divisé en deux principaux types : agglomératif et divisif.

2.3.1 Le regroupement agglomératif

Également connu sous le nom d'AGNES (Agglomerative Nesting). Il fonctionne de manière ascendante. Autrement dit, chaque objet est initialement considéré comme un cluster à élément unique (feuille). À chaque étape de l'algorithme, les deux clusters les plus similaires sont combinés dans un nouveau cluster plus important (nœuds). Cette procédure est itérée jusqu'à ce que tous les points appartiennent à un seul grand groupe (racine). Le résultat est un arbre qui peut être tracé en tant que dendrogramme.

2.3.2 Le regroupement divisif

Également connu sous le nom de DIANA (Divise Analysis) et fonctionne de manière descendante. L'algorithme est un ordre inverse d'AGNES. Il commence par la racine, dans laquelle tous les objets sont inclus dans un cluster unique. À chaque étape d'itération, le groupe le plus hétérogène est divisé en deux. Le processus est itéré jusqu'à ce que tous les objets se trouvent dans leur propre cluster.

3. https ://cran.r-project.org/web/packages/flexclust/index.html

Il est à noter que le regroupement agglomératif est un excellent candidat pour identifier un petit nombre de clusters. Le regroupement divisif est quant à lui meilleur candidat pour identifier de un grand nombre clusters.

Comme nous l'avons vu avec l'approche des k-moyennes, nous mesurons la similitude entre les observations en utilisant des mesures de distance comme : distance euclidienne, distance de Manhattan, etc.). Cependant, dans la classification hiérarchique, une question peut se poser est : *comment mesurons-nous la dissemblance entre deux clusters d'observations ?* Un certain nombre de méthodes d'agglomération des différents clusters (c'est-à-dire des méthodes de liaison) ont été développées pour répondre à cette question.

Les types les plus courants sont les suivants :

- **Le regroupement de liaison maximum ou complète :** il calcule toutes les différences entre les éléments du cluster 1 et les éléments du cluster 2 et considère la plus grande valeur (c'est-à-dire la valeur maximale) de ces dissemblances comme la distance entre les deux clusters. Il a tendance à produire des clusters plus compacts.

- **Le regroupement de liens minimum ou unique :** il calcule toutes les divergences par paires entre les éléments du cluster 1 et les éléments du cluster 2 et considère la plus petite de ces dissemblances comme un critère de liaison. Il a tendance à produire des clusters longs et " en vrac".

- **Le regroupement moyen ou moyenne de liaison :** il calcule toutes les différences entre les éléments du cluster 1 et les éléments du cluster 2 et considère la moyenne de ces dissemblances comme la distance entre les deux clusters.

- **Le clustering de liaison des centroïdes :** il calcule la dissemblance entre le centroïde pour le cluster 1 (vecteur moyen des variables de longueur p) et le centroïde pour le cluster 2.

- **Méthode de variance minimale de Ward :** elle minimise la variance totale du groupe. À chaque étape, la paire de grappes avec une distance intermédiaire minimale est fusionnée.

Il existe différentes fonctions disponibles dans R pour calculer le regroupement hiérarchique. Les fonctions couramment utilisées sont : **hclust()** dans le package **stats**[4], **diana()** et **agnes()** dans le package **cluster**[5] pour le clustering hiérarchique divisif et agglomératif respectivement.

2.3.3 Le clustering hiérarchique agglomératif

Nous pouvons effectuer le regroupement agglomératif avec la fonction **hclust()**. Tout d'abord, nous calculons les valeurs de dissimilarité avec **dist()**, puis alimentons ces valeurs dans l'hclust et spécifions la méthode d'agglomération à utiliser (c'est-à-dire "complete", "average", "single", "ward.D"). Nous pouvons ensuite tracer le dendrogramme.

```
R> # matrice de dissimilarité
R> d <- dist(newdata, method = "euclidean")
R> # Le clustering hierarchique par la méthode Complete Linkage
R> hc1 <- hclust(d, method = "complete" )
```

4. https ://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html
5. https ://cran.r-project.org/web/packages/cluster/index.html

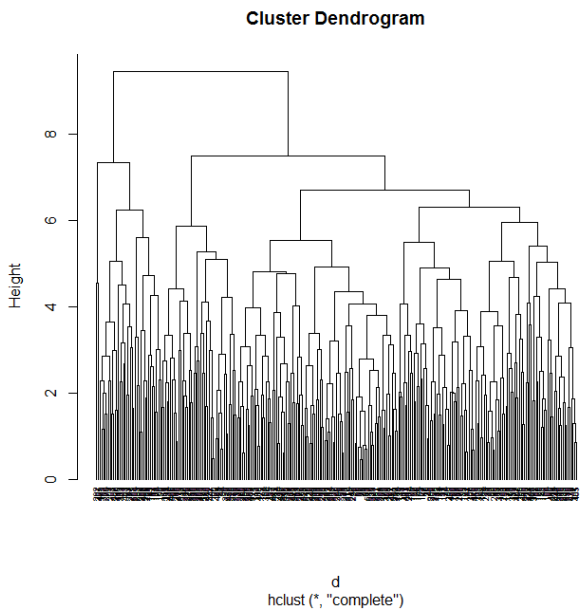


Fig. 2.6. Dendrogramme pour la classification hiérarchique agglomérative.

Alternativement, nous pouvons utiliser la fonction **agnes()**. Ces fonctions se comportent de manière très similaire ; cependant, avec la fonction agnes(), vous pouvez également obtenir le coefficient agglomératif, qui mesure la quantité de structure de regroupement trouvée (des valeurs proches de 1 qui suggèrent une forte structure de regroupement).

```
R> hc2 <- agnes(newdata, method = "complete")
R> hc2$ac
```

Nous pouvons effectuer cette évaluation pour toutes les approches, cela nous permet de trouver les méthodes de regroupement hiérarchiques qui peuvent identifier des structures de cluster les plus fortes.

2.3.4 Le clustering hiérarchique divisif

La fonction R **diana()**, nous permet d'effectuer un clustering hiérarchique divisif. Diana fonctionne comme Agnes ; Cependant, il n'existe aucune méthode à fournir.

```
R> hc2 <- diana(newdata)
R> hc2$dc
R> pltree(hc2, cex = 0.6, hang = -1,main ="Dendrogram of diana")
```

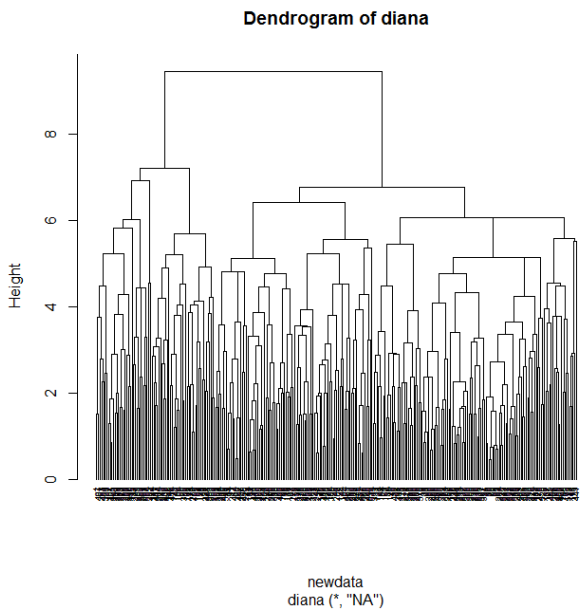


Fig. 2.7. Dendrogramme pour la classification hiérarchique divisive.

2.3.5 Analyse des dendrogrammes

Dans les dendrogrammes, chaque feuille correspond à une observation. Au fur et à mesure que nous avançons dans l'arbre, des observations similaires sont combinées en branches, qui sont elles-mêmes fusionnées à une hauteur plus élevée.

La hauteur de la fusion, fournie sur l'axe vertical, indique la (dis)similitude entre deux observations. Plus la hauteur de fusion est élevée, moins les observations sont similaires. Il est à noter que les conclusions sur la similarité de deux observations ne peuvent être établies qu'en fonction de la hauteur où les branches contenant ces deux observations sont fusionnées au préalable. Nous ne pouvons utiliser la proximité de deux observations le long de l'axe horizontal comme un critère de leur similitude.

La hauteur de la coupe au dendrogramme contrôle le nombre de clusters obtenus. Il joue le même rôle que k dans l'algorithme des K-moyennes. Afin d'identifier les sous-groupes (c.-à-d. les clusters), nous pouvons couper le dendrogramme par l'utilisation de la fonction **cutree()**.

```
R> # méthode Ward's
R> hc2 <- hclust(d, method = "ward.D2" )
R> # couper le dendrogramme en 2 groupes
R> sub_grp <- cutree(hc2, k = 2)
R> # Nombre d'observations dans chaque cluster
R> table(sub_grp)
```

Il est également possible de dessiner le dendrogramme avec une bordure autour des 2 clusters.

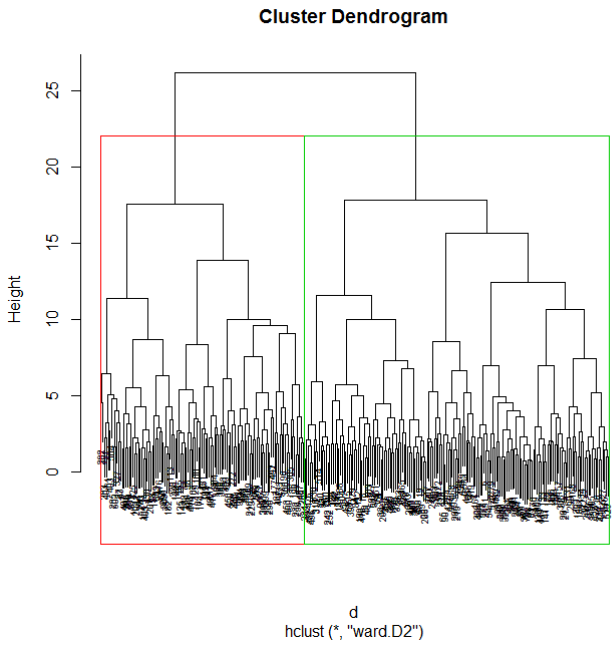


Fig. 2.8. Dendrogramme avec une bordure autour des 2 clusters.

```
R> plot(hc2, cex = 0.6)
R> rect.hclust(hc2, k = 2, border = 2:5)
```

Nous pouvons également utiliser la fonction **fviz_cluster()** du package **factoextra** [6] pour visualiser le résultat dans un diagramme de dispersion.

Fig. 2.9. Diagramme de dispersion

6. https ://cran.r-project.org/web/packages/factoextra/index.html

Le regroupement peut être un outil très utile pour l'analyse des données dans le cadre non supervisé. Cependant, il existe un certain nombre de problèmes qui se posent lors de la mise en forme des clusters. Dans le cas du regroupement hiérarchique, nous devons nous préoccuper :

1. Quelle mesure de dissemblance devrait être utilisée ?
2. Quel type de liaison devrait-on utiliser ?
3. Où devrions-nous réduire le dendrogramme pour obtenir des clusters ?

Chacune de ces décisions peut avoir un fort impact sur les résultats obtenus. Dans la pratique, nous essayons plusieurs choix différents, et nous recherchons celui avec la solution la plus optimale ou interprétable. Avec ces méthodes, il n'y a pas une seule réponse correcte : toute solution qui expose certains aspects intéressants des données devrait être considérée.

2.4 Travail demandé

1. Lancez sur votre base de données l'algorithme des k-moyennes. Quel est votre choix de K ? justifiez ?
2. Effectuez une classification hiérarchique agglomérative avec les quatre méthodes "complete", "average", "single", "ward.D". Comparez les résultats ?
3. Réalisez une classification hiérarchique divisive, comparez avec ceux obtenus par l'approche agglomérative et celle des k-moyennes.
4. Effectuez une coupe des dendrogrammes du regroupement agglomératif et divisif, que remarquez-vous ?
5. Lancez sur votre base de données l'algorithme des k-moyennes et la classification hiérarchique agnes et diana, mesurez les temps de calcul et comparez. Qu'en concluez-vous ?

3

TP3 : Les méthodes de structuration et de classification en apprentissage supervisé

Introduction

L'apprentissage supervisé est simplement une formalisation de l'idée d'apprendre à partir d'exemples. Dans l'apprentissage supervisé, l'apprenant (généralement, un programme informatique) apprend avec deux ensembles de données, un ensemble d'apprentissage et un ensemble de tests. L'idée est que l'apprenant "apprend" à partir d'un ensemble d'exemples étiquetés dans l'ensemble de formation afin qu'il puisse identifier avec la plus grande précision possible des exemples non étiquetés dans l'ensemble de tests.

En plus simple, l'objectif de l'apprenant est d'élaborer une règle, un programme ou une procédure qui classe de nouveaux exemples (dans l'ensemble de tests) en analysant les exemples auxquels une étiquette de classe a été déjà prédéfinie.

Un certain nombre de méthodes d'apprentissage supervisées ont été introduites au cours de la dernière décennie. Les algorithmes comme les arbres de décision classiques (Breiman, 1984[6]) sont très populaires, utilisés depuis des décennies, leur succès est dû à leur principe assez intuitif et la représentation visuelle des résultats, les rendant facilement interprétables. Les arbres de décision font partie de la catégorie des algorithmes supervisés, ils permettent de prédire une valeur (prédiction) ou une catégorie (classement).

Les variations modernes de ces algorithmes ont donné naissance à la méthode des forêts aléatoires (Breiman, 2001[5]) qui sont parmi les techniques les plus puissantes disponibles. Les forêts aléatoires sont composées (comme le terme "forêt" l'indique) d'un ensemble d'arbres décisionnels. Ces arbres se distinguent les uns des autres par le sous-échantillon de données sur lequel ils sont entraînés. Ces sous-échantillons sont tirés au hasard (d'où le terme "aléatoire") dans le jeu de données initial.

3.1 Objectif

L'objectif de ce TP est la mise en œuvre de méthodes de classification supervisée, l'accent étant mis sur l'application des arbres de décision et leurs évolutions en Forêt aléatoire, la mesure de leurs performances selon la nature des échantillons, l'utilisation des approches de validation pour l'évaluation des modèles construit avec R, l'interprétation des résultats.

3.2 L'arbre de décision CART (Classification And Regression Tree)

L'algorithme CART [6] dont l'acronyme signifie "Classification And Regression Trees", construit un arbre de décision d'une manière récursive. A chaque étape de la récursion, il calcule parmi les attributs restant pour la branche en cours, celui qui maximisera le gain d'information. C'est-à-dire l'attribut qui permettra le plus facilement de classer les exemples à ce niveau de cette branche de l'arbre. L'arbre de décision généré par CART est binaire et le critère de segmentation est l'indice de Gini.

À un attribut binaire correspond un test binaire. À un attribut qualitatif ayant n modalités, on peut associer autant de tests qu'il y a de partitions en deux classes, soit 2^{n-1} tests binaires possibles. Enfin, dans le cas d'attributs continus, il y a une infinité de tests envisageables. Dans ce cas, on découpe l'ensemble des valeurs possibles en segments, ce découpage peut être fait par un expert ou fait de façon automatique.

Nous nous intéressons pour notre étude de cas à la base de données *«Heart Disease»* qui contient des données de 462 patients pour lesquels on souhaite prédire l'exposition à un infarctus. Les détails concernant cette base de données sont dans l'annexe C.

```
R> base=read.table("C:\\...\\Documents\\heart.data.txt, sep=",",
header= TRUE,row.names=1)
```

3.2.1 Echantillonnage : Apprentissage vs Test

Comme pour tout modèle, nous avons besoin de construire l'arbre de décision sur une base d'apprentissage et de la tester ensuite sur une base de test. La librairie rpart pour la construction d'arbre de décision inclus de la validation croisée mais il est toujours préférable de calculer la performance sur un échantillon qui n'est pas impliqué dans le calcul. De ce fait, nous commençons par séparer nos données en 2 échantillons :

```
R># Tirage aléatoire et sans remise avec 65\% d'individus
R> d = sort(sample(nrow(base), nrow(base) * 0.65))
R> appren <- base[d, ] # Echantillon d'apprentissage
R> test <- base[-d, ] # Echantillon de test
R> summary(appren)
```

3.2.2 Construction de l'arbre de decision

Le package **rpart** [1] est dédié aux arbres de décision. Nous allons construire un modèle d'arbre de décision grâce à la commande suivante :

```
R> library(rpart)
R > arbre = rpart(chd~., method="class",minsplit=20, xval=300, data=appren)
R> summary(arbre)
```

arbre	est le nom choisi pour l'objet R qui contiendra les résultats ;
rpart	la commande permettant de créer un arbre de décision ;

1. https ://cran.r-project.org/web/packages/rpart/index.html

chd~.	est une formule R, la tilde signifie « est à expliquer en fonction de » : on met donc à gauche de la tilde la variable que l'on souhaite expliquer (ici, *chd*) et à droite les variables explicatives séparées par des +. Dans notre cas, nous avons mis un point à droite : cela signifie « tout le reste ».
method = "class"	argument totalement optionnel, mais il permet de rappeler à R que la variable à expliquer est qualitative (c'est une sécurité supplémentaire) ;
minsplit=20	signifie qu'il faut au moins 20 individus dans un nœud pour tenter un split ;
xval	permet de régler la stratégie de validation croisée : on y indique le nombre de « portions » du jeu de données à créer pour effectuer la validation croisée. Par exemple, xval = 2 signifiera qu'on divisera le jeu de données en deux parties égales : l'une correspondant à un jeu d'apprentissage, l'autre à un jeu de validation. Ici, l'argument xval = 300 correspond à la stratégie classique de leave-one-out (LOOCV), puisqu'il y a 300 individus dans le jeu de données d'apprentissage ;
data=base	indiquer le nom du jeu de données à utiliser.

Il est à noter que l'arbre de décision construit avec l'algorithme CART peut fonctionner avec tous types de variables : qualitatives, ordinales et quantitatives continues. Il s'agit d'une grande force de cette méthode, qui permet de créer des règles de décision mixant tous types d'information. Mais n'empêche que les données manquantes posent plusieurs problèmes pour la construction de l'arbre de décision. De ce fait, il est toujours nécessaire de prévoir une étape de gestion des données manquantes et réaliser les prétraitements adéquats au jeu de données utilisé.

Les commandes suivantes permettent d'afficher l'arbre de décision sous forme graphique :

```
R> plot(arbre, uniform=TRUE, margin=0.1,main="Arbre de décision")
R> text(arbre, fancy=TRUE, use.n=TRUE,pretty=0, all=TRUE)
R> #ou encore
R> plot(arbre, uniform=TRUE,branch=0.5,margin=0.1,main="Arbre de décision")
R> text(arbre, all=FALSE, use.n=TRUE)
```

La commande **plot()** ne trace que l'armature de l'arbre, et la commande **text()** y ajoute les étiquettes textuelles (dans les nœuds intermédiaires et les feuilles terminales). Ces deux commandes possèdent de très nombreuses options graphiques pour personnaliser l'apparence esthétique de l'arbre : consulter **help(plot.rpart)** et **help(text.rpart)** pour trouver les réglages vous correspondant.

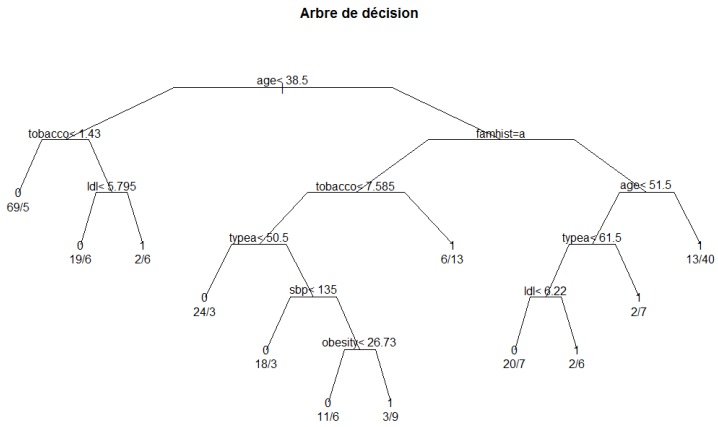


Fig. 3.1. Représentation de l'arbre de décision

3.2.3 Élagage de l'arbre de décision

L'un des inconvénients majeur des arbres de décisions est l'utilisation fréquente des variables moins pertinentes pour l'étape de construction de l'arbre (sur-apprentissage). Ce problème est résolu par l'étape d'élagage qui consiste à supprimer les sous-arbres superflus ou trop liés aux données, dans le but d'améliorer d'une part, l'aspect prédictif de l'arbre, et réduire d'autre part sa complexité. À ce stade, si nous souhaitons élaguer l'arbre de décision, il suffit d'exécuter la commande **plotcp()** pour déterminer la taille optimale :

```
R> plotcp(arbre) # résultats en forme graphique
R> printcp(arbre) # résultats en format tableaux
```

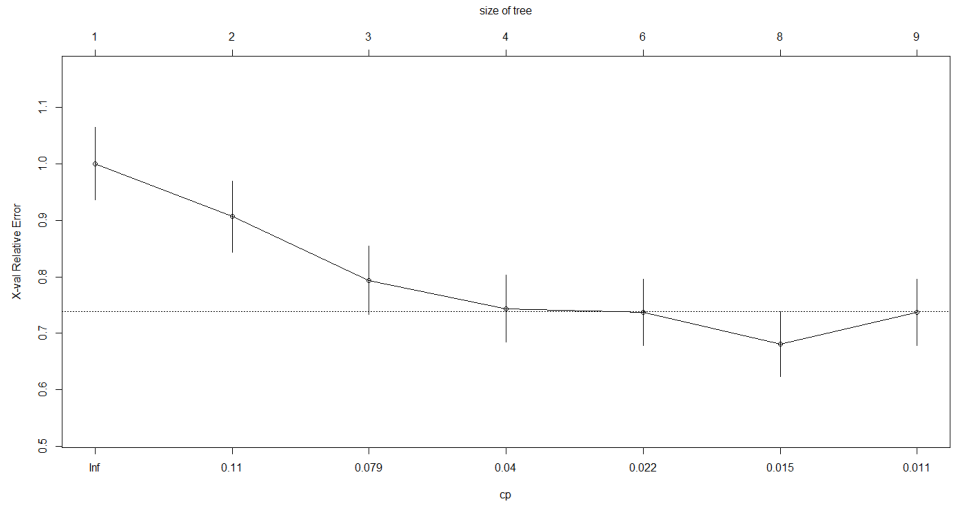


Fig. 3.2. Erreur commise en validation croisée en fonction de la taille de l'arbre

De la Figure 3.2, nous constatons que l'erreur de prédiction commise en validation croisée est minimale si $sizeoftree = 8$, c'est-à-dire si l'arbre ne comporte que huit feuilles terminales. L'arbre correspondant enregistre une complexité $cp = 0.015$: c'est de cette façon qu'on va indiquer à R la taille d'arbre souhaitée, grâce à la commande suivante.

```
R> arbre2 <- prune(arbre,cp=0.0015)
R> plot(arbre2, uniform=TRUE, branch=0.5, margin=0.1,
main="Arbre de décision")
R> text(arbre2, all=FALSE, use.n=TRUE)
```

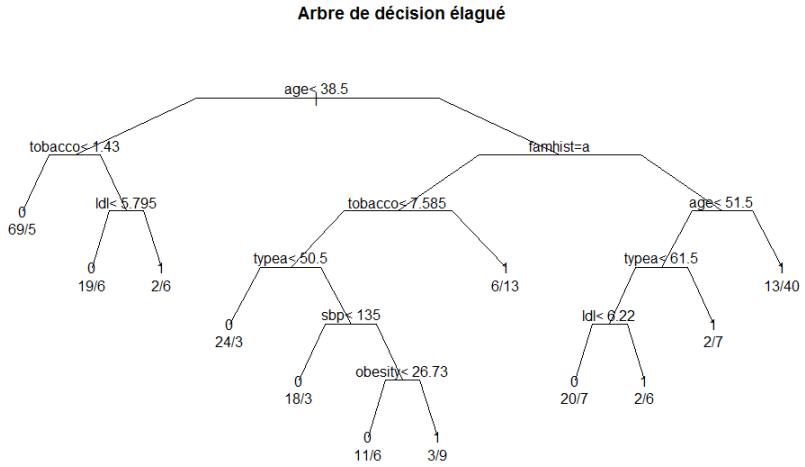


Fig. 3.3. Arbre de décision élagué pour la prédiction de la variable chd.

3.2.4 Évaluation des performances

La prédiction de la variable classe *chd* pour la base de test s'opère par la commande **predict()**, dont le premier argument est le nom du modèle utilisé, et le second, *newdata*, est le nom du data frame contenant les éléments de la base de test :

```
R> test_Predict = predict(arbre2, newdata=test, type="class")
```

Il est capital que le data frame contenant la base de test soit calqué sur celle ayant servi à la construction de l'arbre : mêmes noms de colonnes (classe incluse), mêmes types de variables, etc.

```
R> #Matrice de confusion
R> mc<-table(test$chd,test_Predict)
R> print(mc)
R> #Erreur de classement
R> erreur.classement<-1.0-(mc[1,1]+mc[2,2])/sum(mc)
R> print(erreur.classement)
R> #Taux de prédiction
R> prediction=mc[2,2]/sum(mc[2,])
R> print(prediction)
```

Le logiciel R permet de réaliser de manière très simple la courbe *ROC* [12] Receiver Operating Characteristic (transmission de signal) qui représente l'évolution de la sensibilité (taux de vrais positifs) en fonction de 1 - spécificité (taux de faux positifs).

- C'est une courbe croissante entre le point (0,0) et le point (1, 1) et en principe au-dessus de la première bissectrice.
- Meilleure est la prédiction, plus la courbe est au-dessus de la première bissectrice.
- Une prédiction idéale est l'horizontale y=1 sur]0,1] et le point (0,0).
- L'aire sous la courbe ROC (AUC, Area Under the Curve) donne un indicateur de la qualité de la prédiction (1 pour une prédiction idéale, 0.5 pour une prédiction random).

```
R> library("ROCR")
R> Pred.cart=predict(arbre2,newdata = test,type="prob")[,2]
R> Pred2 = prediction(Pred.cart, test$chd)
R> Perf = performance(Pred2, "tpr", "fpr")
R> plot(Perf, colorize = TRUE, main = "La courbe ROC")
R> abline(0, 1, lty = 2)
```

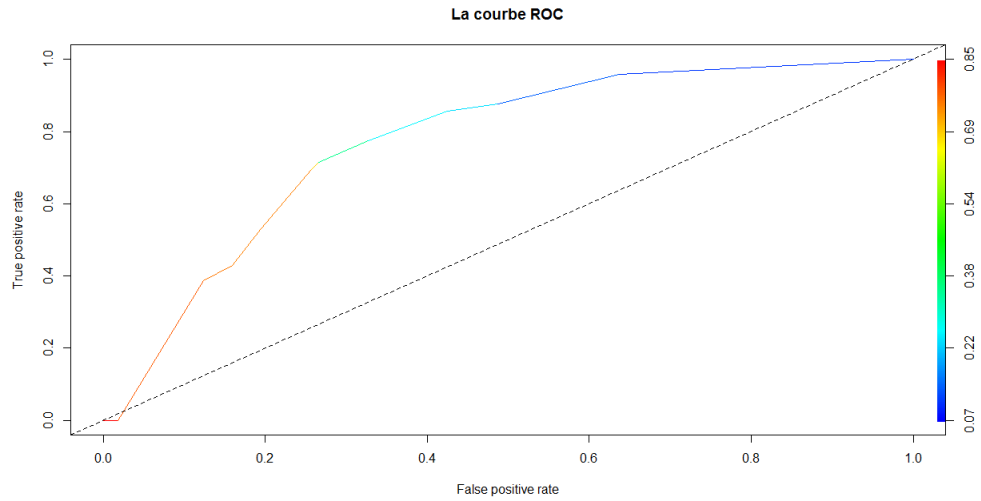


Fig. 3.4. La courbe ROC

Pour mesurer l'aire sous la courbe, nous utilisons les instructions suivantes :

```
R> Perf2 <- performance(Pred2, "auc")
R> Perf2@y.values[[1]]
```

Plus une courbe à des valeurs élevées, plus l'aire sous la courbe est grande, moins le classifieur fait d'erreur.

3.2.5 Règles de classification

Un des autres avantages de l'application des arbres de décision, est qu'il est possible d'afficher les règles de décision et les exploiter.

```
R> print(arbre2)
```

Chaque ligne correspond à un nœud ou à une feuille de l'arbre. On commence par la racine qui contient les individus de l'échantillon d'apprentissage.

3.3 Forêts aléatoires

Cet algorithme appartient à la famille des agrégations de modèles, c'est en fait un cas particulier de bagging (bootstrap aggregating) appliqué aux arbres de décision de type CART. Les forêts aléatoires ou Random Forest ont été inventées par Breiman en 2001 [5]. Dans Random Forest, nous avons une collection d'arbres de décision (appelés «forêts»). On subdivise l'ensemble de données en plusieurs parties par le bootstrap puis on apprend un arbre de décision à partir de chaque partie. Un nouvel exemple est testé par tous les arbres construits et sa classe est la classe majoritaire. Elles sont en général plus efficaces que les simples arbres de décision mais possède l'inconvénient d'être plus difficilement interprétables.

3.3.1 Principe de Boostrapping

Un bootstrap d'un ensemble T est l'ensemble obtenu en tirant $|T|$ fois des éléments de T uniformément au hasard et avec remise. Le bootstrapping d'un ensemble d'entraînement T produit un nouvel ensemble T' qui présente en moyenne 63% instances uniques différentes de T quand $|T| >> 1$ [7, 8](la création de plusieurs bags à partir d'une base de donnée).

Dans les forêts aléatoires, il n'est pas nécessaire de procéder à une validation croisée ou à un ensemble de tests séparé pour obtenir une estimation impartiale de l'erreur de test. Il est estimé en interne. En effet, chaque arbre est construit en utilisant les exemples ré-échantillonnés dans les bags. Environ un tiers des cas sont laissés à l'écart de l'échantillon bootstrap et ne sont pas utilisés dans la construction de l'arbre qui sont nommés OOB= Out Of Bag. L'erreur OOB est la prédiction moyenne sur chaque échantillon d'apprentissage x, en utilisant uniquement les arbres qui n'ont pas x dans leur échantillon bootstrap.

3.3.2 Principe du Bagging

L'idée du Bagging [4] est d'utiliser plusieurs ensembles de données ré-échantillonnées à partir de l'ensemble des données observées et ce à l'aide d'un tirage aléatoire avec remise. Ainsi chaque classifieur élémentaire de l'ensemble sera entraîné sur un des échantillons bootstrap de sorte qu'ils soient tous entraînés sur un ensemble d'apprentissage différent. L'agrégation de ces classifieurs permet d'obtenir un prédicteur plus performant.

3.3.3 Construction de la forêt aléatoire

Chaque arbre dans la forêt aléatoire est construit comme suit :

- Si le nombre de cas dans l'ensemble d'entraînement est N, un échantillon de N cas est pris au hasard mais avec remplacement. Cet exemple sera l'ensemble de formation pour la croissance de l'arbre.
- Si il existe des variables d'entrée M, un nombre $mtry << M$ est spécifié de sorte que, à chaque nœud, $mtry$ variables sont sélectionnées au hasard hors du M et la meilleure division sur ces $mtry$ sert à diviser le nœud. La valeur de m est maintenue constante pendant la croissance de la forêt.
- Chaque arbre est cultivé dans la plus grande mesure possible. Il n'y a pas d'élagage.
- Prédiction de nouvelles données en regroupant les prédictions des arbres $ntree$ (c'est-à-dire les votes majoritaires pour la classification, la moyenne pour la régression).

L'algorithme 1 appartient à une famille plus large des forêts aléatoires définis Breiman [5] (voir figure Fig.**??**).

Algorithm 1 Pseudo code de la création de la forêt aléatoire

Entrée : L'ensemble d'apprentissage L, Nombre d'arbres N.
Sortie : *Ensemble d'arbres* E
1: **Processus :**
2: **for** $i = 1 \rightarrow N$ **do**
3: $T^i \leftarrow BootstrapSample(T)$
4: $C^i \leftarrow ConstructTree(T^i)$ où à chaque noeud :
- Sélection aléatoire de $mtry = \sqrt{M}$ Variables à partir de l'ensemble d'attributs M
- Sélection de la variable la plus informatif K en utilisant l'index de Gini
- Création d'un nœud fils en utilisant cette variable

5: $E \leftarrow E \cup \{C^i\}$
6: **end for**
7: Retourner E

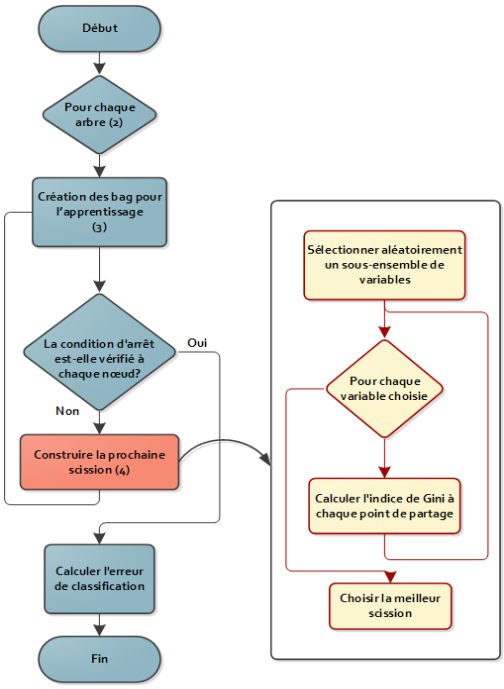


Fig. 3.5. Organigramme du principe de fonctionnement de l'algorithme RF

Le package **randomForest** [2] a la fonction **randomForest()** qui est utilisée pour créer et analyser des forêts aléatoires.

```
R> train = sample(1:nrow(base), nrow(base)*0.7)
R> Training_data =  base[train,]
R>Testing_data =  base[-train,]
R> library(randomForest)
R> set.seed(222) # permet à chaque fois que vous exécutez le modèle,
d'obtenir un échantillon aléatoire différent.
```

L'application de la fonction d'échantillonnage aléatoire seed() rend vos résultats reproductibles la prochaine fois que vous chargez le code, car sinon vous pouvez obtenir différentes classifications pour chaque exécution. Le numéro à l'intérieur n'est pas important, il vous suffit de vous assurer d'utiliser le même nombre dans **seed()** à chaque fois afin que les mêmes nombres aléatoires soient générés dans la fonction **randomforest**.

```
R> RF_Model= randomForest(as.factor(chd)~.,data=Training_data,ntree=500)
```

Au lieu de spécifier **method = "class"** comme avec **rpart**, nous forçons le modèle à prédire notre classification en modifiant temporairement notre variable cible en un facteur avec seulement deux niveaux en utilisant **as.factor()**. L'argument *ntree* spécifie le nombre d'arbres que nous voulons développer.

```
R> print(RF_Model)
```

2. https ://cran.r-project.org/web/packages/randomForest/index.html

```
Call:
 randomForest(formula = as.factor(chd) ~ ., data = Training_data,      ntree = 500)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 29.1%
Confusion matrix:
    0  1 class.error
0 190 28   0.1284404
1  66 39   0.6285714
```

```
R> plot(RF_Model)
```

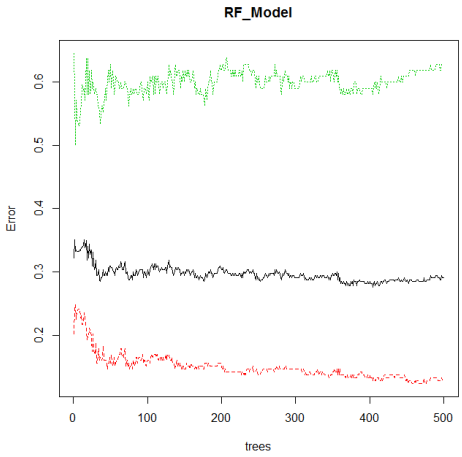


Fig. 3.6. L'erreur OOB pour chaque arbre

Pour identifier le nombre optimal de caractères *mtry* (variables à sélectionner à chaque fractionnement), il est possible d'appliquer la fonction **RFtune()** qui peut nous aider à décider le nombre optimal de variables de scission dans chaque arbre.

```
R> set.seed(222)
R> Try_various_m = tuneRF(Training_data[,-9],Training_data$chd,
ntreeTry=500,stepFactor=2,improve=0.05,  plot=TRUE)
```

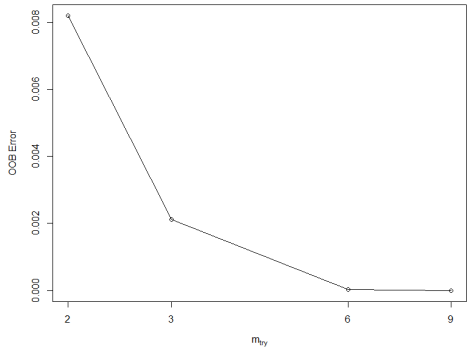


Fig. 3.7. Le graphe de l'erreur OOB en fonction du nombre de Mtry.

Le premier argument dans **tuneRF()** est la donnée contenant des variables indépendantes, la seconde est une variable prédominante, le troisième est le nombre d'arbres, le facteur de démarcation est le pas. Delà nous pourrons relancer l'apprentissage de la forêt aléatoire avec le paramétrage optimal.

```
R> set.seed(222)
R> RF_Model2= randomForest(as.factor(chd)~.,data=Training_data,
mtry=6,importance=T,ntree=500)
R> print(RF_Model2)
```

3.3.4 La mesure des variables d'importance

Lorsque l'ensemble d'apprentissage pour l'arbre actuel est tiré en échantillonnant avec remplacement (boostrapping), environ un tiers des cas sont exclus de l'échantillon. Cette donnée (hors sac ou oob : out of bag) est utilisée pour obtenir une estimation impartie de l'erreur de classification lorsque les arbres sont ajoutés à la forêt mais aussi il est également utilisé pour obtenir des estimations de l'importance des variables.

```
R> importance(RF_Model2)
```

La fonction **importance()** permet de mesurer deux types de mesures d'importance. La précision est testée pour voir si le modèle est toujours performant sans chaque variable, de sorte qu'une forte diminution de précision serait attendue pour les variables très prédictives. Le Gini index est le test mathématique derrière les arbres de décision, il mesure essentiellement la pureté des nœuds à la fin de l'arbre. Encore une fois, on fait appel à ce test pour voir le résultat sur chaque variable, en effet, si elle est retirée et un score élevé est enregistré cela signifie que la variable était importante.

```
R> varImpPlot(RF_Model2)
```

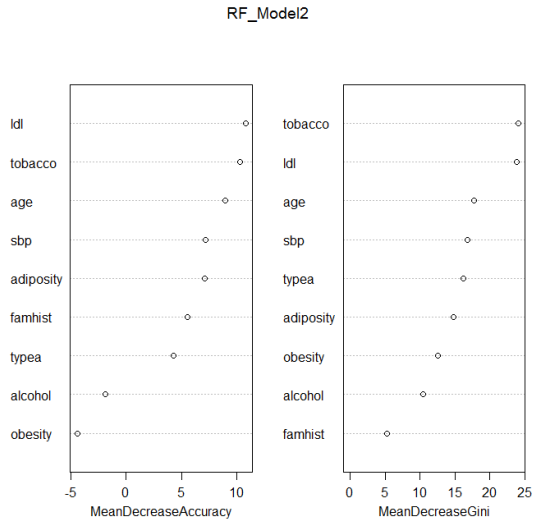


Fig. 3.8. Les mesures d'importance des variables.

La fonction de prédiction fonctionne de manière similaire aux arbres de décision, et nous pouvons construire notre fichier de soumission exactement de la même manière. De ce fait, nous utilisons le modèle optimal pour la prédiction sur les données de test et nous essayons de voir comment sont les prédictions :

```
R> # Pour obtenir la prédiction en termes de probabilité
R> pred_1=predict(RF_Model2,type = "prob",Testing_data)
R> # Pour obtenir la prédiction en termes de réponse "Oui", "Non"
R> pred_2=predict(RF_Model2,type = "response",Testing_data)
R> # Calculer l'erreur de prediction
R> mean(pred_2 !=Testing_data$chd)
R> result = cbind(Testing_data, pred_1, pred_2)
R> result
```

3.3.5 La forêt aléatoire pour le traitement des données manquantes

La bibliothèque randomForest fournit deux façons de traiter les valeurs manquantes : na.roughfix() : traite les valeurs manquantes de manière classique. Pour les valeurs manquantes numériques, il impute avec la médiane de la colonne et pour les valeurs manquantes des caractères, l'imputation arrive avec le mode (la plus grande valeur).

```
R> data_missing = Training_data
R> data_missing[3,5] = NA
R> data_missing[3,6] = NA
R> with_roughfix <- na.roughfix(data_missing)
R> # L'option avec le code de modélisation
R> Model_1 = randomForest(as.factor(chd)~ .,
data_missing, na.action=na.roughfix)
```

Rfimpute() :

l'option permet d'imputer les valeurs manquantes en utilisant le concept de proximité.

```
R> prox_imp <- rfImpute(as.factor(chd) ~ .,
data_missing, iter = 10, ntree = 100)
```

Dans cette deuxième méthode, les valeurs manquantes sont imputées avec la méthode na.roughfix(), puis le système commence à construire le modèle de forêt aléatoire sur les données complètes. La matrice de proximité de la forêt aléatoire est utilisée pour mettre à jour l'imputation des NA.

Pour les prédicteurs continus, la valeur imputée est la moyenne pondérée des observations non manquantes, où les poids sont les proximités. Pour les prédicteurs catégoriques, la valeur imputée est la catégorie avec la plus grande proximité moyenne. Ce processus est itérer un nombre $iter$ fois.

La proximité est la similitude entre les observations mesurées dans les termes de ceux qui tombent dans le même nœud. Donc, si nous disposons de données sur lesquelles nous construisons un modèle de forêt aléatoire avec 500 arbres. Supposons qu'il y ait deux observations "x" et "y" qui tombent dans le même nœud dans 140 arbres, donc leur proximité est de 140. À la fin, les proximités sont normalisées en divisant par le nombre d'arbres (500). La Proximité normalisée = 140/500. Cette deuxième approche est considérée comme une meilleure méthode d'imputation de valeur manquante.

Jusqu'à présent, nous avons vu les capacités de la technique de Random Forest, mais nous devrions aussi connaître ses limites. La technique est une boite noire dans le sens, nous n'avons pas une image juste du modèle comme nous l'avons dans le cas de CART.

Nous ne savons pas ce que les règles ont été dérivées par la technique de classification. Tout ce que nous obtenons à la fin est la prédiction et l'importance des variables, mais pas des règles ou des équations, comme nous l'avons vu dans l'arbre de décision.

3.4 Travail demandé

1. Construire l'arbre de décision de votre base de données de travail. En premier lieu avec une stratégie de validation classique LOOCT Leave One Out puis avec une stratégie de validation croisée. Que remarquez-vous ?
2. Interprétez le graphe de l'arbre généré, quelles sont les règles vérifiées par les experts du domaine ou de manière théorique en général ?
3. Réalisez l'élagage de votre arbre, mesurer les performances, que concluez-vous ?
4. Modélisez la forêt aléatoire sur votre base de données avec les paramètres par défaut.
5. Essayez plusieurs valeurs de $ntree$, $mrty$ pour obtenir de meilleurs résultats. Dessinez la courbe des différentes variations.
6. Consultez le **help(randomForest)** et essayez d'utiliser l'argument additionnel et optionnel $nodesize$ avec lequel vous pouvez élaguer les arbres. Avec cette option, vous assurez le nombre minimum d'observations dans chacun des nœuds terminaux.
7. Mesurez les variables d'importances ? Interprétez vos résultats ? Quelles sont les différences avec une sélection de variables par les approches par régression et celles automatiques (effectuée dans le TP1).
8. Si nécessaire réaliser un traitement des données manquantes par l'approche statistique avec le package **MLR** (appliquée dans le TP1) et celle disponible avec la forêt aléatoire, comparez ?

4

TP4 : Les méthodes descriptives et prédictives

Introduction

Les données stockées peuvent cacher un certain nombre de connaissances, de dépendances ou de corrélations, qui sont implicites et utiles, et qui n'attendent qu'à être explorées. Nous présentons dans cette session pratique deux approches assez récentes de fouille de données qui sont fondées sur la découverte et la recherche de modèles à partir d'un ensemble de données. Ces thèmes sont considérés aujourd'hui comme faisant partie des approches d'apprentissage symbolique non supervisé, utilisé dans le domaine de fouille de données (data mining) et d'extraction de connaissances.

- **Les méthodes descriptives (recherche de « patterns ») :** visent à mettre en évidence des informations présentes mais cachées par le volume des données (c'est le cas des segmentations de clientèle et des recherches d'associations de produits sur les tickets de caisse). Elles ont le potentiel de réduire, résumer, synthétiser les données.

- **Les méthodes prédictives (modélisation) :** visent à extrapoler de nouvelles informations à partir des informations présentes (communément appelées les méthodes de régression) qui permettent d'expliquer les données.

Dans ce TP, nous allons étudier en premier lieu une méthode descriptive connue sous le nom d'analyse des associations et qui est utilisée pour découvrir des associations ou des relations cachées dans les grandes bases de données. Les relations découvertes peuvent être représentées sous forme de règles d'association ou d'ensemble d'items fréquents.

En second lieu, une méthode prédictive connue sous le nom de régression linéaire simple ou multiple utilisée pour l'estimation des valeurs continues. Son objectif est de trouver le meilleur modèle qui décrit la relation entre une variable continue de sortie et une ou plusieurs variables d'entrée. Il s'agit de trouver une fonction f qui se rapproche le plus possible d'un scénario donné d'entrées et de sorties.

4.1 Objectif

Le travail proposé ici consiste, à partir d'un jeu de données, à faire une recherche sur les informations, découvrir les associations et liaisons entre les variables et de les représenter sous forme de règles ou d'items fréquents. Il sera aussi question de trouver un modèle prédictif par rapport à une ou plusieurs variables explicatives.

L'objectif est d'expérimenter l'algorithme APriori de génération de règles d'association et les algorithmes de régression linéaire simple et multiples pour la prédiction.

4.2 Les règles d'association

Etant une méthode d'apprentissage non supervisé, les règles d'association permettent de découvrir à partir d'un ensemble de transactions, un ensemble de règles qui exprime une possibilité d'association entre différents items. Une transaction est une succession d'items exprimée selon un ordre donné ; de même, l'ensemble de transactions contient des transactions de longueurs différentes.

Rechercher les associations consiste à rechercher les règles du type :

« Si pour un individu, la variable $A = xA$, la variable $B = xB$, etc, alors, dans 80% des cas, la variable $Z = xZ$, cette configuration se rencontrant pour 20 % des individus »

La valeur de 80% est appelée indice de confiance et la valeur de 20% est appelée indice de support.

Avec : L'indice de support est la probabilité : Prob (condition et résultat) et L'indice de confiance est la probabilité : *Prob (condition et résultat) / Prob (condition)*

L'amélioration apportée par une règle, par rapport à une réponse au hasard est appelée « **lift** » et vaut :

lift (règle) = confiance (règle) / Prob (résultat)

Si une règle n'est pas utile, on peut donc essayer la règle inverse, en espérant que cette dernière soit intéressante en termes de résultats recherchés.

Les algorithmes d'apprentissage des règles d'association les plus populaires sont : l'Algorithme Eclat et l'Algorithme Apriori. Nous nous intéressons dans ce TP à l'étude de ce dernier.

4.2.1 L'Algorithme Apriori

C'est l'algorithme le plus répandu (Agrawal et al. [1]), la figure Fig.4.1 illustre son principe de fonctionnement qui est en deux étapes :

Etape1 : il commence par rechercher les sous-ensembles d'items ayant une probabilité d'apparition (indice de support) supérieure à un certain seuil s.

- élimination des items moins fréquents que s,
- constitution des combinaisons de deux items parmi les précédents, et élimination des combinaisons moins fréquentes que s (,
- les ensembles fréquents de taille n qui nous intéressent sont ceux provenant d'ensembles de taille $n - 1$ eux-mêmes fréquents.

Etape2 : puis il tente de décomposer chaque sous-ensemble sous une forme *Condition* $\bigcup$ *Résultat* telle que le quotient *«Prob (Condition et Résultat) / Prob (Condition)»* (indice de confiance), soit supérieur à un certain seuil s.

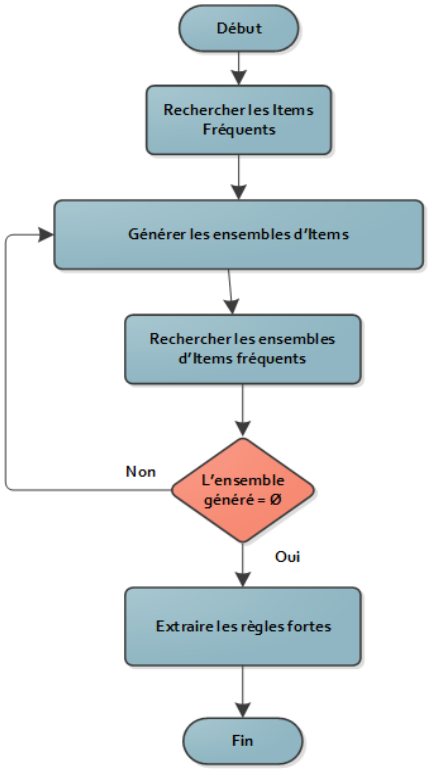


Fig. 4.1. Oraganigramme du principe de fonctionnement de l'algorithme Apriori

4.2.2 Construction des règles d'association

Michael Hahsler, et al.[10] a mis en place deux paquets R très utiles concernant l'extraction des règles d'association : le package **arules** [1] et le package **arulesViz** [2]. En outre, Hahsler a fourni deux très bons exemples d'articles fournissant des détails sur l'utilisation de ces paquets dans Introduction aux **arules** et la Visualisation des règles d'association .

L'algorithme Apriori ne fonctionne que sur des données transactionnelles, de ce fait il est nécessaire de faire un prétraitement (discrétisation) de la base de données. Dans R, les données doivent être sous forme de transaction. Si les données ne sont disponibles que dans un data.frame puis pour créer (ou forcer) le cadre de données à la transaction, il faudra passer par une étape de transaction.

Dans ce TP, nous allons travailler sur la base « *lenses* » du dépôt d'UCI [3], qui contient des données de 24 patients pour lesquels on souhaite prédire le type de lentille de contacte attribué suivant 4 paramètres. Les détails concernant cette base de données sont dans l'annexe D.

```
R> base=read.table("C:\\...\\Documents\\lenses.txt",sep="\t",
header= TRUE,row.names=1)
R> transact <- subset(base,select = c('age','prescription','astigmatic',
'tear_rate', 'Class'))
R> transact$transaction_id <- paste(as.character(transact$age),
```

1. https ://cran.r-project.org/web/packages/arules/index.html
2. https ://cran.r-project.org/web/packages/arulesViz/index.html
3. http ://archive.ics.uci.edu/ml/datasets/Lenses

```
as.character(transact$prescription),as.character(transact$astigmatic),
as.character(transact$tear_rate),as.character(transact$Class),sep =' ')
R> transaction.id <- unique(transact$transaction_id)
R> transaction.trns <- sqldf('select transaction_id,
age from transact group by astigmatic,Class')
R> base_trans <- as.data.frame(t(transaction.trns))
R> # Utilisez split pour créer des données de transaction:
R> lst1 <- split(transaction.trns[,"transaction_id"],
transaction.trns[,"age"])
R> transactions_data <- as(lst1, "transactions")
R> library(arules)
R> library(arulesViz)
R> rules = apriori(transactions_data,parameter=list(support=0.01,
confidence=0.5))
```

Le paramétrage sur seuil de confiance est généralement plus sévère que celui de support, surtout si l'on recherche des règles rares, l'exemple le plus courant de filtre est 75% pour la confiance et 5% pour le support (et bien sûr 1 pour le lift).

```
R> inspect(head(sort(rules, by="lift"),0.5))
R> plot(rules)
```

En pratique, les règles demeurent très nombreuses, et la plupart des logiciels permettent de stocker ces règles dans un fichier, dans lequel il est possible de filtrer les règles selon leur indice de support, leur confiance ou leur lift.

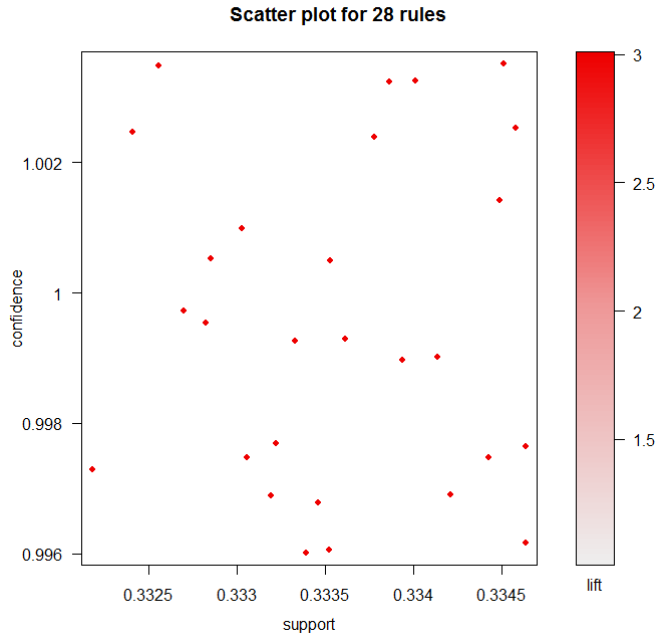


Fig. 4.2. Le graphique des règles d'association générées

```
R> head(quality(rules))
R> plot(rules, measure=c("support","lift"),shading="confidence")
```

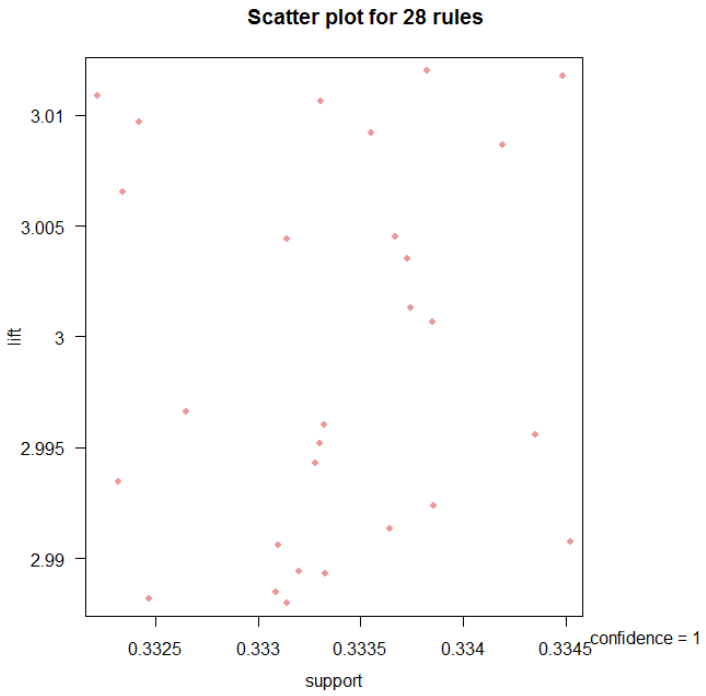


Fig. 4.3. Le graphique des règles d'association générées

```
R> plot(rules, shading="order",control=list(main ="Two-key plot"))
```

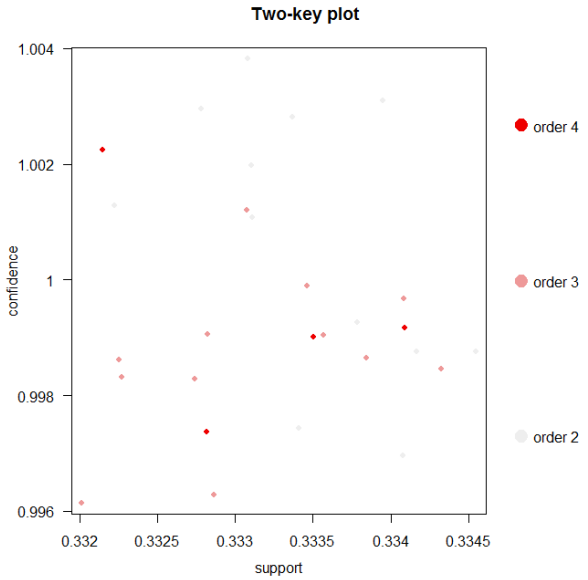


Fig. 4.4. Le graphique des règles d'association générées par filtre.

Même avec ces filtres, le nombre de règles peut vite atteindre plusieurs millions pour seulement quelques centaines d'items et quelques milliers d'observations. Certains logi-

ciels permettent d'ajouter un filtre sur le contenu des règles, pour ne conserver que celles qui contiennent un item donné dans leur résultat ou leurs conditions.

```
R> itemFrequencyPlot(transactions_data,support = 0.1,cex.names=0.8)
```

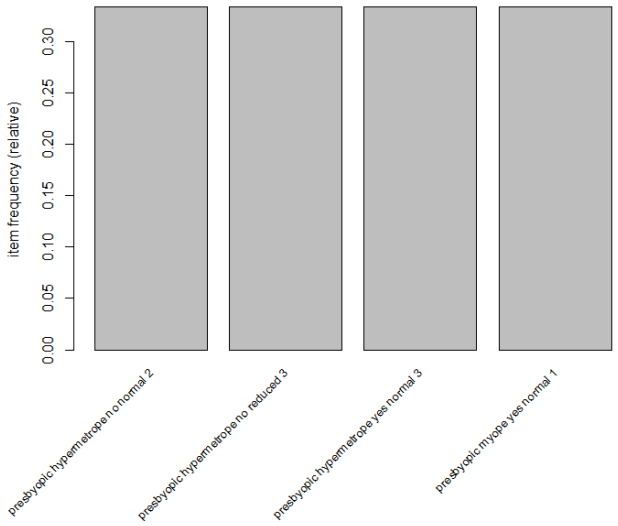


Fig. 4.5. L'histogramme des items les plus fréquents.

4.3 Régression linéaire simple et multiple

Dans la plupart des situations, nous sommes amenés à étudier la relation entre une variable d'intérêt Y (souvent quantitative) et une ou plusieurs variable(s) $X_1, X_2, ..., X_k$, avec pour objectif d'expliquer les variations de la variable d'intérêt. La variable Y est appelée variable « à expliquer » (ou parfois variable dépendante), et les variables $X_1, X_2, ..., X_k$ sont dites « explicatives » et représentent, en épidémiologie, les facteurs de risque ou de confusion. L'utilisation des méthodes d'analyse multivariée, et plus particulièrement des modèles de régression linéaire, permet donc :

- de prendre en compte simultanément plusieurs facteurs pouvant expliquer la variation ou la distribution de la variable Y ;
- d'étudier le rôle de modification d'effet ou de confusion d'un ou de plusieurs facteur(s) ;
- de prédire les valeurs ou la distribution de la variable à expliquer connaissant les valeurs des variables explicatives.

Pour introduire les concepts clés de la régression linéaire, nous allons commencer par présenter dans un premier temps, le modèle de régression linéaire simple en considérant une variable à expliquer quantitative Y et une seule variable explicative X quantitative, même si la variable X peut en principe être aussi qualitative. Dans un second temps, nous présenterons le modèle de régression linéaire multiple permettant d'étudier la relation entre une variable dépendante quantitative Y et plusieurs variables explicatives $X_1, X_2, ..., X_k$ qui peuvent être quantitatives ou qualitatives.

4.3.1 Préparation de la base de données

Dans ce TP, nous allons reprendre l'exemple présenté dans le livre de Lafaye de Micheaux et al. [14]. De ce fait nous allons travailler sur le jeu de données « *Poids-naissance* » du dépôt biostatistique de springer[4]. Ces données expliquent la variabilité du poids de naissance de l'enfant en fonction des caractéristiques de la mère, de ses antécédents et de son comportement pendant la grossesse. La variable à expliquer est le poids de naissance de l'enfant (variable quantitative BWT, exprimée en grammes) et les facteurs étudiés (variables explicatives) sont décrits dans l'annexe E.

```
R> base=read.table("C:\\...\\Documents\\Poids_naissance.csv",
 sep="\t", header= TRUE)
```

Le poids de la mère étant exprimé en livres, nous commençons par effectuer une transformation du data.frame des données pour recoder cette variable en kilogrammes ($1livre = 0.45359237kg$).

```
R> base<- transform(base,LWT=LWT*0.4535923)
R> attach(base) # Accès au nom des variables.
```

4.3.2 Régression linéaire simple

Nous cherchons « expliquer » les variations d'une variable quantitative Y (par exemple, le poids de naissance de l'enfant, noté BWT) par une variable explicative Xégalement quantitative (par exemple, le poids noté LWT).

Il s'agira d'étudier le modèle suivant :

$$BWTi = B0 + B1LWTi + C$$

avec $i = 1, ...n$

Sous le langage R la fonction **lm()** pour linear models permet de réaliser cette opération. Le paramètre principal de cette fonction est une formule, symbolisée par un tilde $_{\sim}$ qui permet de préciser le sens de la relation entre BWT et LWT.

```
R> model1 <- lm(BWT~LWT,data=base)
R> model1
```

La sortie du model1 fournit les estimations par moindres carrés de $B0$ et de $B1$. Nous pouvons maintenant représenter la droite de régression sur le nuage de points au moyen de la fonction **abline()** :

```
R> plot (BWT~LWT, xlab="Poids de la mère",ylab="Poids de l'enfant")
R> abline(model1,col="blue")
```

4. http ://www.biostatisticien.eu/springeR/jeuxDonnees2.html

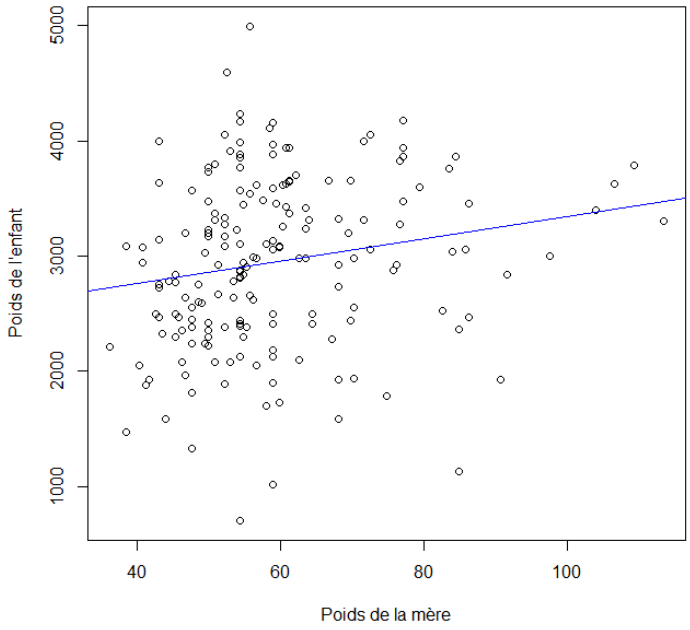


Fig. 4.6. Représentation de la droite de régression des moindres carrés sur le nuage de points du poids de l'enfant (en grammes) versus le poids de la mère (en kilogrammes).

Considérons une nouvelle observation Xo de la variable X pour laquelle nous n'avons pas observé la valeur Yo correspondante de la variable à expliquer Y. Nous cherchons à définir l'intervalle de confiance et de prédiction pour cette nouvelle valeur.

La fonction permettant de définir l'intervalle de prévision et l'intervalle de confiance pour une nouvelle valeur Xo est **predict()**.

Si nous souhaitons calculer la prédiction pour le poids d'un bébé dont la mère a un poids de lwt = 56 kg. Il suffira de :

```
R> lwt0<- 56
R> predict(model1,data.frame(LWT=lwt0),interval="prediction")
```

Pour l'intervalle de confiance de la valeur moyenne du poids des bébés pour un poids de la mère égale à 56 kg.

```
R> predict(model1,data.frame(LWT=lwt0),interval="confidence")
```

Représentons maintenant l'intervalle de confiance et l'intervalle de prévision pour une série de nouvelles valeurs de poids de la mère :

```
R> x <- seq(min(LWT),max(BWT),length=50)
R> intpred <- predict(model1,data.frame(LWT=x),interval= "prediction")
R> intconf <- predict(model1,data.frame(LWT=x),interval= "confidence")
R> plot(BWT~LWT,xlab="Poids de la mère",ylab="Poids de l ' enfant")
R> abline(model1)
R> matlines(x, cbind(intconf, intpred), lty=c(2,2,3,3),col=c ("red", "red",
 "blue", "blue"),lwd=c (2, 2,1,1))
R> legend("bottomright", lty=c (2, 3), lwd=c (2,1),c ("confiance" ,
"prévision") , col=c ("red" , "blue"))
```

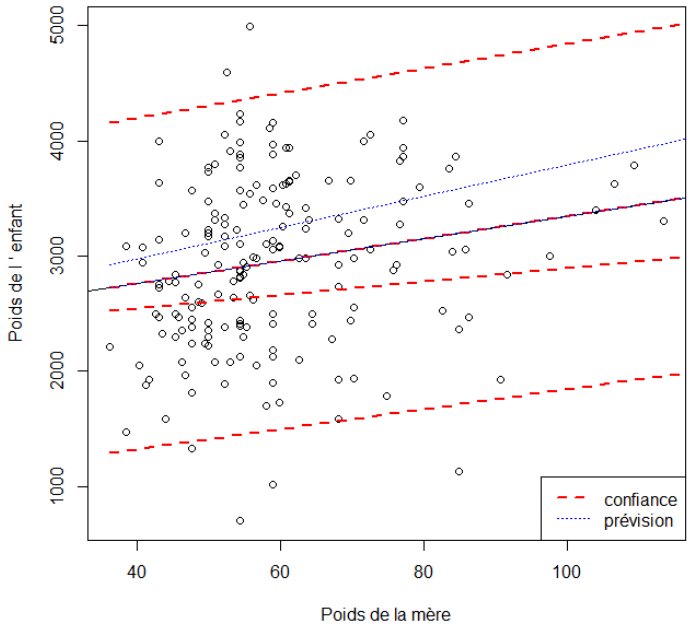


Fig. 4.7. Visualisation de l'intervalle de confiance et de l'intervalle de prévision.

4.3.3 La régression linéaire multiple

Il s'agit d'étudier les variations d'une variable quantitative Y (variable dite à expliquer ou dépendante, supposée aléatoire) en fonction de $p (p > 1)$ variables explicatives $X_1, X_2, ..., X_p$ (variables aussi dites indépendantes). Les variables explicatives peuvent être uniquement quantitatives, uniquement qualitatives (cas appelé ANOVA), ou un mélange de variables quantitatives et qualitatives. Dans ce dernier cas, le modèle de régression linéaire multiple est aussi appelé ANCOVA.

Le modèle de régression linéaire multiple à étudier est de la forme :

$$Y = B0 + B1X_1 + ...BpXp + C$$

Pour ce faire, nous allons étudier la régression du poids de naissance de l'enfant en fonction de l'âge de la mère, de son poids et de son statut tabagique durant la grossesse. Nous noterons X_1 l'âge de la mère (variable AGE), X_2 le poids de la mère (variable LWT), X_3 le statut tabagique de la mère (variable SMOKE) , et Y le poids de naissance de l'enfant (variable à expliquer BWT).

```
R > model2 <- lm(BWT~ AGE + LWT+ as.factor(SMOKE))
R> model2
```

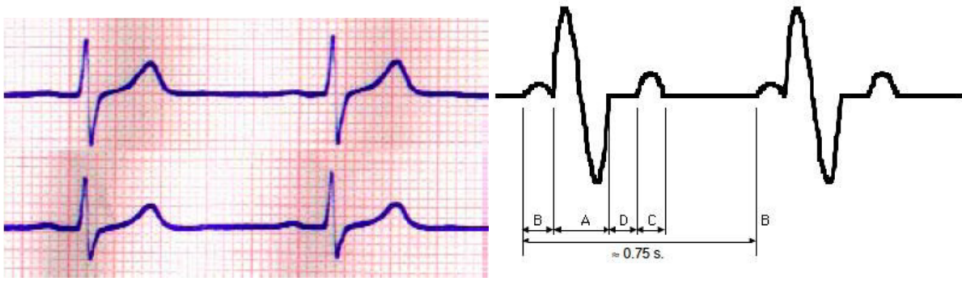
Supposons que nous voulions prédire le poids de naissance d'un enfant dont la mère est âgée de 23 ans, pèse 57 kg et fume. La fonction **predict()** permet d'obtenir une prédiction et son intervalle de prévision ainsi que l'intervalle de confiance du poids moyen des enfants dont les mères ont les caractéristiques décrites précédemment.

```
R> newdata <- data.frame(AGE=23,LWT=57,SMOKE=1)
R> predict(model2,newdata,interval="pred")
R> predict(model2,newdata,interval="conf")
```

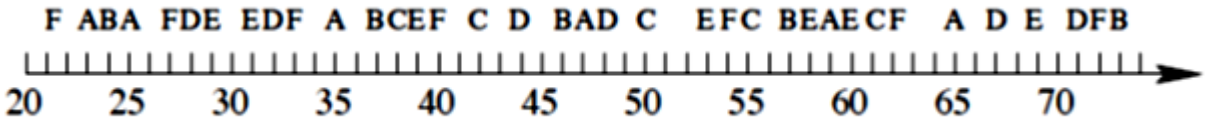
4.4 Travail demandé

4.4.1 Exercice d'application des règles d'association

Un médecin dispose d'un enregistrement de l'électrocardiogramme d'un de ses patients (figure ci-dessous, à gauche). Ce médecin souhaite appliquer Apriori pour détecter des motifs fréquents, et en extraire des règles. Dans un premier, il a utilisé un algorithme de classification pour décrire l'électrocardiogramme à l'aide d'une simple séquence de symboles (figure ci-dessous, droite).



Ainsi, il obtient la séquence ci-dessous, formée des symboles A,B,C,D,E,F sur l'axe du temps.



On ne considérera que les intervalles de temps de 5 secondes. Dans la séquence, il y a donc 50 intervalles ([20,25],[21,26],[22,27],etc...). Il est possible de représenter cette séquence sous la forme d'un tableau de transactions, à raison d'une transaction par intervalle.

1. Ecrivez les premières lignes de ce tableau.
2. Générez les règles d'association avec Apriori.
3. Ajouter un filtre sur le contenu des règles, pour ne conserver que celles qui contiennent un item donné dans leur résultat ou leurs conditions.
4. Effectué un classement de règles suivant l'indice lift ?
5. Que concluez-vous ?

4.4.2 Exercice d'application de la régression linéaire simple et multiple

Nous disposons de données d'étude de l'athérosclérose, c'est un épaississement et une perte d'élasticité des parois internes des artères, dont une des conséquences est l'infarctus du myocarde. La paroi artérielle est constituée de trois couches qui sont respectivement à partir de la lumière artérielle : l'intima, la média et l'adventice. L'épaisseur de l'intima-média est un marqueur reconnu d'athérosclérose. Elle a été mesurée par échographie sur un échantillon de 110 sujets en 1999 dans les CHU de Bordeaux. Des informations sur les principaux facteurs de risques ont aussi été recueillies comme suit :

SEXE	sexe	1=Homme ; 2=Femme
AGE	L'âge du patient	Années
taille	taille	En cm
Poids	poids	En kg
Tabac	Statut tabagique	0= ne fume pas ; 1= A arrêté de fumer ; 2= fume
paqan	Estimation de consommation pour les fumeurs et ex-fumeurs Nombre de paquets /année	
SPORT	Activité physique	0=non ; 1= oui
mesure	Mesure de l'Intima-Média	En millimètre
alcool	Consommation d'alcool	0= ne boit pas ; 1= boit occasionnellement ; 2= boit régulièrement

Régression linéaire simple

1. Tracez le nuage de points de la variable mesure en fonction de la variable AGE. Décrivez-le.
2. Nous cherchons maintenant à ajuster une droite de régression sur ce nuage de points :
   a) proposez un modèle de régression et estimez les paramètres du modèle ;
   b) tracez la droite obtenue sur le nuage de points.
3. Donnez un intervalle de prévision de la mesure de l'intima-média pour une personne de 33 ans.
4. Donnez l'intervalle de confiance de la mesure de l'intima-média moyen pour une personne de 33 ans.
5. Proposez un modèle permettant d'améliorer la prédiction de la mesure de l'intima-média avec pour seule variable explicative AGE.

Régression linéaire multiple

Nous nous intéressons maintenant à ajuster un modèle de régression sur l'ensemble des variables pouvant expliquer les variations de la mesure de l'épaisseur de l'intima-média. L'étude portera sur les variables suivantes : AGE, SPORT, alcool, paqan et la variable IMC que vous devez créer à partir des variables taille, poids. Nous nous focaliserons plus particulièrement à la variable tabac au travers de la variable paqan comme facteur d'exposition principal.

1. Présentez un diagramme de dispersion de toutes les paires de ces variables.
2. Quelles sont les variables explicatives qui vous semblent être les plus explicatives
3. Présentez les résultats du modèle de régression linéaire multiple avec toutes ces variables explicatives.
4. Présentez le modèle final.

Conclusion

L'expérience nous a montré qu'à travers ces travaux pratiques, ce polycopié permet à l'étudiant de découvrir les méthodes de base de fouille de données qui varient principalement en fonction, d'une part, de la nature et de la quantité des données considérées et d'autre part, des questions auxquelles nous cherchons à répondre.

Durant ces séances de travaux pratiques, l'étudiant s'apercevoit que les méthodes de fouille de données ont été empruntées à l'apprentissage automatique, en particulier les méthodes de régression statistique, les méthodes de clustering comme l'algorithme des C-moyennes, ou de classification supervisée, comme les arbres de décision.

La fouille de données représente aujourd'hui un enjeu majeur pour la recherche. Effectivement, l'étudiant est amené de constater par lui même qu'à partir des différentes applications, fouiller des données ne servira à rien si les résultats de cette fouille ne sont pas interprétables. La fouille de données n'est donc qu'une étape d'un processus d'extraction de connaissances plus large qui s'étend de la préparation des données jusqu'à l'interprétation des résultats. En effet, l'étudiant réalise au fur et à mesure des séances de travaux pratiques que la fouille de données établit par ce biais, des connexions vers d'autres disciplines préexistantes telles que les bases de données, les outils de visualisation, les modèles de représentation de la connaissance et plus largement l'intelligence artificielle.

Toutefois, la fouille de données est plus qu'un assemblage hétéroclite de méthodes préexistantes, la fouille de données en tant que domaine de recherche identifié, s'est clairement affirmée quand furent proposées les méthodes de recherche des motifs fréquents et d'extraction des règles d'association. Ces dernières appliquées par l'étudiant dans la dernière partie de ce polycopié, proposent des algorithmes complets pour extraire un grand nombre de règles significatives tout en traitant de grandes quantités de données.

Enfin, l'étudiant découvre à travers ces travaux pratiques le logiciel R. Un langage de programmation simple et efficace, avec un système de documentation intégré très bien conçu, des capacités graphiques évoluées, une aide et une communauté en ligne très active, une flexibilité et des possibilités d'interactions nettement plus importantes qui font de R un outils particulièrement performant et intéressant pour la manipulation de données et l'extraction de connaissances.

ANNEXE A : Résumé des principales fonctions R

Opérateurs usuels

Arithmétique	**Comparaison**	**Logique**
+ addition	< inférieur	!x NON logique
- soustraction	> supérieur	x & y ET logique
multiplication	<= inférieur ou égal	x && y idem
/ division	>= supérieur ou égal	x | y OU logique
^puissance	== égal	x || y idem
%% modulo	!= différent	xor(x,y) OU exclusif
%/% division entière		

Création de données

read.table	lit un data frame à partir d'un fichier. Arguments : $header = TRUE$ si la première ligne correspond aux intitulés des variables ; $sep =$ "," pour indiquer le séparateur de variables dans le fichier ;
skip=n	pour ne pas lire les n premières lignes.
write.table	sauvegarde un data frame dans un fichier.
c	concatène des scalaires en un vecteur.
rbind, cbind	concatène en ligne ou en colonne des vecteurs en une matrice.
list	crée une liste.
matrix	crée une matrice à $nrow$ lignes et $ncol$ colonnes.
data.frame	crée un data frame.
array	crée un tableau dont l'argument dim permet de préciser le nombre de dimensions ainsi que la taille de chaque dimension.
seq	créer une séquence d'entiers.
rnorm, runif	simule la génération d'une variable aléatoire normale, uniforme.

Manipulation de données

x[n]	n-ème élément du vecteur x.
x[n :m]	n-ème au m-ème éléments du vecteur x.
x[c(k,l,m)]	k-ème, l-ème et m-ème éléments du vecteur x.
x[x>m & x<n]	éléments de x compris entre m et n.
l$x ou l [["x"]]	élément x de la liste l.
M[i,j]	élément ligne i et colonne j de la matrice M.
M[i,]	i-ème ligne de la matrice M.
t(M)	transposée de la matrice M.
solve(M)	inverse de la matrice M.
M%*%N	produit des matrices M et N.
sort(x)	tri du vecteur x.

Information sur les variables

length	longueur d'un vecteur.
ncol, nrow	nombre de colonnes et de lignes d'une matrice.
str	affiche le type d'un objet.
as.numeric, as.character	change un objet en un nombre ou une chaine de caractères.
is.na	teste si la variable est de type 'NA' (valeur manquante).

Statistiques

sum	somme d'un vecteur.
mean	moyenne d'un vecteur.
sd, var	écart-type et variance d'un vecteur (dénominateur $n-1$).
rowSums, rowMeans, colSums ou colSums	somme et moyenne en ligne ou en colonne d'une matrice.
max, min	maximum et minimum d'un vecteur.
quantile (x,0.1)	quantile d'ordre 10% du vecteur x.

Graphiques

plot(x)	représente une série de points (ordonnée x et numéro d'indice en abscisse).
plot(x,y)	représente un nuage de points d'abscisse x et d'ordonnée y.
image(x,y,z)	représente en niveau de couleur une image où z représente l'intensité au point x, y (z est une matrice dont le nombre de ligne est la longueur de x et le nombre de colonne celle de y).
lines, points	ajoute une ligne ou des points sur un graphique existant.
hist	histogramme.

barplot	graphique en barre.
qqnorm(x)	quantiles de x en fonction des valeurs attendues selon une loi normale.
qqplot(x,y)	quantiles de y en fonction de ceux de x.
abline	représente une ligne en précisant la pente b et l'ordonnée à l'origine a. Une ligne verticale d'abscisse $x(v = x)$ ou horizontale d'ordonnée $y(v = y)$.
legend	ajoute une légende en précisant les symboles (lty ou pch et col), le texte (text) et l'emplacement (x="topright").
axis	ajoute un axe. Argument : side (1 : bas, 2 : gauche, 3 : haut, 4 : droite).
grid	ajoute un quadrillage.
par(mfrow=c(n,p))	partage la fenêtre graphique en $n * p$ sous graphiques.

Paramètres graphiques

type	"l" pour ligne et "p" pour points.
col	"black", "red", "blue", "red" ... (ou 1, 2, 3, 4...).
lty	type de lignes (1 : solide, 2 : pointillée...).
pch	type de points (1 : cercle, 2 : triangle...).
main	titre principale.
xlab, ylab	titre des axes.
log	échelle logarithmique ("x" pour l'axe des abscisses, "y" pour l'axe des ordonnées, "xy" pour les deux axes).

ANNEXE B : Le jeu de données «satisfaction_hopital»

Présentation

C'est une enquête de satisfactions dans un hôpital, récupéré lors d'un cours de FUN (France Unité Numérique) [1]. Il s'agit d'une étude évaluant la qualité de relation et la quantité d'information reçue par le patient lors de son séjour à l'hôpital. 534 patients ont été recrutés sur plusieurs hôpitaux de la région parisienne.

Description du jeu de données

Variables	**description**	**Code/Unité**
service	service ayant accueilli le patient	code de 1 à 8
sexe	sexe du patient	0 homme, 1 femme
age	âge	en années
profession	profession exercé par les patients	1 : agriculteur exploitant 2 : artisan, commerçant, chef d'entreprise 3 : cadre, profession intellectuelle ou artistique, profession libérale 4 : profession intermédiaire de l'enseignement, de la santé, du travail social ou de la fonction publique, 6 : ouvrier 7 : étudiant, militaire, chômeur sans avoir jamais travaillé 8 : autre
amelioration.sante	impression d'amélioration de la santé du fait du séjour à l'hôpital	0 : aggravée, à 3 : nettement améliorée
amelioration.moral	impression d'amélioration du moral du fait du séjour à l'hôpital	0 : aggravé, à 3 : nettement amélioré
recommander	recommander le service à son entourage	0 : non, 1 : oui, probablement, 2 : oui, sûrement.
score.information	score relatif à la qualité de l'information reçue pendant le séjour	score variant de 10 à 40
score.relation	score relatif à la qualité des relations avec le personnel soignant pendant le séjour	score variant de 10 à 40

1. https ://www.fun-mooc.fr/

ANNEXE C : Le jeu de données « Heart disease »

Présentation

Le jeu de données « Heart disease » regroupe des informations sur un échantillon rétrospectif d'homme vivant dans une région à risque élevé de maladies cardiaques du Cap-Occidental, en Afrique du Sud. Il existe environ deux contrôles par cas de maladie coronarienne CHD. Beaucoup d'hommes atteints de CHD ont subi un traitement de réduction de la pression artérielle et d'autres programmes pour réduire leurs facteurs de risque après leur événement de maladie coronarienne. Dans certains cas, les mesures ont été effectuées après ces traitements. Ces données sont extraites d'un plus grand Ensemble de données, décrit par Rousseauw et al, 1983, South African Medical Journal.

Description du jeu de données

Variables	**description**
sbp	pression sanguine systolique (systolic blood pressure)
tobacco	la quantité de tabac consommé cumulée (cumulative tobacco (kg))
ldl	taux de cholestérol dans le sang (low density lipoprotein cholesterol)
adiposity	adiposité
famhist	antécédents familiaux : Present s'il y a eu des antécédants familiaux (family history of heart disease (Present, Absent))
typea	comportement de type A(type-A behavior)
obesity	obésité
alcohol	consommation courante d'alcool (current alcohol consumption)
age	age au moment de l'attaque cardiaque (age at onset)
chd	variable réponse codée 1 si la maladie du coeur est présente, 0 sinon (response, coronary heart disease)

ANNEXE D : Le jeu de données « Lenses »

Présentation

Un jeu de données réalisé par Benoit Julien pour le dépôt d'UCI[2]. Traite du type de lentilles de contact suivant les paramètres âge, leurs conditions ophtalmologiques.

ID	age	prescription	astigmatic	tear_rate	Class
1	young	myope	no	reduced	3
2	young	myope	no	normal	2
3	young	myope	yes	reduced	3
4	young	myope	yes	normal	1
5	young	hypermetrope	no	reduced	3
6	young	hypermetrope	no	normal	2
7	young	hypermetrope	yes	reduced	3
8	young	hypermetrope	yes	normal	1
9	pre-presbyopic	myope	no	reduced	3
10	pre-presbyopic	myope	no	normal	2
11	pre-presbyopic	myope	yes	reduced	3
12	pre-presbyopic	myope	yes	normal	1
13	pre-presbyopic	hypermetrope	no	reduced	3
14	pre-presbyopic	hypermetrope	no	normal	2
15	pre-presbyopic	hypermetrope	yes	reduced	3
16	pre-presbyopic	hypermetrope	yes	normal	3
17	presbyopic	myope	no	reduced	3
18	presbyopic	myope	no	normal	3
19	presbyopic	myope	yes	reduced	3
20	presbyopic	myope	yes	normal	1
21	presbyopic	hypermetrope	no	reduced	3
22	presbyopic	hypermetrope	no	normal	2
23	presbyopic	hypermetrope	yes	reduced	3

2. http ://archive.ics.uci.edu/ml/datasets/Lenses

Description du jeu de données

Variables	**description**
Class	1 : le patient doit être équipé de lentilles de contact dures, 2 : le patient doit être équipé de lentilles de contact doux, 3 : le patient ne doit pas être équipé de lentilles de contact.

ANNEXE E : Le jeu de données « Poids de naissance »

Présentation

Il s'agit d'une enquête concernant les facteurs de risque associés au faible poids de naissance de nourrissons (données collectées au centre médical de Baystate dans le Massachusetts pendant l'année 1986 [3]). Le faible poids de naissance est un événement qui intéresse les médecins depuis plusieurs années en raison du taux de mortalité infantile et du taux d'anomalies infantiles très élevés chez les nourrissons de faible poids. Le comportement d'une femme pendant la grossesse (régime alimentaire, habitudes tabagiques ...) peut altérer de façon importante les chances de mener la grossesse à terme, et par conséquent, de donner naissance à un enfant de poids normal. Le fichier de données contient les informations sur 189 femmes (Numéro d'identification : $verb\#ID\#$) venant consulter dans le centre médical. On considère qu'un enfant a un faible poids de naissance (prématuré) si celui-ci est inférieur à 2500 g.

Description du jeu de données

Variables	**description**	**Code/Unité**
AGE	L'âge de la mère	Années
LWT	Poids de la mère lors du dernier cycle menstruel	En livres
RACE	Race de la mère	0=blanche 1= noire 3= autre
SMOKE	Tabagisme durant la grossesse	0=non 1= oui
PTL	Nombre d'antécédents de prématurité	0= aucun 1= un etc
HT	Antécédents d'hypertension	0=non 1= oui
UI	Présence d'irritabilité urinaire	0=non 1= oui
FVT	Nombre de visites à un médecin durant le premier trimestre de la grossesse.	0= aucune 1= une etc
BWT	Poids de naissance	En grammes
LOW	Poids de naissance inférieur ou égale à 2500g	0=non 1= oui

3. http ://www.biostatisticien.eu/springeR/Poids$_n aissance.txt$

Références Bibliographiques

[1] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). Advances in knowledge discovery and data mining. chapter Fast Discovery of Association Rules. (pp. 307–328). Menlo Park, CA, USA : American Association for Artificial Intelligence. URL : http://dl.acm.org/citation.cfm?id=257938.257975.

[2] Becker, R., & Chambers, J. (1984). *S : An Interactive Environment for Data Analysis and Graphics*. Taylor & Francis. URL : https://books.google.dz/books?id=_TBadcl04xAC.

[3] Becker, R. A., Chambers, J. M., & Wilks, A. R. (1988). *The New S Language : A Programming Environment for Data Analysis and Graphics*. Monterey, CA, USA : Wadsworth and Brooks/Cole Advanced Books & Software.

[4] Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, *24*, 123–140. URL : http://dx.doi.org/10.1023/A:1018054314350. doi :10.1023/A:1018054314350.

[5] Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

[6] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification And Regression Trees*. New York : Chapman and Hall.

[7] Efron, B. (1979). Bootstrap methods : Another look at the jackknife. *Ann. Statist.*, *7*, 1–26. URL : http://dx.doi.org/10.1214/aos/1176344552. doi :10.1214/aos/1176344552.

[8] Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. New York : Chapman & Hall.

[9] Goulet, V. (2014). Introduction a la programmation en r. Page consultée le 28 Aout 2017 [online]. URL : http://cran.r-project.org/doc/contrib/Goulet_introduction_programmation_R.pdf.

[10] Hahsler, M., Gruen, B., & Hornik, K. (2005). arules – A computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, *14*, 1–25. URL : http://dx.doi.org/10.18637/jss.v014.i15.

[11] Ihaka, R., & Gentleman, R. (1996). R : A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, *5*, pp. 299–314. URL : http://www.jstor.org/stable/1390807.

[12] Izenman, A. J. (2008). Recursive partitioning and tree-based methods. In *Modern Multivariate Statistical Techniques : Regression, Classification, and Manifold Learning* (pp. 281–314). New York, NY : Springer New York. URL : https://doi.org/10.1007/978-0-387-78189-1_9. doi :10.1007/978-0-387-78189-1_9.

[13] MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. L. Cam, & J. Neyman (Eds.), *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297). University of California Press volume 1.

[14] de Micheaux, P., Drouilhet, R., & Liquet, B. (2011). *Le logiciel R : Maîtriser le langage - Effectuer des analyses statistiques*. Statistique et probabilités appliquées. Springer. URL : **https://books.google.dz/books?id=59KNFF_8aMIC**.

[15] Paradis, E. (2002). R pour les débutants. Page consultée le 09 septembre 2017 [online]. URL : **http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf**.

[16] RDevelopmentCoreteam (29 février 2000). R- projet. Page consultée le 18 février 2017 [online]. URL : **https://www.r-project.org**.