République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة 8 ماي 1945 قالمة
Université 8 Mai 1945 Guelma
Faculté des Sciences de la Nature et de la Vie, Sciences de la Terre et de l'Univers

Domaine : Sciences de la Nature et de la Vie
Filière : Ecologie et Environnement
Département : Ecologie et Génie de l'Environnement

Polycopié de cours donné au Master1
Spécialité/ Option : Biodiversité et Environnement

Matière

ECOLOGIE NUMERIQUE

Présenté par : Dr BOUCHELAGHEM EL Hadi

Année universitaire : 2018/2019

TABLE DES MATIERES

GLOSSAIRE

windows. (littéralement « Fenêtres » en anglais) est au départ une interface graphique unifiée produite par Microsoft, qui est devenue ensuite une gamme de systèmes d'exploitation à part entière, principalement destinés aux ordinateurs compatibles PC.

linux. est un système d'exploitation, tout comme Windows ou MacOS X. Il permet de travailler comme on le ferait sous Windows. Mais il fonctionne différemment.

macOS X. est un système d'exploitation partiellement propriétaire développé et commercialisé par Apple depuis 1998, fonctionnant sur tous les ordinateurs Mac compatibles, dont la version la plus récente est macOS Mojave (version 10.14) lancée le 24 septembre 2018.

Tinn-R. éditeur gratuit est une alternative à l'interface graphique installée par défaut avec R sous Windows, et ne fonctionne que sur ce système. Il s'agit d'un éditeur de texte, mais qui propose la coloration syntaxique des scripts R, la soumission de commandes directement depuis l'éditeur, et une aide en ligne efficace. Son interface n'est cependant disponible qu'en anglais. Il s'agit d'un logiciel libre et gratuit. Son installation est vivement recommandée en cas d'utilisation régulière de R sous Windows.
Le site officiel se trouve à l'adresse : http://www.sciviews.org/Tinn-R/.
Le téléchargement peut s'effectuer directement depuis *Source forge* :

- http://sourceforge.net/project/platformdownload.php?group_id=144024

en sélectionnant le fichier nommé *Tinn-R_2.1.1.6_setup.exe* (ou quelque chose y ressemblant). Une fois le fichier téléchargé il suffit de l'exécuter et de poursuivre l'installation en laissant les options par défaut. Une fois **Tinn-R** installé, vous pouvez le lancer *via* le menu *Démarrer*. Plusieurs étapes de configuration restent à effectuer avant de pouvoir réellement commencer à travailler.
Tout d'abord, sélectionnez le menu R, puis *Customize*, puis *Rconfigure.r*. Un nouveau fichier apparaît. Ajoutez alors les lignes suivantes à un endroit quelconque

Package Rcmdr. interface graphique avec menus déroulants et zones "script" et "sortie"

ggplot2. est une librairie R de visualisation de données ou ensemble de méthodes de représentation graphique, en deux ou trois dimensions, utilisant ou non de la couleur et des trames. Les moyens informatiques permettent de représenter des ensembles complexes de données, de manière plus simple, didactique et pédagogique. Développée par Hadley Wickham. La librairie est développée selon les principes développés par Leland Wilkinson dans son ouvrage *The Grammar of Graphics*.

ANOVA. analysis of variance (analyse de la variance)

CRAN. Comprehensive R Archive Network

USTHB. University of Science and Technology Houari Boumediene

GNU. General Public Licence

RGui. R Graphical User Interface (interface graphique)

RStudio. Un IDE (environnement de développement intégré) pour R.

ACFAS. Association francophone pour le savoir-Acfas est un organisme à but non lucratif contribuant à l'avancement des sciences au Québec et dans la Francophonie canadienne. Fondée en 1923, sous le nom **d'Association canadienne-française pour l'avancement des sciences (Acfas)**, l'association est renommée en 2001 afin de mieux refléter ses activités et son dynamisme. Elle devient alors l'Association francophone pour le savoir, et l'acronyme Acfas est conservé.

▪ Création d'objets

file. le nom du fichier (entre "" ou une variable de mode caractère), éventuellement avec son chemin d'accès (le symbole \ est interdit et doit être remplacé par /, même sous Windows), ou un accès distant à un fichier de type URL (http://...)

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.

Eol. le caractère imprimé à la fin de chaque ligne ("\n" correspond à un retour chariot)

dec. le caractère utilisé pour les décimales.

row.names. un vecteur contenant les noms des lignes qui peut être un vecteur de mode character, 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 facteur (si FALSE) ou les conserve en caractères (TRUE); as.is peut être un vecteur logique, numérique ou caractère précisant les variables conservées en caractère.

na.strings. indique la valeur des données manquantes (sera converti en NA (not available)).

colClasses. un vecteur de caractères donnant les classes à attribuer aux colonnes.

nrows. le nombre maximum de lignes à lire (les valeurs négatives sont ignorées)

nlines. le nombre de lignes à lire

skip. le nombre de lignes à sauter avant de commencer la lecture des données

blank. lines. skip. si TRUE, ignore les lignes « blanches »

quiet. si FALSE, scan affiche une ligne indiquant quels champs ont été lus

flush. si TRUE, scan va à la ligne suivante une fois que le nombre de colonnes est atteint (permet d'ajouter des commentaires dans le fichier de données).

check.names. si TRUE, vérifie que les noms des variables sont valides pour R

fill. si TRUE et que les lignes n'ont pas tous le même nombre de variables, des « blancs » sont ajoutés

strip.white. (conditionnel à sep) si TRUE, efface les espaces (= blancs) avant et après les variables de mode caractère

blank. lines. skip. si TRUE, ignore les lignes « blanches »

multi. line. si what est une liste, précise si les variables du même individu sont sur une seule ligne dans le fichier (FALSE)

comment.char. un caractère qui définit des commentaires dans le fichier de données, la lecture des données passant à la ligne suivante (pour désactiver cet option, utiliser comment.char = "")

x. le nom de l'objet à écrire

append. si TRUE ajoute les données sans effacer celles éventuellement existantes dans le fichier.

file. le nom du fichier (entre ""), éventuellement avec son chemin d'accès (le symbole n est interdit et doit être remplacé par /, même sous Windows), ou un accès distant à un fichier de type URL (http://…) ; si file="", les données sont entrées au clavier (l'entrée étant terminée par une ligne blanche)

fichier .txt. (Fichier texte, séparateur tabulation), Les fichiers texte délimités (fichiers .txt), dans lesquels les champs de texte sont séparés par des tabulations (caractère ASCII, code 009).

fichier texte CSV. Séparateur point-virgule, dans lesquels les champs de texte sont en général séparés par des virgules. (Comma separated values).

what. indique le(s) mode(s) des données lues (numérique par défaut)

nmax. le nombre de données à lire, ou, si what est une liste, le nombre de lignes lues (par défaut, scan lit jusqu'à la fin du fichier)

n. le nombre de données à lire (par défaut, pas de limite)

qmethod. spécifie, si quote=TRUE, comment sont traités les guillemets doubles " inclus dans les variables de mode caractère : si "escape" (ou "e", le défaut) chaque " est remplacé par \", si "d" chaque " est remplacé par ""

data. un vecteur ou une matrice.

start. le temps de la 1re observation, soit un nombre, ou soit un vecteur de deux entiers (cf. les exemples ci-dessous).

end. le temps de la dernière observation spécifié de la même façon que start.

frequency deltat. nombre d'observations par unité de temps, la fraction de la période d'échantillonnage entre observations successives (ex. 1/12 pour des données mensuelles) ; seulement un de frequency ou deltat doit être précisé.

ts.eps. tolérance pour la comparaison de séries. Les fréquences sont considérées égales si leur différence est inférieure à ts.eps.
class. classe à donner à l'objet ; le défaut est "ts" pour une série simple, et c("mts", "ts") pour une série multiple.

names. un vecteur de mode caractère avec les noms des séries individuelles dans le cas d'une série multiple ; par défaut les noms des colonnes de data, ou Series 1, Series 2, etc.

▪ Fonctions simples

sum(x). somme des éléments de x.

prod(x). produit des éléments de x.

max(x). maximum des éléments de x.

min(x). minimum des éléments de x.

which.max(x). retourne l'indice du maximum des éléments de x.

which.min(x). retourne l'indice du minimum des éléments de x.

range(x). idem que c(min(x), max(x)).

length(x). nombre d'éléments dans x

mean(x). moyenne des éléments de x.

median(x). médiane des éléments de x.

var(x) ou cov(x). variance des éléments de x (calculée sur n-1) ; si x est une matrice ou un tableau de données, la matrice de variance covariance est calculée.

cor(x). matrice de corrélation si x est une matrice ou un tableau de données (1 si x est un vecteur).

var(x, y) ou cov(x, y). covariance entre x et y, ou entre les colonnes de x et de y si ce sont des matrices ou des tableaux de données.

cor(x, y). corrélation linéaire entre x et y, ou matrice de corrélations si ce sont des matrices ou des tableaux de données.

round(x, n). arrondit les éléments de x à n chiffres après la virgule.

rev(x). inverse l'ordre des éléments de x.

sort(x). trie les éléments de x dans l'ordre ascendant ; pour trier dans l'ordre descendant : rev(sort(x)).

rank(x). rangs des éléments de x.

log(x, base). calcule le logarithme à base de x.

scale(x). si x est une matrice, centre et réduit les données ; pour centrer uniquement ajouter l'option center=FALSE, pour réduire uniquement scale=FALSE (par défaut center=TRUE, scale=TRUE)

pmin(x,y,…). un vecteur dont le i^e élément est le minimum entre x[i], y[i]…

pmax(x,y,…). idem pour le maximum.

cumsum(x). un vecteur dont le i^e élément est la somme de x[1] à x[i]

cumprod(x). idem pour le produit.

cummin(x). idem pour le minimum.

cummax(x). idem pour le maximum.

match(x, y). retourne un vecteur de même longueur que x contenant les éléments de x qui sont dans y (NA sinon).

which(x == a). retourne un vecteur des indices de x pour lesquels l'opération de comparaison est vraie (TRUE), dans cet exemple les valeurs de i telles que x[i] == a (l'argument de cette fonction doit être une variable de mode logique).

choose(n, k). calcule les combinaisons de k événements parmi n répétitions = n!/ [(n-k)!k!]

na.omit(x). supprime les observations avec données manquantes (NA) (supprime la ligne correspondante si x est une matrice ou un tableau de données).

na.fail(x). retourne un message d'erreur si x contient au moins un NA.

unique(x). si x est un vecteur ou un tableau de données, retourne un objet similaire mais avec les éléments dupliqués supprimés.

table(x). retourne un tableau des effectifs des différentes valeurs de x (typiquement pour des entiers ou des facteurs).

table(x, y). tableau de contingence de x et y.

subset(x,…). retourne une sélection de x en fonction de critères (…, typiquement des comparaisons : x$V1 < 10) ; si x est un tableau de données, l'option select permet de préciser les variables à sélectionner (ou à éliminer à l'aide du signe moins).

sample(x, size). ré-échantillonne aléatoirement et sans remise size éléments dans le vecteur x, pour ré-échantillonner avec remise on ajoute l'option replace = TRUE.

▪ Fonctions graphiques

plot(x). graphe des valeurs de x (sur l'axe des y) ordonnées sur l'axe des x

plot(x, y). graphe bivarié de x (sur l'axe des x) et y (sur l'axe des y).

sunflowerplot(x, y). idem que plot() mais les points superposés sont dessinés en forme de fleurs dont le nombre de pétales représente le nombre de points.

pie(x). graphe en camembert.

boxplot(x). graphe boites et moustaches.

stripchart(x). graphe des valeurs de x sur une ligne (une alternative à boxplot() pour des petits échantillons).

coplot(x~y j z). graphe bivarié de x et y pour chaque valeur (ou intervalle de valeurs) de z.

interaction.plot(f1, f2, y). si f1 et f2 sont des facteurs, graphe des moyennes de y (sur l'axe des y) en fonction des valeurs de f1 (sur l'axe des x) et de f2 (différentes courbes) ; l'option fun permet de choisir la statistique résumée de y (par défaut fun=mean)

matplot(x,y). graphe bivarié de la 1re colonne de x contre la 1re de y, la 2e de x contre la 2e de y, etc.

dotchart(x). si x est un tableau de données, dessine un graphe de Cleveland (graphes superposés ligne par ligne et colonne par colonne).

fourfoldplot(x). visualise, avec des quarts de cercles, l'association entre deux variables dichotomiques pour différentes populations (x doit être un tableau avec dim=c(2, 2, k) ou une matrice avec dim=c(2, 2) si k = 1).

assocplot(x). graphe de Cohen-Friendly indiquant les déviations de l'hypothèse d'indépendance des lignes et des colonnes dans un tableau de contingence à deux dimensions.

mosaicplot(x). graphe en « mosaïque » des résidus d'une régression log-linéaire sur une table de contingence.

pairs(x). si x est une matrice ou un tableau de données, dessine tous les graphes bivariés entre les colonnes de x.

plot.ts(x). si x est un objet de classe "ts", graphe de x en fonction du temps, x peut être multivarié mais les séries doivent avoir les mêmes fréquences et dates.

ts.plot(x). idem mais si x est multivarié les séries peuvent avoir des dates différentes et doivent avoir la même fréquence.

hist(x). histogramme des fréquences de x.

barplot(x). histogramme des valeurs de x.

qqnorm(x). quantiles de x en fonction des valeurs attendues selon une loi normale.

qqplot(x, y). quantiles de y en fonction des quantiles de x

contour (x, y, z). courbes de niveau (les données sont interpolées pour tracer les courbes), x et y doivent être des vecteurs et z une matrice telle que dim(z)=c(length(x), length(y)) (x et y peuvent être omis).

filled. contour(x, y, z). idem mais les aires entre les contours sont colorées, et une légende des couleurs est également dessinée image(x, y, z) idem mais les données sont représentées avec des couleurs.

persp(x, y, z). idem mais en perspective stars(x) si x est une matrice ou un tableau de données, dessine un graphe en segments ou en étoile où chaque ligne de x est représentée par une étoile et les colonnes par les longueurs des branches.

symbols(x, y,…). dessine aux coordonnées données par x et y des symboles (cercles, carrés, rectangles, étoiles, thermomètres ou « boxplots ») dont les tailles, couleurs, etc, sont spécifiées par des arguments supplémentaires.

termplot(mod.obj). graphe des effets (partiels) d'un modèle de régression (mod.obj).

INTRODUCTION AU LOGICIEL R

«Le logiciel libre n'a pas de prix, mais il a une grande valeur,
comme l'eau, l'air ou les mathématiques.»
Pierre Cohen-Bacrie

Objectifs

1. But pédagogique

Conforme aux programmes officiels du LMD, ce polycopié de cours en Ecologie numérique s'adresse aux étudiants de Master1 de l'enseignement supérieur en Biodiversité et Environnement. Il est conçu de façon à aplanir au mieux les difficultés inhérentes au discours scientifique tout en conservant la rigueur nécessaire. Le langage R est un langage open source de traitement des données et d'analyse statistique. L'objectif de cet ouvrage est d'apprendre le traitement des données avec R à tous ceux qui doivent produire des statistiques descriptives, des graphiques et des exports de tableaux.

Le contenu ne se limite pas à la modélisation statistique, mais il montre tout ce qu'il faut savoir faire avant, autour et après la construction du modèle qu'il s'agisse d'importation et de préparation des données ou de restitution des résultats.

2. Le programme d'Ecologie numérique du S1 se compose de huit grandes parties :

Introduction au logiciel R
1. Présentation de R
2. Installation
3. Package
4. Importation et lecture de données
5. Accès aux variables
6. Fonctions simples
7. Boucles et autres fonctions
8. Les outils graphiques

Au terme de ce cours, l'étudiant sera capable de :

1. Lire les ensembles de données externes et les manipuler dans R.
2. Utilisation de librairies R spécialisées
3. Ecrire le code informatique en matière de R à mettre en œuvre des méthodes statistiques.
4. Ecrire le code informatique en R pour représenter des données avec l'utilisation de graphiques conventionnelles.

INTRODUCTION

Ce polycopié de cours est né comme une introduction au langage de programmation R et avec l'intention d'être un guide pratique pour apprendre les bases de programmation dans ce langage. Ce langage de programmation est dédié aux statistiques et à la manipulation de données.

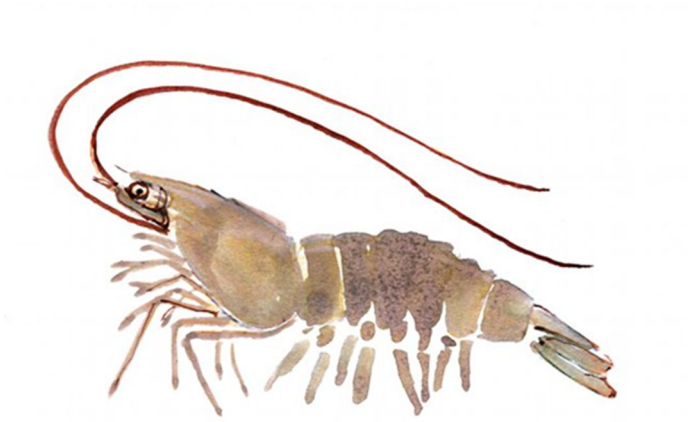
Nous allons commencer avec les structures de base de R, vecteurs, matrices, listes, et dataframe. Nous allons voir comment créer des objets en R et comment utiliser les premières fonctions de R, par exemple pour réorganiser un vecteur ou ajouter des colonnes à une matrice. Nous parlerons de la façon de mettre en place notre environnement de travail de R, comment installer et charger en mémoire un package, comment créer et extraire des sous-ensembles de données, supprimer les données en double. Nous allons voir comment importer des données de R, différents formats, principalement le .csv, mais aussi des données dans Excel, txt.

On va apprendre à manipuler nos données grâce à des fonctions et des packages spécifiques. Pour conclure, on traitera un peu de statistique sur R, et la création de diagrammes, soit avec les fonctions de base, soit avec de packages spécifiques.

Enfin, je me dois de souligner que ce travail est le fruit d'une écoute mutuelle : la mienne, devant les étudiants et chercheurs en Sciences de l'Environnement de l'Université 8 Mai 1945 Guelma se retrouvant dans le besoin d'analyser leurs résultats de recherche à l'aide de tests statistiques, me demandant beaucoup d'efforts pour rendre mon cours utile et compréhensible. J'espère que nombre de lecteurs néophytes en statistique trouveront dans cet ouvrage solutions à leurs problèmes.

CHAPITRE 1

HISTORIQUE ET PRESENTATION DE R



1.1 Historique

Dans le milieu des années1970, une équipe de chercheurs de AT&T Bell Laboratories, composée de John Chambers [2], Douglas Bates, Rick Becker [3], Bill Cleveland, Trevor Hastie, Daryl Pregibon et Allan Wilks, développe un langage de programmation appelé S (la lettre S faisant référence à statistics). Il s'agit d'un langage permettant de manipuler les données et d'effectuer des analyses statistiques et graphiques. Dans le milieu des années 1990, Ross Ihaka et Robert Gentleman [4] créent le R au département de Statistiques de l'Université d'Auckland. Ce langage et logiciel, inspiré du S et de Scheme est distribué sous les termes de la « GNU General Public Licence » (http://www.gnu.org/). Un logiciel sous licence GNU est un « logiciel libre » qui permet à tout à chacun :

- de disposer du logiciel gratuitement (généralement téléchargeable gratuitement) ;
- d'utiliser le programme pour tout usage, y compris commercial ;
- d'étudier le fonctionnement du programme, de l'améliorer, le modifier, publier ces améliorations (l'accès libre au code source est donc requis) ;
- de redistribuer des copies, gratuitement ou non, sous quelque forme que ce soit ; Cela favorise le développement « collaboratif » du logiciel par une communauté importante d'utilisateurs.

Le développement et la distribution de R sont assurés par plusieurs statisticiens regroupés dans le « R Development Core Team », qui rend public ses développements dans le « Comprehensive R Archive Network (CRAN) » (http://www.cran.r-project.org). Un grand nombre de personnes ont également contribué à R en programmant ou en établissant des Bug Reports. R est à la fois un langage et un logiciel ; parmi ses caractéristiques les plus remarquables citant :

- un système performant de stockage et de manipulation de données ;
- la possibilité d'effectuer du calcul matriciel et autres opérations complexes ;
- une large collection intégrée et cohérente d'outils d'analyse statistique ;
- un large éventail d'outils graphiques particulièrement flexibles ;
- un langage de programmation simple et efficace qui inclut de nombreuses facilités.

1.2 L'évolution de l'écologie numérique

L'utilisation de l'outil statistique ne requiert pas de compétences mathématiques élevées [6]. Dans le milieu scientifique, le nom de Pierre Legendre commence à circuler en 1979, année de la publication de l'ouvrage Écologie numérique, dont il est le coauteur avec son frère Louis. La carrière du scientifique, alors âgé d'une petite trentaine d'années, est lancée. Pourtant, à l'origine, l'écologie n'aurait pas été le premier choix du chercheur. La famille de Pierre Legendre est une véritable pépinière de scientifiques, et le grand responsable est Vianney Legendre — le père de Pierre —, un biologiste qui travaille pour le gouvernement du Québec. Il s'intéressera aux poissons des lacs du Québec et publiera des ouvrages sur le sujet. Chargé de cours à l'Université de Montréal, il avait une certaine aisance dans l'enseignement des sciences. Ce passionné souhaite par-dessus tout transmettre son savoir à ses deux fils : « Le soir à la maison, autour de la table, il y avait toujours un sujet qu'il abordait à l'heure du dessert », se souvient Pierre Legendre. Toutefois, ce père à la fois chimiste et biologiste ne poussait pas nécessairement ses fils vers une carrière en biologie, mais plutôt vers les mathématiques et la physique.

Arrive l'épisode de cette grande réunion en France où Pierre Legendre avait été invité à parler de ses recherches : « C'est là-bas que je me suis rendu compte que mon frère [l'océanographe Louis Legendre] s'intéressait au même problème, mais de manière complémentaire à ce que je faisais. On est en 1975, et on décide alors d'écrire un livre ensemble. » Pour l'époque, l'idée de la publication d'un tel volume représente une opération à haut risque puisque de jeunes professeurs d'université ne sont pas les bienvenus pour écrire un tel livre. Un éditeur québécois refuse donc la publication. Les deux frères ne baissent pas les bras et, avec l'audace de leur jeunesse, ils adressent leur manuscrit aux éditions Masson, à Paris, véritable référence en publication d'ouvrages scientifiques… Et c'est en 1979 que sera publiée chez cet éditeur la première édition d'Écologie numérique. Depuis 1980, Pierre Legendre est professeur au Département de sciences biologiques de l'Université de Montréal. Il y poursuit son travail sur l'écologie numérique entouré de toute une équipe de chercheurs. « C'est une science qui se développe rapidement. Peut-être parce que nous avons jeté les bases avec nos manuels dans lesquels, d'ailleurs, on décrit quels sont les trous à boucher. »

Il ne faut pas croire que notre chercheur passe sa vie les yeux rivés sur un écran d'ordinateur. L'homme a toujours voulu continuer à faire du terrain. Le prix de l'Acfas

Adrien- Pouliot qui lui a été remis l'an 2015, souligne sa coopération avec la France. Pierre Legendre travaille entre autres en collaboration avec l'Institut français de recherche pour l'exploitation de la mer (l'IFREMER) : « Je me suis déjà embarqué avec eux pour une mission de deux semaines à bord d'un navire et je me suis retrouvé aux commandes d'un sous-marin téléguidé à deux kilomètres de profondeur ! »

1.3 Présentation de R

Le système R connaît depuis plus d'une décennie une progression remarquable dans ses fonctionnalités, dans la variété de ses domaines d'application ou, plus simplement, dans le nombre de ses utilisateurs. La documentation disponible suit la même tangente : plusieurs maisons d'édition proposent dans leur catalogue des ouvrages — voire des collections complètes — dédiés spécifiquement aux utilisations que l'on fait de R en sciences naturelles, en sciences sociales, en finance, etc [1].

R est un logiciel de traitement de données : il permet de réaliser d'importants calculs statistiques sur une grande quantité de données et de générer des représentations graphiques rigoureuses.

Plus qu'un logiciel, "R" est un langage de programmation objet. Chaque donnée est stockée dans un objet que l'on doit nommer. R est un logiciel de calcul scientifique interactif libre qui possède une large collection d'outils statistiques et graphiques. Plusieurs sites sont consacrés à ce logiciel, en particulier le site http://www.r-project.org/ offre une description exhaustive sur le langage R et fournit les liens indispensables pour les différents téléchargements, accéder aux différentes bibliothèques de fonctions ainsi que les des documents d'aide. Le site miroir du cict peut être utilisé aussi : http://cran.cict.fr/index.html. Des versions compilées de R sont disponibles pour Linux, Windows et Mac OS X.

1.4 Connaissances préalables recommandées

Maitriser les outils informatiques et statistiques nécessaires aux sciences des données, en particuliers, la gestion de projets d'analyse des données, l'importation, remaniement et transformation des données, la visualisation sous forme de graphiques, et l'inférence. Pouvoir présenter clairement et rigoureusement les résultats de ces analyses dans un rapport scientifique de manière reproductible. Etre capable d'analyser correctement des données biologiques usuelles en pratique. Statistiques uni- et bivariées de base, y compris ANOVA, variance, covariance et corrélation.

1.5 L'utilisation de R présente plusieurs avantages

R s'utilise principalement en ligne de commande (au clavier ou par redirection) mais il existe aussi des interfaces avec des menus et des boutons. Nous utiliserons la ligne de commande afin de ne pas dépendre du système d'exploitation. Ecrire **x <- calcul** effectue le calcul et stocke le résultat dans la variable x. Les variables peuvent être des scalaires (une seule valeur), un vecteur (plusieurs valeurs de même type), une matrice (plusieurs vecteurs de même dimension et de même type), une liste (ensemble de diverses variables nommées), un cadre de données ou *data frame* (plusieurs vecteurs de même dimension et mais pouvant être de type différent). Tout ce qui suit le symbole dièse (#) est un commentaire et est ignoré par R, ce qui est bien pratique pour expliquer et se relire. De nombreuses fonctions s'appliquent tout autant à un scalaire qu'à un vecteur qu'à une matrice (via apply) et même à une liste (via lapply). On peut construire des vecteurs avec l'opérateur de séquence : (le symbole deux points) et la fonction collect qui s'écrit c().

- c'est un logiciel multiplateforme, qui fonctionne aussi bien sur des systèmes Linux, Mac OS X ou Windows;
- c'est un logiciel libre[1], développé par ses utilisateurs et modifiable par tout un chacun ;
- c'est un logiciel gratuit ;
- c'est un logiciel très puissant, dont les fonctionnalités de base peuvent être étendues à l'aide d'extensions[2] ;
- c'est un logiciel dont le développement est très actif et dont la communauté d'utilisateurs ne cesse de s'élargir ;
- c'est un logiciel avec d'excellentes capacités graphiques.

1. Pour plus d'informations sur ce qu'est un logiciel libre, voir : http://www.gnu.org/philosophy/free-sw.fr.html
2. Il en existe actuellement plus de 1500, disponibles sur le *Comprehensive R Archive Network* (CRAN) : http://cran. r-project.org/

1.6 Comme rien n'est parfait, on peut également trouver quelques inconvénients

- le logiciel, la documentation de référence et les principales ressources sont en anglais. Il est toutefois parfaitement possible d'utiliser R sans spécialement maîtriser cette langue ;
- par son mode de fonctionnement, R charge normalement l'intégralité des données traitées en mémoire. Il nécessite donc une machine relativement puissante pour travailler sur des grosses enquêtes de plusieurs milliers d'individus ;
- il n'existe pas encore d'interface graphique pour R équivalente à celle d'autres logiciels comme SPSS ou Modalisa. R fonctionne à l'aide de scripts (des petits programmes) édités et exécutés au fur et à mesure de l'analyse, et se rapprocherait davantage de SAS dans son utilisation (mais avec une syntaxe et une philosophie très déférentes).

À noter que ce dernier point, qui peut apparaître comme un gros handicap, s'avère après un temps d'apprentissage être un mode d'utilisation d'une grande souplesse.

1.7 Philosophie de R

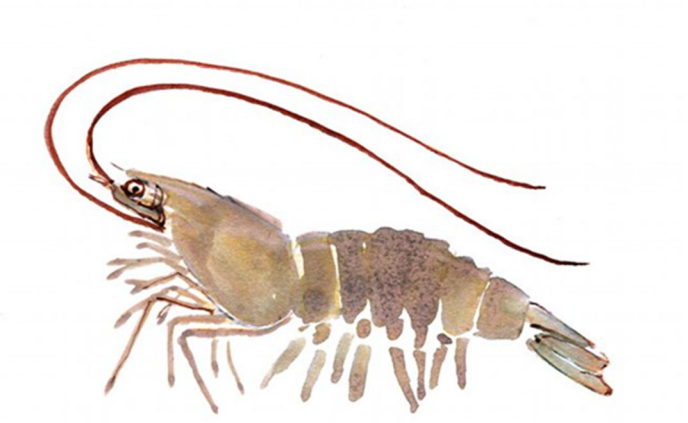
Deux points particuliers dans le fonctionnement de R peuvent parfois dérouter les utilisateurs habitués à d'autres logiciels :

- sous R, en général, on ne voit pas les données sur lesquelles on travaille ; on ne dispose pas en permanence d'une vue des données sous forme de tableau, comme sous Modalisa ou SPSS. Ceci peut être déroutant au début, mais on se rend vite compte qu'on n'a pas besoin de voir en permanence les données pour les analyser ;

- avec les autres logiciels, en général la production d'une analyse génère un grand nombre de résultats de toutes sortes dans lesquels l'utilisateur est censé retrouver et isoler ceux qui l'intéressent. Avec R, c'est l'inverse : par défaut l'affichage est réduit au minimum, et c'est l'utilisateur qui demande à voir des résultats supplémentaires ou plus détaillés. Inhabituel au début, ce fonctionnement permet en fait assez rapidement de gagner du temps dans la conduite des analyses.

CHAPITRE 2

PROCEDURE D'INSTALLATION DU LOGICIEL R



2.1 Procédure d'installation du logiciel R

R est disponible sous différente plateformes : Windows, Mac OS et Linux. Il faudra donc adapter l'installation au type de machine que vous avez. Nous ne couvrons ici que l'installation de R sous Windows. Rappelons qu'en tant que logiciel libre, R est librement et gratuitement installable par quiconque. Plusieurs sites sont consacrés à ce logiciel [8]. La première chose à faire est de télécharger la dernière version du logiciel. Pour cela il suffit de se rendre à l'adresse suivante :

Page web de R : http://www.r-project.org/

Aide à l'installation : http://cran.r-project.org/faqs.html

- cliquez sur CRAN pour rejoindre le site http://cran.r-project.org/mirrors.html
- cliquez sur https://cran.usthb.dz/ dans la rubrique "Algeria" (University of Science and Technology Houari Boumediene).
- Une fois votre miroir choisi, cliquez sur votre système d'exploitation, par exemple (Download R for Windows, R fonctionne pour les versions de Download R for Linux ou Download R for (Mac) OS X)
- cliquez sur "base"

Un miroir est une copie du site situé en un lieu différent. Afin de bénéficier de la meilleure vitesse de téléchargement, il est préférable de sélectionner un miroir situé à proximité de chez vous. Quel que soit le lien choisi, le site et son contenu seront identiques.

Note: CRAN does not have Windows systems and cannot check these binaries for viruses. Use the normal precautions with downloaded executables.
Subdirectories:
base Binaries for base distribution. This is what you want to install R for the first time.
contrib Binaries of contributed packages (managed by Uwe Ligges)

Puis télécharger le fichier rw2001.exe :

README.rw2001 Installation and other instructions.
CHANGES New features of this Windows version.
NEWS New features of all versions.
rw2001.exe Setup program (about 23 megabytes). Please download this from a mirror near you.
This corresponds to the file named SetupR.exe in pre-1.6.0 releases.

Vous allez alors vous voir proposer le téléchargement d'un fichier nommé R-3.5.1-win32.exe (les X étant remplacées par les numéros de la dernière version disponible). Une fois ce fichier sauvegardé sur votre poste, exécutez-le et procédez à l'installation du logiciel : celle-ci s'effectue de manière tout à fait classique, c'est-à-dire en cliquant un certain nombre de fois[1] sur le bouton Suivant.
Une fois l'installation terminée, vous devriez avoir à la fois une magnifique icône R sur votre bureau ainsi qu'une non moins magnifique entrée R dans les programmes de votre menu Démarrer. Il ne vous reste donc plus qu'à lancer le logiciel pour voir à quoi il ressemble.

Enfin signalons l'existence, grâce au travail de Mayeul Kauffmann [9], d'une version portable de R pour Windows permettant notamment d'installer facilement sur une clé USB R, l'éditeur Tinn-R ainsi que plusieurs extensions. On peut donc avoir en permanence sur soi un environnement de travail complet :
http://otan.ecoledelapaix.org/spip.php?article102

- cliquez sur "Download R 3.5.1 for <votre système d'exploitation>", par exemple (for Windows, 62 megabytes, 32/64 bit)
- une fois le fichier d'installation téléchargé, lancez le
- choisissez les options par défaut

Un raccourci est créé sur le bureau Rgui.exe (R Graphical user interface).
Vous pouvez changer le répertoire de travail en modifiant démarrer dans : pour cela créer un répertoire sous la racine par exemple userdata, et taper le chemin complet dans la case démarrer dans : c:\userdata.

Pour vérifier votre dossier de travail, il suffit de double cliquer sur le raccourci rgui.exe, puis à l'invite de R [8] signalant que la console est en attente d'une instruction (désigné par le symbol >) taper getwd() :

1. Voire un nombre de fois certain. Vous pouvez laisser les options par défaut à chaque étape de l'installation.

2.2 Quitter R

La fenêtre "R Console" étant active, sélectionnez "File/Exit" et répondez "Oui" à la question "Save workspace image?"

Et voilà...

Tout votre travail est-il perdu ?

Non. Redémarrez R, et remarquez la ligne :

[Previously saved workspace restored]

Tapez "ls()" et constatez que vos variables sont toujours là.

Le "workspace" ("espace de travail"), c'est à dire l'ensemble des variables, a été sauvegardé sur le disque. Cela permet de reprendre une analyse de données au point où on l'a laissé quand on a quitté R.

Si vous voulez "nettoyer" le workspace, c'est à dire supprimer toutes les variables qu'il contient, tapez la commande "rm(list=ls())".

Il est possible de choisir le nom de fichier où est sauvegardé le workspace (par défaut "RData"). Cela permet de faire plusieurs analyses indépendantes sans les mélanger. (Voir les menus File/Load workspace/ Save Workspace). Une alternative plus recommandée et de créer un dossier pour chaque analyse de données indépendantes.

Si vous quittez R en fermant la fenêtre, une boîte de dialogue s'affichera, cliquer sur oui :

Dans votre répertoire de travail userdata, deux fichiers sont créés.

Le fichier .RData contient les objets créés lors de la session. Les objets de R peuvent être des données (tableaux), des fonctions (formule, expression…), des graphiques…

En double cliquant sur .RData, vous pouvez le consulter dans R. L'historique des instructions lors de la session est enregistré dans le fichier. Rhistory qui peut être consulté à l'aide d'un éditeur de texte.

Tapez la commande history(). Une fenêtre s'affiche listant les dernières commandes que vous avez tapés

La manière la plus efficace de travailler avec R consiste à sauvegarder les commandes au fur et à mesure dans un fichier texte. Pour cela, en parallèle avec R, ouvrez un éditeur de fichier

texte (le plus simple d'entre eux, bien qu'il soit très limité, est le bloc-notes de Windows disponible dans les accessoires) [1].

En utilisant le copier/coller, copier dans le fichier texte les commandes qui font l'essentiel de l'analyse. A la fin de votre session de travail, sauvez ce fichier avec un nom explicite (par exemple le nom de l'expérience) et une extension "R".

Quand vous reprendrez cette analyse quelques jours ou mois plus tard, vous pourrez réutiliser ce fichier, qu'on appelle habituellement un script. R vous permettra de ré-exécuter les commandes de ce script en utilisant la commande source.

Faites un essai : créez un fichier qui contient les lignes suivantes :

```
a=rnorm(100)
b=rnorm(100)
summary(a)
summary(b)
cor.test(a,b)
```

1Pour ceux qui emploient l'éditeur Emacs, il existe un package appelé ESS qui fournit la colorisation syntaxique des commandes R, et plein d'autres fonctions utiles (voir stats.ethz.ch/ESS).

Sauvez-le dans "Mes documents", sous le nom "test.R".

Dans R, utilisez le menu "File/Change Dir" pour aller dans "Mes Documents". Puis tapez :

```
source("test.R", echo=T)
```

Vérifiez que cela marche.

Sous Linux, il n'est pas nécessaire de démarrer R : on peut entrer "R BATCH script.R" sur une ligne de commande dans un terminal et les résultats sont écrits automatiquement dans le fichier "script.Rout".

2.3 Mise à jour de R sous Windows

Pour mettre à jour R sous Windows, il suffit de télécharger et d'installer la dernière version du programme d'installation.
Petite particularité, la nouvelle version sera installée à côté de l'ancienne version. Si vous souhaitez faire de la place sur votre disque dur, vous pouvez désinstaller l'ancienne version en utilisant l'utilitaire Désinstaller un programme de Windows.
Lorsque plusieurs versions de R sont disponibles, RStudio choisit par défaut la plus récente. Il est vous est possible de spécifier à RStudio quelle version de R utiliser via le menu Tools > Global Options > General.

Petit défaut, les extensions (packages) sont installées par défaut sous Windows dans le répertoire Documents de l'utilisateur > R > win-library > x.y avec x.y correspondant au numéro de la version de R. Ainsi, si l'on travaillait avec la version 3.0 et que l'on passe à la version 3.2, les extensions que l'on avait sous l'ancienne version ne sont plus disponibles pour la nouvelle version. Une astuce consiste à recopier le contenu du répertoire 3.0 dans le répertoire 3.2. Puis, on lancera RStudio (s'il était déjà ouvert, on le fermera puis relancera) et on mettra à jour l'ensemble des packages, soit avec la fonction, update.packages soit en cliquant sur Update dans l'onglet Packages du quadrant inférieur droit.

La méthode conseillée pour mettre à jour R sur les plateformes Windows est la suivante[2] :

- Désinstaller R. Pour cela on pourra utiliser l'entrée Uninstall R présente dans le groupe R du menu Démarrer.
- Installer la nouvelle version comme décrit précédemment.
- Se rendre dans le répertoire d'installation de R, en général C:\Program Files\R. Sélectionner le répertoire de l'ancienne installation de R et copier le contenu du dossier nommé library dans le dossier du même nom de la nouvelle installation.
- En clair, si vous mettez à jour de R 3.5.1 vers R 3.5.2, copiez tout le contenu du répertoire C:\Program Files\R\R-3.5.1\library dans C:\Program Files\R\R-3.5.2\library.
- Lancez la nouvelle version de R et exécuter la commande update packages pour mettre à jour les extensions.

2. Méthode conseillée dans l'entrée correspondante de la FAQ de R pour Windows : http://cran.r-project.org/bin/ Windows/rw-FAQ.html#What_0027s-the-best-way-to-upgrade_003f

2.4 Bases du fonctionnement de R

2.4.1 Premières instructions avec R

Lancer R

Sous Windows ou Mac OS :

Double-cliquez sur l'icône du programme. Lorsque vous lancez le programme, il se présente sous la forme d'une fenêtre appelée RGui (GUI pour "Graphical User Interface", interface graphique), qui permet l'accès à quelques menus de base, et d'une fenêtre R Console, dans laquelle l'utilisateur entre des instructions sous forme de texte.

L'invite de commandes

Après un message d'avertissement, le prompt de la forme du signe plus-grand-que > apparaît alors comme vous le voyez, un curseur clignote au niveau du chevron (le signe ">"), c'est là que sera rentré le code, indiquant que R est en attente des commandes, l'invite de commande de R (>) apparaît. Toutes les étapes qui suivent se feront par l'intermédiaire de ce terminal. Commençons donc avec des choses faciles et tapons le chiffre 42 et voyons ce que cela donne : Rappelons que R est un langage interactif, donc toujours en attente d'une instruction signalée par l'invite « < » et sur laquelle devrait se trouver votre curseur. Cette ligne est appelée l'invite de commande (ou prompt en anglais). Elle signifie que R est disponible et en attente de votre prochaine commande.

R est un langage orienté-objet, c'est-à-dire que les variables, les données, les matrices, les fonctions, les résultats, etc. sont stockés dans la mémoire vive de l'ordinateur sous forme d'objets qui ont un nom : ils suffit alors de taper le nom de l'objet pour afficher son contenu.

Help (l'aide en ligne)

R dispose d'une aide en ligne très exhaustive et qui peut vous être très utile car la plupart des fonctions de R nécessitent plusieurs arguments et options, donc ne pas hésiter à faire appel cette aide en ligne !!.

En tapant ? suivi du nom de l'instruction ou help ("nom de l'instruction"). La description de cette instruction, ses arguments, le type d'objet retourné par cette instruction sont affichés ainsi quelques exemples d'utilisation. Sous R, l'aide s'obtient avec la fonction help() qui affiche en ligne des explications.

Voici différentes manières d'ouvrir la page d'aide pour la fonction log()

```
> help("log")
> help(log)
> ?log,
```

avec example() qui montre l'utilisation standard de la fonction. Si vous souhaitez plonger plus en détail dans la documentation R vous pouvez alors appeler les fonctions help.start() et help.search() qui utilisent le navigateur web par défaut pour afficher l'aide. Après un petit temps de chargement elle vous affichera une documentation très complète (en anglais) dans votre navigateur internet par défaut :

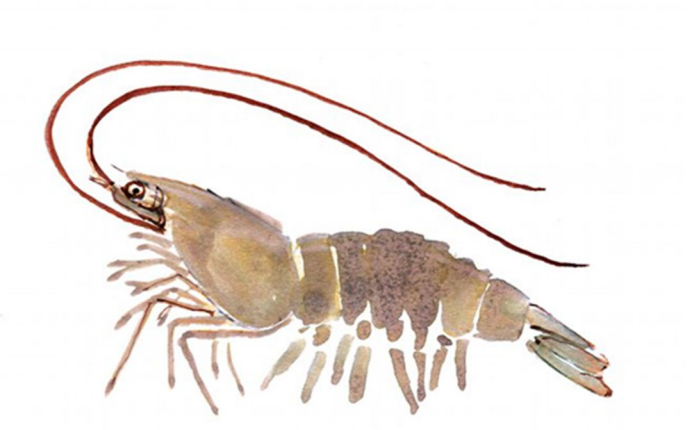
```
> help.start()
> help.search()
```

Cette instruction permet de lancer votre navigateur (par exemple Mozilla ou Internet-explorer) et d'accéder directement à l'aide html

file:///C:/PROGRA~1/R/rw1081/doc/html/rwin.html

CHAPITRE 3

INSTALLATION ET CHARGEMENT D'UN PACKAGE R



3.1 Installation et chargement d'un package R

Il est simple de charger en R des packages supplémentaires (Extensions) à ceux chargés par défaut. Il suffit d'utiliser la commande library comme dans cet exemple :

```
>library(ggplot2)
```

Cette commande fonctionne uniquement si le package a été préalablement installé sur notre ordinateur. L'installation d'un package et le chargement d'un package sont donc deux étapes distinctes. L'installation n'a pas à être faite à chaque fois que le package est chargé. Certains packages R sont installés automatiquement lors de l'installation de R. Ils ne sont pas pour autant tous chargés lors d'une ouverture de session. La fonction installed.packages retourne des informations à propos des packages R installés sur l'ordinateur local.

```
>i <- installed.packages()
>head(i[, "Package"])
# Sortie non affichée, car trop longue
```

L'installation nécessite d'avoir une connexion active à Internet. Depuis la version 0.2-6, l'extension est hébergée sur le **CRAN** (*Comprehensive R Archive Network*), le réseau officiel de diffusion des extensions de R. L'installation d'une extension se fait par la fonction install.packages, à qui on fournit le nom de l'extension. Ici on souhaite installer l'extension ggplot2 :

```
>install.packages("ggplot2", dep=TRUE)
```

L'option dep=**TRUE** indique à R de télécharger et d'installer également toutes les extensions dont celle choisie dépend pour son fonctionnement.
En général R va alors vous demander de choisir un miroir depuis lequel récupérer les données nécessaires. Choisissez de préférence un miroir le plus proche possible de l'endroit où vous vous trouvez.

Une fois l'extension installée, elle peut être appelée depuis la console ou un fichier script avec la commande :

```
>library(ggplot2)
```

À partir de là, on peut utiliser les fonctions de l'extension, consulter leur page d'aide en ligne, accéder aux jeux de données qu'elle contient, etc.

Pour mettre à jour l'ensemble des extensions installées, une seule commande suffit :

```
>update.packages()
```

Si on souhaite désinstaller une extension précédemment installée, on peut utiliser la fonction

```
>remove.packages("ggplot2")
```

Il est important de bien comprendre la différence entre install.packages et library. La première va chercher les extensions sur l'Internet et les installe en local sur le disque dur de l'ordinateur. On n'a besoin d'effectuer cette opération qu'une seule fois. La seconde lit les informations de l'extension sur le disque dur et les met à disposition de R. On a besoin de l'exécuter à chaque début de session ou de script.

Il est important de bien comprendre la différence entre install.packages et library. La première va chercher les extensions sur l'Internet et les installe en local sur le disque dur de l'ordinateur. On n'a besoin d'effectuer cette opération qu'une seule fois. La seconde lit les informations de l'extension sur le disque dur et les met à disposition de R. On a besoin de l'exécuter à chaque début de session ou de script.

3.2 À partir du CRAN :

Le CRANest le dépôt informatique de packages géré par le R core team. C'est là que la majorité des packages R sont rendus disponibles publiquement. Pour installer un package à partir d'un des miroirs du CRAN, il suffit d'utiliser la fonction install.packages comme dans cet exemple :

```
>install.packages("ggplot2")
```

Cette commande lance le téléchargement du fichier compressé d'installation du package, puis lance la décompression du fichier dans le dossier approprié. Bien sûr, il faut être connecté à internet pour que la commande fonctionne. La version du package correspondant à notre système d'exploitation est automatiquement sélectionnée :

- .tar.gz (package source) pour Linux / Unix,
- .zip pour Windows,
- .tgz pour Mac OS X / OS X / macOS.

Aussi, install.packages télécharge et installe par défaut les packages dont dépend le package installé. Cette option est très pratique, car certains packages dépendent d'un grand nombre de packages (ggplot2 en est un bon exemple). Le répertoire dans lequel les packages R sont

installés par défaut est identifié dans le premier élément du vecteur retourné par la commande R suivante :

```
.libPaths()
## [1] "C:/Users/Sophie/Documents/R/win-library/3.4"
## [2] "C:/Program Files/R/R-3.4.3/library"
```

Sur mon ordinateur, les packages provenant de l'installation de R sont installés dans le répertoire "C:/Program Files/R/R-3.4.3/library" et les packages supplémentaires que j'ai installés sont dans le répertoire utilisateur "C:/Users/Sophie/Documents/R/win-library/3.4". RStudio offre une fonctionnalité pour rendre encore plus conviviale l'installation de packages. Dans la sous-fenêtre « Packages », le bouton « Install » (en haut à gauche) permet d'installer des packages. En fait, cette fonctionnalité ne fait qu'écrire et soumettre la commande install.packages en fonction des options choisies par l'utilisateur.

3.3 À partir d'un fichier compressé d'installation

Afin d'installer un package non publicisé, qui n'est pas téléchargeable sur le web, il faut d'abord avoir sur son ordinateur le fichier compressé d'installation du package correspondant au système d'exploitation. Le package peut être installé avec la commande install.packages, comme dans cet exemple :

```
>install.packages ("C:/coursR/ggplot2_2.1.0.zip", repos = NULL)
```

Si nous possédons plutôt le package source, il est aussi possible d'installer le package à partir d'une plateforme Windows, mais à la condition d'avoir une installation des Rtools (voir le Guide d'installation ou de mise à jour de R et RStudio). Dans l'appel à la fonction install.packages, l'argument type = "source" doit être ajouté comme suit

```
>install.packages ("C:/coursR/ggplot2_2.1.0.tar.gz", type = "source", repos = NULL)
```

Cette technique d'installation à partir d'un fichier compressé est aussi utile lorsque nous devons télécharger manuellement le fichier compressé d'installation du package. Ça arrive par exemple lorsqu'un package est seulement disponible à partir du site web personnel de l'auteur du package. Nous devons aussi procéder à un téléchargement manuel lorsque nous avons besoin d'une ancienne version d'un package. Si nous laissons install.packages télécharger du CRAN un package, il téléchargera la dernière version. Il est possible que des

mises à jour d'un package rendent certains de nos programmes non fonctionnels et qu'en conséquence nous souhaitions continuer d'utiliser une version antérieure du package. Pour installer une version d'un package qui n'est pas sa dernière version, il faut aller sur sa page web du CRAN (par exemple http://CRAN.R-project.org/package=ggplot2), et aller télécharger manuellement la version dont nous avons besoin dans « Old Sources ». Ce type d'installation peut se réaliser par le menu « Install » de la sous-fenêtre « Packages » de RStudio, en sélectionnant : « Install from: > Package Archive File (…) ».

3.4 À partir d'un dépôt informatique autre que le CRAN

En plus du CRAN, on retrouve des packages R à d'autres endroits sur le web, notamment sur :

- le dépôt informatique de packages en bio-informatique Bioconductor : des instructions d'installation sont disponibles sur la page web : http://www.bioconductor.org/install/,
- un service web d'hébergement et de gestion de développement de logiciels tel que Githubou Bitbucket : le package devtools offre des fonctions pour télécharger et installer directement à partir de ces sites (ex. fonctions install_github et install_bitbucket).

3.5 Accéder au contenu d'un package R chargé

Une fois un package chargé en R avec la commande library, son contenu est accessible dans la session R. Nous avons vu dans des notes précédentes comment fonctionne l'évaluation d'expressions en R. Nous savons donc que le chargement d'un nouveau package ajoute un environnement dans le chemin de recherche de R, juste en dessous de l'environnement de travail. Le chargement de certains packages provoque aussi le chargement de packages dont ils dépendent. Ainsi, parfois plus d'un environnement est ajouté au chemin de recherche de R lors du chargement d'un package. L'environnement d'un package contient les fonctions publiques et les données du package.

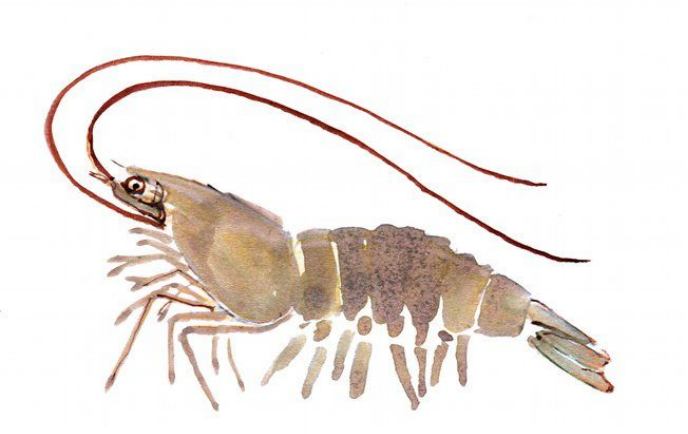
3.6 Installation de RStudio

Une fois **R** correctement installé, rendez-vous sur http://www.rstudio.com/products/rstudio/download/ pour télécharger la dernière version stable de **RStudio**. Plus précisément, il s'agit de l'édition *Open Source* de **RStudio Desktop** (en effet, il existe aussi une version serveur). Choisissez l'installateur correspondant à votre système d'exploitation et suivez les instructions du programme d'installation. Si vous voulez

tester les dernières fonctionnalités de **RStudio**, vous pouvez télécharger la version de développement (plus riche en fonctionnalités que la version stable, mais pouvant contenir des bugs) sur http://www.rstudio.com/products/rstudio/download/preview/.

CHAPITRE 4

IMPORTATION ET LECTURE DE DONNEES



4.1 Manipulation de données en R

Pour créer une variable, on utilise l'affectation avec les caractères < et -. R est typé mais il n'y a pas de déclaration explicite de typage : on peut mettre 2 dans x puis TRUE puis "kilo" sans aucune difficulté. Pour créer un vecteur, on utilise la fonction c() et la fonction matrix() pour créer une matrice ; la création d'une liste se fait avec list(). Le nombre d'éléments d'un vecteur ou d'une liste s'obtient par length(). Pour une matrice, on dispose des fonctions dim(), nrow(), ncol(), colnames() et row.names(). Les fonctions seq() (dont la version élémentaire est implémentée via la symbole :) et rep() permettent de remplir les variables, les lignes et les colonnes d'un vecteur. La virgule à l'intérieur des crochets dans une matrice permet de spécifier un élément, une ligne, une colonne alors que le symbole $ permet d'accéder aux élements par nom. Pour connaitre les premiers ou les derniers éléments, on peut utiliser les fonctions head() et tail(). Les fonctions print() et cat() permettent d'afficher texte et données. La fonction sprintf() sert à gérer les formats d'affichage.

R fonctionne en mode «vectoriel» c'est-à-dire que de nombreuses opérations et fonctions s'appliquent terme à terme en R. Les tests logiques avec <, >, ==, !, &, | et peuvent renvoyer des vecteurs logiques qu'on peut réutiliser comme indices. De plus FALSE vaut 0 et TRUE vaut 1, ce qui permet des comptages rapides dans les conditions. Les fonctions sort() et order() permettent respectivement de trier et de fournir l'ordre de tri des éléments. Avec attach() les colonnes d'un *data frame* sont accessibles via leur nom. Quand on n'en a plus besoin, on utilise detach().

4.2 Sauver les résultats d'une analyse

Les commandes et les résultats des analyses statistiques et les graphiques peuvent être copiés/collés dans un document.

Les résultats (sans les commandes) peuvent être copiés automatiquement dans un fichier texte grâce à "sink". Tapez:

```
>sink("monanalyse.txt",split=T)
>a=1:10
>mean(a)
>summary(a)
>sink()
```

Puis ouvrez le fichier "monanalyse.txt".

Les graphiques peuvent être sauvés directement dans des fichiers graphiques en utilisant les commandes postscript, jpeg ou png (voir l'aide en ligne de ces fonctions). Mentionnons le paquetage R2HTML qui permet de créer des rapports au format html de façon semi-automatique.

4.3 Organisation du travail

L'expérience prouve que la meilleure stratégie est de créer un répertoire (dossier) par analyse de données, et d'y disposer : (a) les fichiers de données brutes; (2) le fichier script contenant les commandes R; (3) le workspace et le(s) fichiers(s) résultats (textes et graphiques).

4.4 Opérations élémentaires

Toutes les opérations élémentaires peuvent être exécutées à l'invite de **R**, toutes les instructions seront validées en tapant sur la touche entrée du clavier. Attention certains caractères constituent des opérateurs pour R : $, [, [[, :, ?, <- (voire plus loin). Noter que R fait la différence entre une majuscule une minuscule.

4.4.1 Opérations élémentaires sur les scalaires: * : multiplication, - : soustraction, + : addition, / : division, ^ : puissance.

```
> 4+8
```

La réponse de R ne se fait pas attendre :

```
[1]  12
```

Bien, nous savons désormais que R sait faire les additions à un chiffre[2]. Nous pouvons désormais continuer avec d'autres opérations arithmétiques de base :

```
> 8 – 12
 [1] -4

> 14 * 25
[1] 350

> -3/10
[1] -0.3

> (6+5*2)/2
[1]   8

> 2^2 (puissance)
[1]   4
```

```
> sqrt (9) (racine carrée)
[1]  3
```

4.4.2 Opérations avec affectation (avec ou sans affichage)

```
>x=2+4
>x
[1]  6
```

```
> (x=2+4)       # Avec affichage du résultat
 [1]  6
```

Remarque : A l'aide des flèches de déplacement du clavier vous pouvez rappeler les dernières instructions tapées précédemment, que vous pouvez alors facilement réexécuter ou modifier.

Lorsqu'on fournit à R une commande incomplète, celui-ci nous propose de la compléter en nous présentant une invite de commande spéciale utilisant les signe +. Imaginons par exemple que nous avons malencontreusement tapé sur Entrée alors que nous souhaitions calculer 4*3 :

```
> 4 *
+
```

On peut alors compléter la commande en saisissant simplement 3 :

```
> 4 *
+ 3
[1] 12
```

2. La présence de nombres entre crochets [1], [18] …etc en début de chaque ligne, indiquent la position du premier élément de la ligne dans notre vecteur.

4.4.3 Créer, lister et effacer les objets en mémoire

Un objet peut-être créé à l'aide de l'opérateur « assigner » qui s'écrit avec le symbole < suivi du signe moins (–), ce symbole pouvant être orienté dans un sens ou dans l'autre [5] : Cet objet est stocké dans la mémoire vive et peut être modifié en lui assignant une autre valeur.

```
> n <- 100
> n
[1] 100

> 100 -> n
> n
 [1] 100
```

En tapant le nom d'un objet le contenu de ce dernier est affiché (quand cet objet ne requiert pas des arguments).

```
> n
[1] 100

> n <- 100/2+5
> n
[1]  55

> x <- 1
> x <- 10
> x
 [1] 1

> x
[1] 10
```

Si l'objet existe déjà, sa valeur précédente est effacée (la modification n'affecte que les objets en mémoire vive, pas les données sur le disque). La valeur ainsi donnée peut être le résultat d'une opération et/ou d'une fonction :

```
> n <- 10 + 2
> n
[1] 12

> n <- 3 + rnorm (1)
> n
[1] 2.208807
```

La fonction rnorm(1) génère une variable aléatoire normale de moyenne zéro et variance unité. On peut simplement taper une expression sans assigner sa valeur à un objet, le résultat est alors affichée à l'écran mais n'est pas stocké en mémoire :

```
> (10 + 2) * 5
[1] 60
```

Dans nos exemples, on omettra l'assignement si cela n'est pas nécessaire à la compréhension. La fonction ls permet d'afficher une liste simple des objets en mémoire, c'est-à-dire que seuls les noms des objets sont affichés.

```
> name <- "Carmen"; n1 <- 10; n2 <- 100; m <- 0.5
 > ls()
[1]  "m"  "n1"  "n2"  "name"
```

Notons l'usage du point-virgule pour séparer des commandes distinctes sur la même ligne. Si l'on veut lister uniquement les objets qui contiennent un caractère donnée dans leur nom, on utilisera alors l'option pattern (qui peut s'abréger avec pat) [5] :

```
> ls(pat = "m")
[1] "m" "name"
```

Pour restreindre la liste aux objets dont le nom commence par le caractère en question :

```
> ls(pat = "^m")
 [1] "m"
```

La fonction ls.str affiche des détails sur les objets en mémoire :

```
> ls.str()
m : num 0.5
n1 : num 10
n2 : num 100
name : chr "Carmen"
```

L'option pattern peut également être utilisée comme avec ls. Une autre option utile de ls.str est max.level qui spécifie le niveau de détails de l'affichage des objets composites. Par défaut, ls.str affiche les détails de tous les objets contenus en mémoire [5], y compris les colonnes des jeux de données, matrices et listes, ce qui peut faire un affichage très long. On évite d'afficher tous les détails avec l'option max.level = -1 :

```
> M <- data.frame(n1, n2, m)
> ls.str(pat = "M")
M : 'data.frame':       1 obs. of 3 variables:
 $ n1: num 10
 $ n2: num 100
 $ m : num 0.5
> ls.str(pat="M", max.level=-1)
M : 'data.frame':       1 obs. of 3 variables:
```

Pour effacer des objets de la mémoire, on utilise la fonction rm:rm(x) pour effacer l'objet x, rm(x, y) pour effacer les objets x et y, rm(list=ls()) pour effacer tous les objets en mémoire [5]; on pourra ensuite utiliser les mêmes options citées pour ls() pour effacer sélectivement certains objets : rm(list=ls(pat = "^m")).

Il est possible d'afficher le nom des objets stockés ou/et leur contenu dans la mémoire vive :

```
> n <- 12
> m <-8
> s <- 12+8
> name <- "Selma"
ls ()
[1]  "n"  "m"  "s"  "name"

> ls.str()
m : num 8 (num, variable numérique)
n : num 12
s : num 20
nom : chr "Selma" (chr,variable caractère)
```

4.4.4 Création et opérations sur les vecteurs

Création via une saisie :

Le vecteur est à la fois l'objet de base dans R le plus simple et le plus fondamental du langage R. Il se crée grâce à la fonction c() concatène des scalaires ou des vecteurs [7], qui prend comme arguments les éléments du vecteur. Tous ces éléments doivent être du même type : valeurs numériques, chaînes de caractères ou encore niveaux d'un facteur.

Pour en connaître le type, on utilise class()

EXEMPLE(S)

Pour créer un vecteur numérique :

```
> vecteur <- c (7,9,4,12,18)
> vecteur
[1] 7 9 4 12 18
```

Pour créer un vecteur de chaînes de caractères :

```
> vecteur <- c ("H","C","I","G","F")
> vecteur
[1] "H" "C" "I" "G" "F"
```

Pour créer un facteur :

```
 > vecteur <- factor(c ("niv1","niv2","niv2","niv3","niv1"))
 > vecteur
[1] niv1 niv2 niv2 niv3 niv1
Levels : niv1 niv2 niv3
```

Création via une saisie console

Utilisation de la commande scan() Tant que saisie de valeurs, la taille du vecteur s'accroit Arrêt dès que l'utilisateur fait ENTREE 2 fois consécutivement

Création via la commande c() directement dans le code prog.

```
v <- c(1.2,2.5,3.2,1.8)     #crée un vecteur de réels de taille 4
length(v)                   #renvoie le nombre d'éléments
```

C'est l'approche que tout le monde adopte pour la saisie de valeurs, ça ne pose pas de soucis parce R est un langage interprété. De fait, nous allons utiliser un autre éditeur de code (soit celui de R, soit des outils tels que R-Studio, soit Eclipse avec StatET…).

Il existe des fonctions ou des abréviations qui permettent de simplifier la création de certains vecteurs usuels:

EXEMPLE(S)

```
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> seq (from=1, to=3, by=0.25)
[1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00
> LETTERS [1:5]
[1] "A" "B" "C" "D" "E"
```

4.4.5 Structure d'un fichier texte

Dès lors, avant d'importer un fichier texte dans R, il est indispensable de regarder comment ce dernier est structuré. Il importe de prendre note des éléments suivants :

- La première ligne contient-elle le nom des variables ? Ici c'est le cas.
- Quel est le caractère séparateur entre les différentes variables (encore appelé séparateur de champs) ? Dans le cadre d'un fichier CSV, il aurait pu s'agir d'une virgule ou d'un point-virgule.
- Quel est le caractère utilisé pour indiquer les décimales (le séparateur décimal) ? Il s'agit en général d'un point (à l'anglo-saxonne) ou d'une virgule (à la française).

- Les valeurs textuelles sont-elles encadrées par des guillemets et, si oui, s'agit-il de guillements simple (') ou de guillemets doubles (") ?
- Pour les variables textuelles, y a-t-il des valeurs manquantes et si oui comment sont-elles indiquées ? Par exemple, le texte NA est parfois utilisé.

Il ne faut pas hésiter à ouvrir le fichier avec un éditeur de texte pour le regarder de plus près.

4.4.6 Importer des données

L'import et l'export de données depuis ou vers d'autres applications, est couvert en détail dans l'un des manuels officiels (en anglais) nommé R Data Import/Export et accessible, comme les autres manuels, à l'adresse suivante : http://cran.r-project.org/manuals.html.
Le langage R peut lire des données provenant de sources externes sous forme de fichiers, comme il peut en créer et sauvegarder dans des formats transportables.
La fonction qui permet la lecture de ces fichiers est read.table, R dispose d'autres variantes de cette fonction (scan(), read.fwf,…).

Ces fonctions nécessitent des arguments qui permettent la lecture de tous les fichiers (csv, texte délimité, largeur fixe…) tout en précisant le format interne du fichier (présence de noms de variables, type de séparateur, présence de variables ''caractère'',…….).
Néanmoins le format le plus simple à importer dans R reste les fichiers sauvegardés dans le format '.txt'.

Importer des données est souvent l'une des premières opérations que l'on effectue lorsque l'on débute sous R, et ce n'est pas la moins compliquée. En cas de problème il ne faut donc pas hésiter à demander de l'aide par les différents moyens disponibles avant de se décourager.

- **Importer des données**

Comment importer rapidement plusieurs fichiers CSV

Avec R, il est tout à fait possible de réaliser une multiple importation des fichiers CSV (autres types de fichiers). La fonction ci-dessous utilise les fonctions bind_rows et la syntaxe de la librairie dplyr.

Dans la plupart des cas un fichier de données est créé à partir d'un tableur ou un traitement de texte.

```
multmerge <- function(mypath = getwd()){require(dplyr)dataset <- list.files(path=mypath, full.names=TRUE, pattern="\.csv") %>% lapply(read.csv, header=TRUE, sep="t") %>% bind_rows()dataset}
```

Dans cette fonction, l'objet mypath indique l'emplacement où se trouvent tous les fichiers CSV. Par défaut, la fonction va chercher dans l'espace de travail actuel (getwd). Il est possible de le modifier manuel comme ci-dessous:

```
mydata <-  multmerge(mypath="Nom/Du/Chemin/Des/Fichiers")
```

Le fichier data1.xls a été créé avec Excel et contient les données ci-après :

Pour l'importer dans R, commencer par le sauvegarder sous forme texte dans votre répertoire de travail (par exemple, c:\userdata) comme suit :

Saisie de la série

Diagnostique de l'Hypertension artérielle en relation avec le tabagisme chez les personnes âgées de 40 à 65 ans.

Je saisie les données dans un fichier excel format txt (Fichier texte,séparateur tabulation)

Pour importer les données, j'utilise la commande read.table qui a pour effet de créer un tableau de données et est donc le moyen principal pour lire des fichiers de données au format texte.

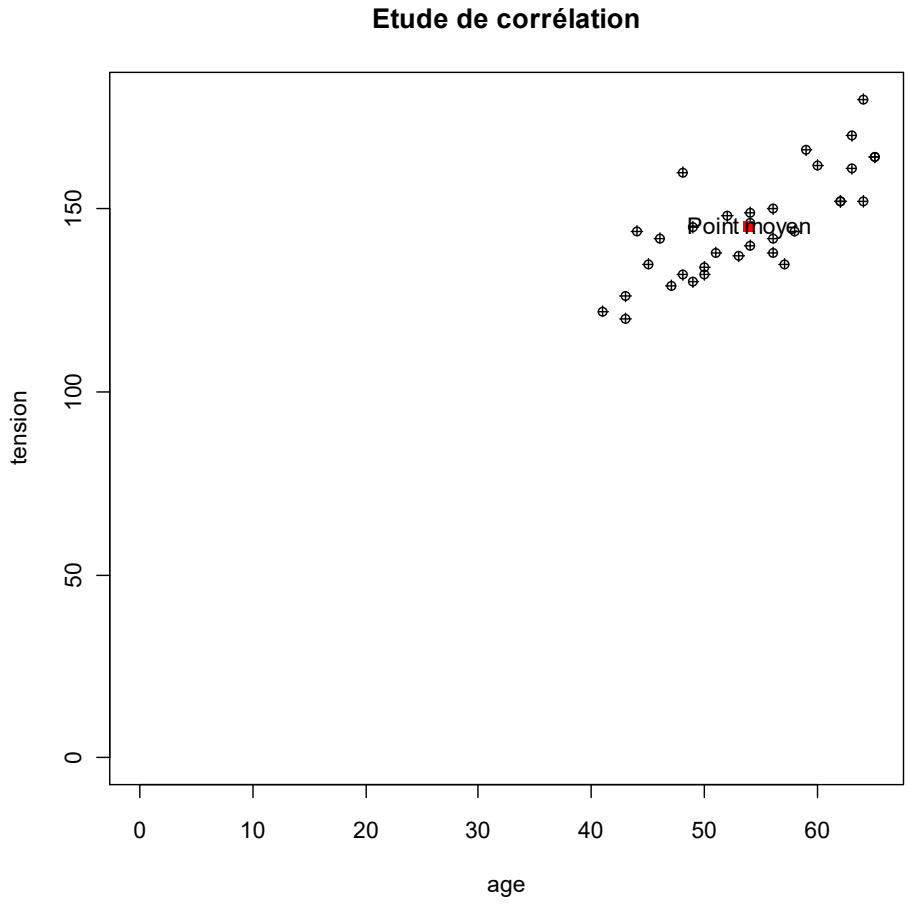
```
> risque=read.table(file.choose(),header=T)
> risque
```

	tension	age	fumeur
1	146	54	1
2	129	47	1
3	162	60	1
4	160	48	1
5	144	44	1

6	180	64	1
7	166	59	1
8	138	51	1
9	140	54	1
10	134	50	1
11	145	49	1
12	142	46	1
13	150	56	1
14	149	54	1
15	132	48	1
16	126	43	1
17	170	63	1
18	135	45	0
19	122	41	0
20	130	49	0
21	148	52	0
22	152	64	0
23	138	56	0
24	135	57	0
25	152	62	0
26	164	65	0
27	142	56	0
28	144	58	0
29	137	53	0
30	132	50	0
31	120	43	0
32	161	63	0
33	152	62	0
34	164	65	0

```
> attach(risque)
> ?plot
> plot(age,tension)
> ?par
> plot(age,tension,pch=16,main="tension en fonction de l'âge")
```

```
> plot(age,tension,pch=20,main="tension en fonction de l'âge")
> mean(age)
[1] 53.85294
> mean(tension)
[1] 145.3235
> plot(age,tension,pch=15,main="tension en fonction de l'âge")
> points(53.85294,145.3235,pch=16,col="red")
> ?text
> text(53.85294,145.3235,labels="point moyen",pos=1)
> ?cor
```



Pour calculer le coefficient de corrélation, préciser la méthode: Pearson (paramètrique), Spearman (non paramétrique) ou Kendall (non paramétrique).Le test de Kendall utilise davantage d'informations que celui de Spearman.

```
> cor(age,tension,methode="pearson"
+ > > cor(age,tension,methode="pearson"
Erreur : '>' inattendu(e) dans ">"
> > cor(age,tension,methode="pearson">
> cor(age,tension,methode="pearson")
Erreur dans cor(age, tension, methode = "pearson") :
```

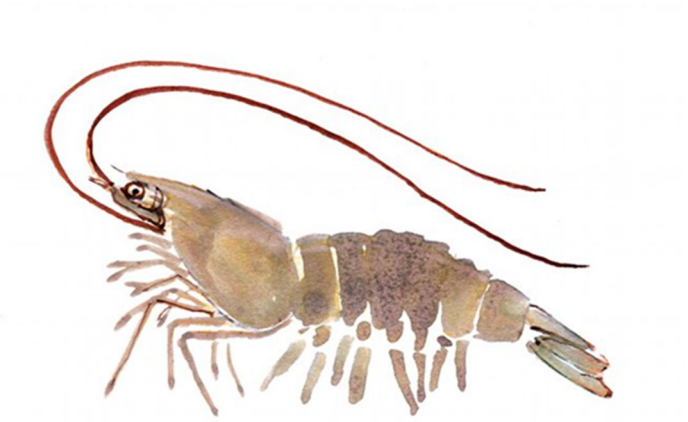
```
argument(s) inutilisé(s) (methode = "pearson")
> cor(age,tension,method="pearson")
[1] 0.7867896
```

Coefficient de corrélation élevé et positif, donc bonne corrélation positive

```
> ?cor.test
> cor(age,tension,method="pearson")
[1] 0.7867896
```

CHAPITRE 5

ACCÈS AUX VARIABLES



5. Accès aux variables

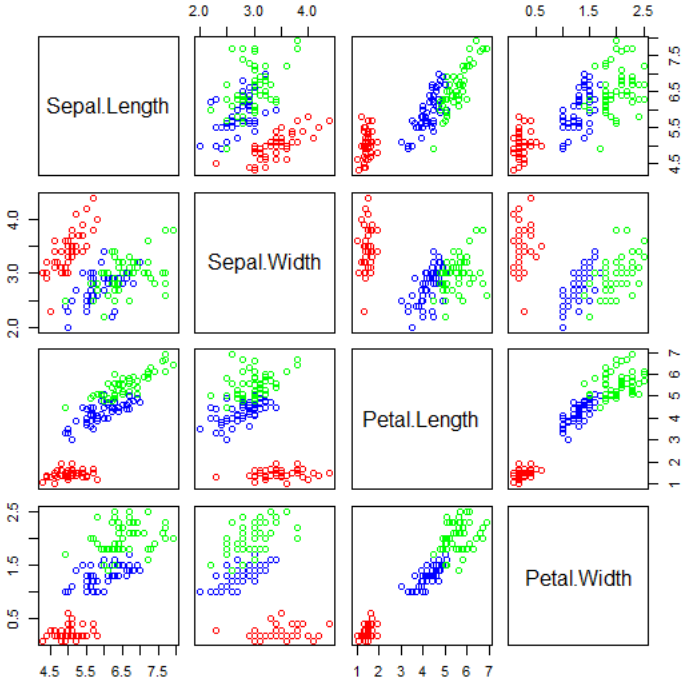
5.1 Accès aux données intégrées dans R (dans les packages de R)

Des données peuvent être intégrées dans des packages (pour les démonstrations, les exemples d'utilisation, etc.). Au démarrage de R, un certain nombre de packages « de base » sont automatiquement chargées, entres autres le package « datasets ».

De fait, plusieurs jeux de données sont directement accessibles dans R. On peut en obtenir la liste avec la commande data().

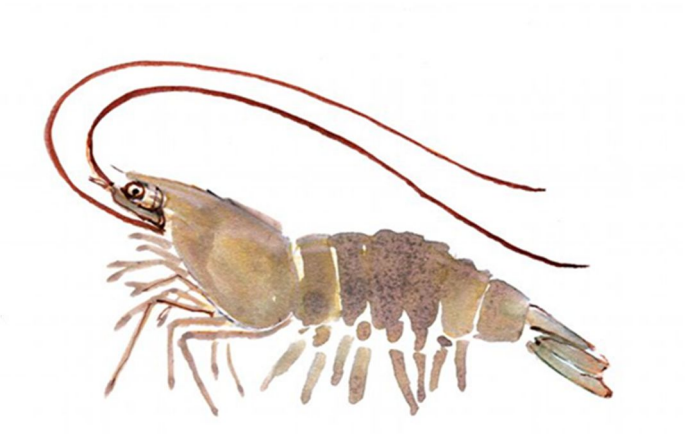
Exemple : les données « iris »

```
> # iris
> data(iris)
> #les 3premières lignes
> print(head(iris,n=3))
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa

> print(summary(iris))
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   : 4.300   Min.   : 2.000   Min.   : 1.000   Min.   : 0.100   setosa     :50
 1st Qu.:5.100   1st Qu. :2.800   1st Qu. :1.600   1st Qu. :0.300   versicolor :50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica  :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu. :3.300   3rd Qu. :5.100   3rd Qu. :1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500

> pairs(iris[1:4],col=c("red","blue","green")[iris$Species])
```


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0 3.0 4.0
0.5 1.5 2.5
4.5 5.5 6.5 7.5
2.0 3.0 4.0
1 2 3 4 5 6 7
0.5 1.5 2.5
4.5 5.5 6.5 7.5
1 2 3 4 5 6 7

CHAPITRE 6

FONCTIONS SIMPLES



6.1 Référence des fonctions de R les plus courantes

6.1.1 La plupart des fonctions de R ont une documentation en ligne

Cette liste de commandes (non exhaustive) est une adaptation de la ref-card R de Tom Short [10]. Elle est un relais entre ce que cherche l'utilisateur et l'aide de R : les nombreuses options disponibles ne sont que rarement spécifiées ici.

help(sujet) documentation sur un sujet. Flèches haut et bas pour se déplacer, touche q pour quitter

?topic idem

help.search("sujet") recherche dans l'aide

apropos("sujet") le nom de tous les objets dans la liste de recherche qui correspondent à l'expression régulière « sujet »

help.start() lance la version HTML de l'aide (indispensable; le moteur de recherche intégré nécessite Java installé sur votre ordinateur) example (function) exécute l'exemple donné en bas de la page d'aide de la function indiquée.

6.1.2 Fonctions de base

<- et -> assignation dans le sens de la flèche (a <-b équivaut donc à b->a) ; e.g.: x<-0; x+1->x (met x à 0, puis additionne x et 1 pour mettre le résultat dans x) ;

NULL l'ensemble vide/ l'objet nul (objetréservé);

NA absensededonnées/valeur manquante (Not Available) ;

TRUE/FALSE vrai et faux logiques Inf valeur infinie

"abc" une chaîne de 3 caractères ;

str(a) affiche la structure d'un objet a ;

head(a) affiche les premiers éléments de l'objet a (selon son type : vecteur, matrice, tableau, etc.)

summary(a) propose un « résumé » de a, généralement un résumé statistique, mais c'est une fonction générique (fonctionne différemment selon la classe de a) ; la plupart du temps un résumé statistique ;

search() affiche l'itinéraire de recherche

ls() liste les objets de la liste de recherche; spécifier e.g. pat="MonTexte" pour chercher selon un patron ;

ls.str() str() pour chaque variable de la liste de recherche ;

dir() lister les fichiers dans le dossier (directory) en cours ;

methods(a) afficher les méthodes S3 de a ;

methods(class=class(a)) lister toutes les méthodes permettant de traiter les objets de la classe de l'objet a ;

options(...) définit ou examine de nombreuses options globales; options fréquentes : width (largeur en nombre de caractères de la fenêtre des résultats), digits (nombre de chiffres significatifs à l'affichage), error (traitement des erreurs) ;

library(x) charge des packages additionnels ; library(help=x) liste les jeux de données et fonctions du package x ;

attach(x) ajoute le contenu de x dans la liste de recherche de R ; x peut être une liste, une data frame, ou un fichier de données R créé avec save. Utilisez search() pour montrer la liste de recherche ;

detach(x) enlève le contenu de x de la liste de recherche de R; x peut être un nom ou une chaîne de caractères désignant un objet préalablement attaché ou un package ;

with(x, expr) évalue la commande expr en ayant placé l'objet x dans l'itinéraire de recherche.

rm(x), remove(x) détruit l'objet x ;

setwd(dir), getwd(dir) affecte ou récupère le chemin du répertoire de travail courant ;

function(arglist) { expr return(result)} définition de function if, while, repeat, etc.

length(x) nombre d'éléments de x

dim(x) nombre de dimensions d'un objet

dimnames(x) noms des dimensions d'un objet

names(x) manipulation de l'attribut names de l'objet x ;

setNames(noms, x) attribue le vecteurs de noms noms au vecteur x ;

nrow(x), ncol(x) nombre de lignes et de colonnes ;

class(x) classe de l'objet x ;

unclass(x) supprime l'attribut class de la variable x ;

attr(x, which) récupère ou spécifie les attributs de x décrits par which ;

attributes(x) récupère ou spécifie tous les attributs de x ;

q() quitter R (répondre y (yes) et Entrée pour confirmer) ;

Entrée et sortie

load() charge le jeu de données écrit avec save ;

data(x) charge le jeu de données spécifié

read.table(file) lit un fichier au format tabulaire et en fait un data frame; le séparateur de colonne par défaut sep="" désigne n'importe quel espacement; utilisez header=TRUE pour prendre la première ligne comme titre (header) de colonne; utilisez as.is=TRUE pour empêcher les vecteurs de caractères d'être transformés en factors; utilisez skip=n pour ignorer les n premières lignes ; consultez l'aide pour les options concernant le nommage des colonnes, le traitement des valeurs manquantes (NA), etc.

read.csv2("filename",header=TRUE) idem mais avec des options pré-définies pour lire les fichiers CSV

read.delim("filename",header=TRUE) idem mais avec des options pré-définies pour lire les fichiers dont les valeurs sont séparées par des tabulations ;

read.fwf(file,widths,header=FALSE,sep=" ",as.is=FALSE) lit un tableau dont toutes les colonnes ont la même largeur (fwf: fixed width format); widths est un vecteur d'entiers donnant la largeur des colonnes dans le fichier ;

save("fichier", x,y) enregistre les objets x et y dans le fichier, au format binaire XDR propre à R ;

save.image("fichier") enregistre tous les objets cat(..., file="", sep=" ") affiche les arguments après les avoir converti en caractères ; sep est le séparateur entre les arguments ;

print(a, ...) affiche les arguments; fonction générique (fonctionne différemment selon la classe de a) ;

format(x,...) formate un objet R pour un affichage personnalisé ;

write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ") affiche x après l'avoir converti en data frame; si quote est TRUE, les colonnes de caractères ou de factors sont entourés par des guillemets; sep est le séparateur de colonnes. Avec file= suivi du chemin d'un fichier, écrit sur le disque dur.

La plupart des fonctions d'entrée/sortie ont un argument file. Cela peut souvent être une chaîne de caractères nommant un fichier ou une connexion. Sous Windows, la connexion peut aussi être utilisée avec description ="clipboard" pour lire un tableau copié d'un tableur par le presse-papier

x <- read.delim("clipboard") pour lire un tableau copié par le presse-papier depuis un tableur ;

write.table(x,"clipboard",sep="\t",col.names =NA) pour écrire un tableau vers le presse-papier pour un tableur.

Création de données

c(...) fonction combinant les arguments pour former un vecteur; avec recursive=TRUE va dans les listes pour combiner leurs éléments en un seul vecteur (plutôt qu'en un vecteur de listes) ;

de:vers génère une séquence d'entiers; ":" est prioritaire: 1:4 + 1 vaut "2,3,4,5"

seq(from,to) génère une séquence; by= spécifie l'incrément; length= spécifie la longueur ;

seq(along=x) génère une suite 1, 2, ..., length(x) ; utile pour les boucles for ;

rep(x,times) répète times fois la valeur x; utilisez each=n pour répéter n fois chaque élément de x; rep(c(1,2,3),2) vaut 1 2 3 1 2 3; rep(c(1,2,3),each=2) vaut 1 1 2 2 3 3

data.frame(...) crée un data frame avec les arguments (nommés ou non); e.g: data.frame(v=1:4, ch=c("a", "b", "c", "d"), lettre= "A"); les vecteurs plus courts (ici: "A") sont réutilisés (recyclés) plusieurs fois pour atteindre la longueur du vecteur le plus long ; à la différence d'une matrix, un data.frame est un tableau dont les colonnes peuvent être de types différents ;

list(...) crée une liste avec les arguments (nommés ou non, qui peuvent être de longueur différente); e.g.: list(a=c(1,2),b="hi",c=3);

matrix(x,nrow=,ncol=) crée une matrice (tous les éléments sont de même type); les éléments se répètent s'ils sont trop courts ;

factor(x,levels=) transforme un vecteur x en factor (les niveaux sont indiqués par levels=) ;

rbind(...) combine les arguments par ligne (row) ;

cbind(...) combine les arguments par colonne ;

Extraction de données

Indexer des listes

x[i] le ou les éléments i de la liste (renvoyé(s) sous forme de liste, à la différence des cas suivants; fonctionne comme pour les vecteurs) ;

x[[n]] n ^{ième} élément de la liste ;

x[["nom"]] l'élément nommé "nom" .

x$nom l'élément nommé "nom"

Indexer des vecteurs

x[n] n ième élément

x[-n] tous sauf le n ième élément ;

x[1:n] les n premier éléments ;

x[-(1:n)] les éléments de n+1 à la fin ;

x[c(1,4,2)] des éléments spécifiques ;

x["nom"] l'élément nommé "nom" ;

x[x > 3] tous les éléments plus grands que 3 ;

x[x > 3 & x < 5] tous les éléments plus grands que 3 et plus petits que 5 ;

x[x %in% c("a","and","the")] les éléments appartenant à l'ensemble donné.

Indexer des matrices

x[i,j] l'élément de la ligne i, colonne j ;

x[i,] toute la ligne i ;

x[,j] toute la colonne j ;

x[,c(1,3)] les colonnes 1 et 3 ;

x["nom",] la ligne nommée "nom".

Indexer des data.frame (comme pour les matrices plus ce qui suit)

x[["nom"]] la colonne nommée "nom"

x$nom la colonne nommée "nom"

Conversion d'objets

as.data.frame(x), as.numeric(x), as.logical(x), as.character(x), ... conversion de type, e.g.: as.logical(x) convertit x en TRUE ou FALSE) ; pour la liste complète, faites methods(as)

Information sur les objets

is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x), ... tests de type; renvoie TRUE ou FALSE; pour une liste complète, faites methods(is) ;

length(x) nombre d'éléments dans x ;

dim(x) récupère ou définit (dim(x) <- c(3,2)) les dimensions d'un objet

nrow(x) et NROW(x) nombre de lignes ; NROW(x) considère un vecteur comme une matrice

ncol(x) et NCOL(x) idem pour les colonnes

class(x) récupère ou définit la classe de x; class(x) <- "maclasse"

unclass(x) enlève l'attribut de classe de x ;

attr(x,which=) récupère ou définit un attribut de x ;

attributes(x) récupère ou définit la liste des attributs de x ;

which.max(x) trouve l'indice du plus grand élément de x ;

which.min(x) trouve l'indice du plus petit élément de x ;

rev(x) renverse l'ordre des éléments de x ;

sort(x) trie les éléments de x par ordre croissant ; pour l'ordre décroissant : rev(sort(x)).

Mathématiques

max(x) maximum des éléments de x ;

min(x) minimum des éléments de x ;

range(x) mini et maxi: c(min(x), max(x)) ;

sum(x) somme des éléments de x ;

diff(x) différence entre chaque élément de x et son prédécesseur ;

prod(x) produit des éléments de x ;

mean(x) moyenne des éléments de x ;

median(x) médiane des éléments de x ;

quantile(x,probs=) quantiles correspondant aux probabilités données; le paramètre par défaut probs=c(0,.25,.5,.75,1) donne les quartiles ;

weighted.mean(x, w) moyenne pondérée de x (pondération par w) ;

rank(x) rang des éléments de x ;

var(x) ou cov(x) variance des éléments de x (calculé avec n−1 au dénominateur); si x est une matrice ou un data.frame, la matrice de variance-covariance est calculée ;

sd(x) écart-type (standard deviation) de x ;

cor(x) matrice de corrélation de x (pour une matrice ou un data.frame) ;

var(x, y) ou cov(x, y) covariance entre x et y, ou entre les colonnes de x et celles de y si ce sont des matrices ou des data frames ;

cor(x, y) coefficient de corrélation linéaire entre x et y, ou matrice de corrélation si ce sont des matrices ou des data frames.

round(x, n) arrondit les éléments de x à n décimales ;

pmin(x,y,...) un vecteur dont le ième élément est le minimum des valeurs x[i], y[i], ...

pmax(x,y,...) idem pour le maximum ;

union(x,y), intersect(x,y), setdiff(x,y), union et intersection d'ensembles;

setdiff(x,y) trouve les éléments de x qui ne sont pas dans y ;

abs(x) valeur absolue ;

filter(x,filter) applique un filtre linéaire à une série temporelle; e.g., pour une moyenne mobile sur trois périodes: filter(x, c(1/3, 1/3, 1/3)) ;

na.rm=FALSE De nombreuses fonctions mathématiques ont un paramètre na.rm=TRUE (non available removed) pour enlever les données manquantes (NA) avant le calcul.

Matrices

t(x) transposée ;

diag(x) diagonale ;

%*% multiplication de matrices ;

solve(a,b) trouve x tel que a %*% x = b ;

solve(a) matrice inverse de a ;

rowsum(x) somme par ligne d'une matrice ou d'un objet similaire ;

colsum(x) somme par colonne ;

rowMeans(x) moyenne des lignes d'une matrice ;

colMeans(x) idem pour les colonnes.

Périphériques graphiques

windows() ouvre une fenêtre graphique sous Windows

x11() idem sous GNU/linux ou MacOSX

pdf(file), png(file), jpeg(file), bmp(file), tiff(file) se prépare à écrire les instructions graphiques qui suivront dans le fichier file, au format désigné (pdf ou png recommandés);

width= et height= fixent les dimensions ;

dev.off() ferme la fenêtre graphique ou le fichier graphique spécifié (par défaut: celui en cours); cf. aussi dev.cur, dev.set ;

Graphiques

plot(x) graphique de x (fonction générique ayant des effets différents selon l'objet)

plot(x, y) nuage de points ;

hist(x) histogramme des fréquences de x ;

barplot(x) diagramme en barres ;

pie(x) diagramme circulaire (« camembert ») ;

boxplot(x) diagramme en boîte [boîte à moustaches]; la boîte et son milieu montrent les 3 quartiles; les moustaches (whisker) un intervalle de confiance de 95% pour la médiane (s'il y a des valeurs en dehors, elles sont affichées) ;

sunflowerplot(x, y) comme plot(x,y) mais les points qui se superposent exactement sont représentés avec des « fleurs » (un pétale par valeur répétée) ;

stripchart(x, method="stack") superpose les valeurs identiques du vecteur x; e.g. stripchart(round(rnorm(30,sd=5)), method="stack") ;

coplot(y~x | a) nuage des points de coordonnées x, y pour chaque valeur ou intervalle de valeur de a ;

mosaicplot(table(x,y)) version graphique de la table de contingence (les surfaces des carrés sont proportionnelles aux effectifs) ;

image(table(x,y)) similaire mais les effectifs influencent la couleur et non la surface ;

pairs(x) tableau des nuages de points entre toutes les paires de colonnes de x ;

plot.ts(x) pour une ou des série(s) temporelle(s) (classe "ts"), valeurs de x en fonction du temps ;

ts.plot(x) idem mais les séries peuvent ne pas commencer ou finir en même temps ;

qqnorm(x) nuage des quantiles observés contre quantiles théoriques; si x suit une loi normale, une droite; comparer qqnorm(rnorm(100)) et qqnorm(1:100) ;

qqplot(x, y) quantiles de y en fonction des quantiles de x

Les paramètres suivants sont communs à de nombreuses fonctions graphiques :

add=TRUE ajoute sur le graphique précédent ;

axes=FALSE ne trace pas les axes ;

type="p" type de représentation des coordonnées; "p": points, "l": lignes, "b": (both) points et lignes, "o": idem mais lignes sur (over) les points, "h": bâtons, "s": escaliers (données en haut des barres verticales), "S": idem (données en bas des barres), "n": définit la zone de coordonnées mais ne trace rien (utiliser après les commandes graphiques de bas niveau qui suivent) ;

xlim=, ylim= limites des zones du graphique, e.g. xlim=c(1,5) ;

xlab=, ylab= titre des axes (caractères) ;

main= titre du graphique (caractères) ;

sub= sous-titre du graphique (caractères)

Paramètres graphiques

points(x, y) ajoute des points (type= peut être utilisé) ;

lines(x, y) ajoute des lignes ;

text(x, y, labels, ...) ajoute du texte (labels) aux coordonnées ; e.g. : plot(x, y, type="n"); text(x, y, names) ;

abline(a,b) trace une droite (de forme y=a+b*x) ;

abline(lm.obj) trace la droite de régression du modèle linéaire lm.obj ;

legend(x, y, legend) ajoute une légende au point (x,y) avec les symboles donnés par legend ;

axis(side) ajoute un axe en bas (side=1), à gauche (2), en haut (3) ou à droite (4); optionnels: at= pour les coordonnées des graduation, labels= pour leur texte ;

box() encadre le graphique ;

rug(x) ajoute près de l'axe des abscisses une petite barre pour chaque valeur de x ;

par(...) définit les paramètres suivants pour les graphiques à venir, e.g. par(cex=2); nombre de ces paramètres peuvent aussi être utilisés directement avec une commande graphique de haut ou bas niveau, e.g. plot(x, cex=2) ; liste complète avec help(par) ;

cex taille du texte et des symboles par rapport à la valeur par défaut (character expansion)

col couleur(s) des symboles et lignes; e.g. col="red", "blue" cf. colors(); e.g. pour créer des vecteurs de 5 couleurs, faire suivre col= de gray(0:5/5), rainbow(5) ou terrain.colors(5)

lty type de ligne; 1: pleine, 2: tirets, 3: pointillés, 4: tiretspoints, 5: longs tirets, 6: tiret-court/tiret-long; (configurable) ;

lwd largeur des lignes ;

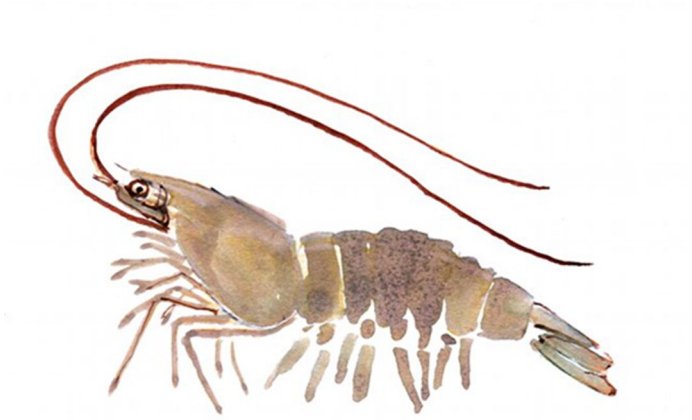
pch type de symboles pour les points (code entier de 1 à 25, ou caractère entre "") ;

xaxt="n" ne trace pas l'axe des abscisses ;

yaxt="n" ne trace pas l'axe des ordonnées.

CHAPITRE 7

BOUCLES ET AUTRES FONCTIONS



7.1 Quelques éléments de programmation

7.1.1 Boucles

Nous traitons succinctement dans cette section des instructions de sélection et de répétitions. Il s'agit des commandes if, while, for

```
> bool=T
>i=0
> while(bool==T) {i=i+1; if (i>10) {bool=F}}
> i
> s=0
> x=rnorm(10000)
> for (i in 1:10000) {s=s+x[i]}
> s
> un=rep(1,10000)
> t(un)%*%x
> s=0
> system.time(for (i in 1:10000) {s=s+x[i]})[3]
> system.time(t(un)%*%x)[3]
```

R peut avoir un problème de mémoire si vous faites appel à un nombre très élevé de boucles, même si elles contiennent des instructions très simples. En effet, comme les deux dernières commandes le montrent, l'utilisation de boucles est très couteuse en temps de calcul. Il est ainsi indispensable de limiter l'utilisation des boucles en les remplaçant par les outils du calcul matriciel (les opérateurs matriciel de R utilisent des boucles C beaucoup plus rapides).

Boucle **for** Voici un exemple tout simple pour illustrer la syntaxe d'une boucle for :

```
> x <- rep(0, 10)
> y <- x
> for (i in 1:10) {
+ x[i] <- i
+ y[i] <- i^2
+ } > x

[1] 1 2 3 4 5 6 7 8 9 10
> y
[1] 1 4 9 16 25 36 49 64 81 100
```

Les accolades ne sont pas nécessaires s'il n'y a qu'une instruction. Les boucles **for** peuvent être évitées du fait que R travaille vectoriellement.

Boucle **while** Voici maintenant un exemple pour illustrer la syntaxe d'une boucle while :

```
> proba <- dpois(0:100, lambda = 20)
> quantile <- 0
> queue <- 1
> while (queue > 0.05) {
+ queue <- queue - proba[quantile + 1]
+ quantile <- quantile + 1
+ }
> quantile
[1] 29

> queue
[1] 0.03433352
```

7.1.2 Tests

La syntaxe d'une boucle *si alors sinon* est la suivante :

```
> x <- rpois(1, 20)
> if (x%%2 == 0) {
+ pair <- TRUE
 + } else {
+ pair <- FALSE
 + }
> x
[1] 15

> pair

[1] FALSE
```

L'instruction sinon n'est pas obligatoire mais si elle existe alors le mot-clé **else** doit être sur la même ligne que l'accolade fermante du si. On peut imbriquer plusieurs boucles si alors sinon si alors sinon etc.

7.1.3 Vectorisation

Comme on l'a déjà vu, un gros avantage de R est sa capacité a travailler vectoriellement ce qui évite d'avoir à programmer des boucles for et/ou des tests. Ainsi, la boucle for du § 9.1.1 peut être remplacée simplement par

```
> x <- 1:10
> y <- x^2
```

tandis que la boucle suivante : ^

```
> n <- 10
> x <- rpois(n, 20)
> pair <- rep(FALSE, n)
> for (i in 1:n) {
+ if (x[i]%%2 == 0)
+ pair[i] <- TRUE
+ }
```

peut s'écrire simplement pair[x%%2==0]<-TRUE. Plus généralement, toute fonction (de R ou créée par l'utilisateur) peut être appliquée à tous les éléments d'une matrice (ou seulement à certaines lignes ou colonnes), d'un vecteur ou d'une liste. Pour cela on utilise les fonctions apply, sapply ou lapply (on en verra un exemple au paragraphe suivant).

7.1.4 Écrire une fonction

L'essentiel du travail sous R se fait à l'aide de fonctions dont les arguments sont indiqués entre parenthèses. Outre les fonctions déjà implémentées dans R (on en a déjà vu un certain nombre), l'utilisateur peut écrire ses propres fonctions (c'est même d'ailleurs conseillé!) afin de pouvoir efficacement répéter certaines opérations. Voici par exemple l'écriture de la fonction compte qui prend deux arguments, lettre (un caractère) et seq (un vecteur de caractères), et retourne le nombre d'occurrences de lettre dans seq :

```
> compte <- function(lettre, seq) {
+ seqbinaire <- rep(0, length(seq))
+ seqbinaire[seq == lettre] <- 1
+ nb <- sum(seqbinaire)
+ nb
+ }
```

Si l'on veut que la fonction retourne le contenu d'un objet (ici la valeur de nb), ce dernier doit figurer seul sur la dernière ligne de la fonction. Pour pouvoir exécuter cette fonction, il faut la charger en mémoire, soit en tapant chacune des lignes au clavier soit en l'écrivant dans un fichier et en "compilant" ce dernier avec la fonction source, comme pour un programme (paragraphe suivant). Dés que la fonction est chargée, on l'utilise naturellement

```
> sequence <- c("a", "a", "t", "g", "a", "g", "c", "t", "a", "g",
+ "c", "t", "g")
> compte("a", sequence)
 [1] 4
```

Dans cet appel `a la fonction compte, le nom des arguments (lettre et seq) n'a pas été utilisé ce qui implique que les objets ("a" et sequence) doivent être passés dans l'ordre. Sinon on peut écrire compte(seq=sequence,lettre="a").
Il est aussi possible d'attribuer des valeurs par défaut aux arguments d'une fonction; par exemple compte<-function(lettre="a",sequence){ ...}.
Si l'on veut obtenir la composition de sequence en (a,c,g,t), on pourra faire :

```
> bases <- c("a", "c", "g", "t")
> composition <- rep(0, 4)
> for (i in 1:4) composition[i] <- compte(bases[i], sequence)
> composition
[1] 4 2 4 3
```

ou "appliquer" la fonction compte au vecteur bases :

```
> sapply(bases, FUN = compte, seq = sequence)
a c g t
4 2 4 3
```

Les variables définies dans les fonctions (nb et seqbinaire dans la fonction compte par exemple) sont considérées comme des variables locales. Si ces variables existaient déjà dans l'environnement "extérieur" à la fonction alors leur contenu n'est pas modifié. Si elles n'existaient pas, alors elles n'existeront pas plus après l'appel à la fonction. Par contre, si la fonction utilise une variable non définie dans la fonction, alors R va la chercher dans l'environnement extérieur ; si elle existe, R utilise sa valeur, sinon un message d'erreur est donné. Le code des fonctions créées et chargées en mémoire s'obtient simplement en tapant le nom de la fonction suivi de "entrée".

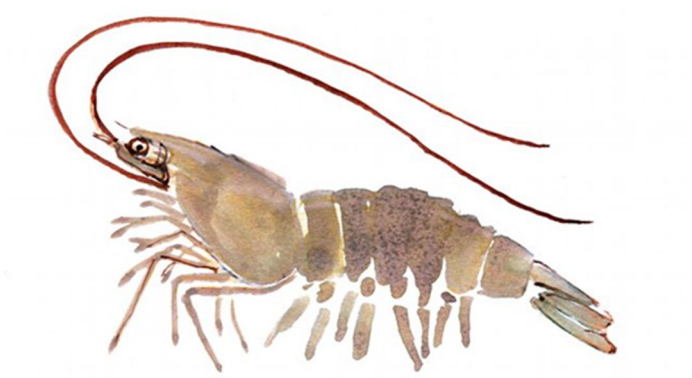
7.1.5 Écrire un programme

On écrira typiquement un programme R dans un fichier texte ayant l'extension '.r', par exemple myprog.r. Si le programme définit des fonctions et les utilise, il convient de les définir avant de les utiliser (par exemple en tête du fichier, éventuellement après l'assignation des variables globales). Le caractère '#' sert à ajouter des commentaires : en effet, R passe à la ligne suivante. Pour exécuter le programme contenu dans le fichier myprog.r, il suffit de taper :

```
> source("myprog.r")
```

CHAPITRE 8

LES OUTILS GRAPHIQUES



R propose de nombreux outils graphiques pour l'analyse et la visualisation des données. Des fonctions graphiques sont fournies par les packages graphics, grid et lattice.
On peut avoir un aperçu des possibilités graphiques grâce à la commande demo(graphics).
Les graphiques sont dessinés sur un périphérique graphique. Au démarrage, R lance automatiquement un périphérique graphique en fonction du système d'exploitation. Ceci peut se faire manuellement. Les commandes sont x11() pour UNIX, windows() pour Windows, quartz() pour MAC OS X.

Les commandes graphiques sont reparties en trois groupes :
Fonctions de **haut niveau** : Création de nouvelles graphiques sur le périphérique graphique avec des axes, des étiquettes, des titres et ainsi de suite.
Fonctions de **bas niveau** : Ajout d'informations à un graphique existant, tels que des points supplémentaires, des lignes et des étiquettes.
Des fonctions graphiques **interactives** : Permettant d'ajouter des informations de manière interactive, ou extraire des informations d'un tracé existant, en utilisant la souris.

8.1 Gestion des fenêtres graphiques

Lorsqu'une fonction graphique est exécutée, R ouvre une fenêtre graphique et y affiche le graphe. On peut spécifier le dispositif de gestion des fenêtres. La liste des dispositifs graphiques disponibles dépend du système d'exploitation. Sous Mac, pour créer une nouvelle fenêtre graphique, on utilise la commande get("quartz")(). Sous PC, pour créer une nouvelle fenêtre graphique, on utilise la commande X11(). Tant sur Mac que sur PC, pour sélectionner une des fenêtres, on utilise la commande dev.set(). Par exemple, pour sélectionner la fenêtre graphique numéro *i*, on tape la commande dev.set(i). Si on souhaite connaître le numéro de la fenêtre active, on tape la commande dev.cur().

8.1.1 R propose deux manières de partitionner les graphiques

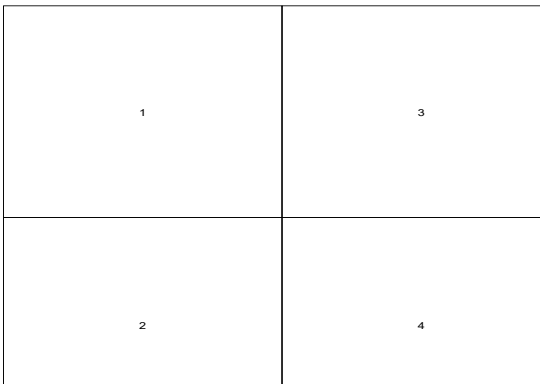
a) La commande split.screen(c(1, 2)) divise le graphique en deux parties qu'on sélectionne avec les commandes screen(1) et screen(2).
b) La fonction layout() partitionne le graphique actif en plusieurs parties sur lesquelles sont affichés les graphes successivement. Cette fonction a pour argument une matrice de valeurs entières qui indiquent le numéro des sous-fenêtres. Pour visualiser la partition crée, on

utilise la fonction layout.show avec, en argument, le nombre de sous-fenêtres. Par exemple, si on veut diviser la fenêtre en quatre parties égales, on tape la commande suivante :

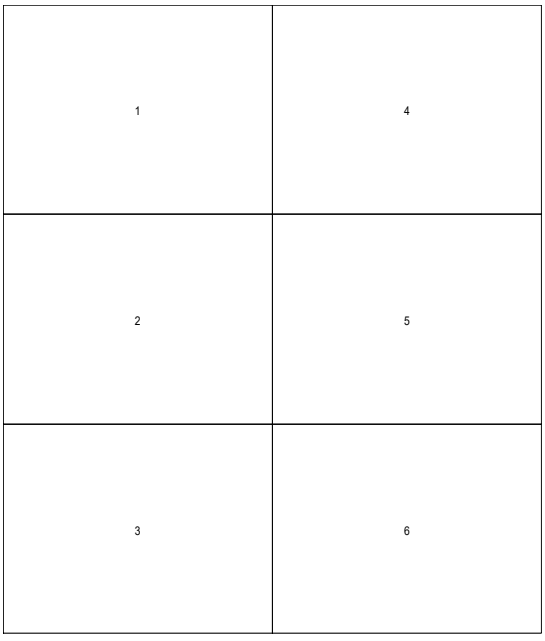
>layout(matrix(1:4, 2, 2)) ;

et pour visualiser la partition créée, on utilise la commande layout.show(4).

```
>layout(matrix(1 :4, 2, 2))
>layout.show(4)
```




```
>layout(matrix(1 :6, 3, 2))
>layout.show(6)
```



8.2 Principales fonctions graphiques du package graphics

Liste des principales fonctions graphiques du package graphics (extrait de R pour les débutants d'Emmanuel Paradis)

Liste des principales fonctions graphiques du package graphics (extrait de R pour les débutants d'Emmanuel Paradis) Les fonctions qui figurent en gris et italique ne sont pas traitées dans ce document

hist(x) Histogramme des fréquences de x

barplot(x) Histogramme des valeurs de x

mosaicplot(x) Si x est une matrice ou un data.frame, graphe en mosaïque des résidus d'une régression log-linéaire sur une table de contingence

pie(x) Graphe en « camembert »

boxplot(x) Graphe « boites à moustaches »

plot(x) Graphe des valeurs de x (sur l'axe des y) ordonnées sur l'axe des x

plot(x, y) Graphe bivarié de x (sur l'axe des x) et y (sur l'axe des y). Si x et y sont des vecteurs, plot(x , y) produit un nuage de points de y contre x. Le même effet peut être produit en fournissant un seul argument (seconde forme) sous forme d'une liste contenant deux éléments x et y ou d'une matrice à deux colonnes.

pairs(x) Dessine tous les graphes bivariés entre les colonnes de x. Si x est une matrice numérique ou une data frame, la commande pairs(x) produit une matrice de graphique où chaque colonne de x est représentée en fonction de toutes les autres colonnes x.

symbols(x, y, ...) Dessine aux coordonnées données par x et y des symboles (cercles, carrés, rectangles, étoiles, thermomètres ou "boxplots") dont les tailles, couleurs, … sont spécifiées par des arguments supplémentaires

sunflowerplot(x, y) Idem que plot() mais les points superposées sont dessinées sous forme de fleurs dont le nombre de pétales représente le nombre de points

coplot(x˜y | z) Graphe bivarié de x et y pour chaque valeur ou intervalle de valeurs de z

matplot(x,y) Graphe bivarié de la 1ère colonne de x contre la 1ère de y, la 2ème de x contre la 2ème de y, etc.

stars(x) Si x est une matrice ou un data.frame, dessine un graphe en segments ou en étoile où chaque ligne de x est représentée par une étoile et les colonnes par les longueurs des branches

assocplot(x) Graphe de Cohen–Friendly indiquant les déviations de l'hypothèse d'indépendance des lignes et des colonnes dans un tableau de contingence à deux dimensions

fourfoldplot(x) Visualise, avec des quarts de cercles, l'association entre deux variables dichotomiques pour différentes populations (x doit être un array avec dim = c(2, 2, k) ou une matrice avec dim = c(2, 2) si k = 1)

qqnorm(x) Quantiles de x en fonction des valeurs attendues selon une loi normale

qqplot(x, y) Quantiles de y en fonction des quantiles de x

interaction.plot(f1, f2, y) Dans le cadre d'une analyse de la variance, si f1 et f2 sont des facteurs, graphe des moyennes de y (sur l'axe des y) en fonction des valeurs de f1 (sur l'axe des x) et de f2 (différentes courbes) ; l'option fun permet de choisir la statistique résumée de y (par défaut fun = mean)

8.2.1 Il y a un certain nombre d'arguments qui peuvent être passés à des fonctions graphiques de haut niveau, comme suit :

add=TRUE Oblige la fonction d'agir comme une fonction graphique de bas niveau, superposant le graphique sur le graphique courant (accepté par certaines fonctions seulement).

axes=FALSE Supprime la génération des axes-utile pour ajouter vos propres axes personnalisés avec la fonction axis(). La valeur par défaut, axes=TRUE, ajoute les axes.
log="x"
log="y"
log="xy"
Met les axes X, Y ou les deux à la fois à l'échelle logarithmique. Cela fonctionne pour beaucoup, mais pas pour tous les types de graphiques.

type= Le type= est un argument de contrôlant le type de graphique produit, comme suit:

type="p" trace des points (par défauts)
type="l" trace des lignes
type="b" Trace des points reliés par des lignes (les deux)
type="o" Trace des points recouverts par les lignes
type="h" Lignes verticales à partir des points à l'axe zéro
type="s" et type="S" Graphique en escalier.

type="n" Aucun point n'est dessiné. Cependant les axes sont toujours dessinés (par défaut) et le système de coordonnées est mis en place en fonction des données. Idéal pour la création des graphiques avec la suite des fonctions graphiques de bas niveau.

xlab= "titre de l'axe des x"
ylab= "titre de l'axe des y"

Titre des axes x et y. Utilisez ces arguments pour changer les étiquettes par défaut, généralement le nom des objets utilisés dans l'appel de la fonction graphique.

main= "titre du graphique"
Titre de la figure, placée en haut du graphique dans une grande police.

sub= "Sous-titre"
Sous-titre, placé juste en dessous de l'axe des x dans une police plus petite.

Parfois les commandes graphiques de haut niveau ne produisent pas exactement le type de graphique désiré.
Dans ce cas, les commandes graphiques de bas niveau peuvent être utilisées pour ajouter des informations supplémentaires (comme des points, des lignes ou du texte) au graphique en cours.

8.2.2 Quelques-unes des fonctions de bas niveau les plus utiles sont les suivantes :

points(x, y)
lines(x, y)
Ajoute des points ou des lignes connectées au graphique en cours. L'argument type= de plot() peut être aussi passé à ces fonctions (par défaut "p" pour points() et "l" pour lines()).

text(x, y, labels, ...) Ajoute du texte au graphique au niveau des points de coordonnées données par x, y . Normalement *labels* est un vecteur de nombre entier ou un vecteur de caractères auquel cas labels *est tracé au point de coordonnées (x[i], y[i]).*
La valeur par défaut est 1 : length(x).

```
plot (x, y, type = "n"); text (x, y, names)
```

pour les points.

abline(h= y)
abline(v= x)

Ajoute une ligne.

h= y Ajoute une ligne horizontale tout au long du graphique en partant du point d'ordonnée y.

v= x ajoute une ligne verticale tout au long du graphique en partant du point d'abscisse x (à spécifier).

legend(x, y, legend, ...) Ajoute une légende au graphique actuel à la position spécifiée.

title(main, sub) Ajoute un titre *main* en haut du graphique en cours dans une police de grande taille et (éventuellement) un sous-titre *sub*, en bas du graphique, dans une police plus petite.

axis(side, ...) Ajoute un axe au graphique actuel sur le côté donné par le premier argument (1 à 4, en comptant dans le sens horaire en partant du bas).

Dans certains cas, il est utile d'ajouter des symboles et des formules mathématiques à un graphique. Ceci peut être réalisé dans R en spécifiant une expression plutôt qu'une chaîne de caractères dans l'une des fonctions quelconques text , mtext , axis , ou title. Par exemple, le code suivant dessine la formule pour la fonction de probabilité binomiale :

```
text(x, y, expression(paste(bgroup("(", atop(n, x), ")"), p^x, q^{n-x})))
```

Plus d'informations peuvent être obtenues en utilisant les commandes R

```
help(plotmath)
example(plotmath)
demo(plotmath)
```

R fournit également des fonctions qui permettent aux utilisateurs d'extraire ou d'ajouter des informations sur un graphique à l'aide d'une souris. La plus simple d'entre elles est la fonction locator() :

locator(n,type)
Attend que l'utilisateur sélectionne des points sur le graphique actuel en utilisant le bouton gauche de la souris. Cela continue jusqu'à ce n (par défaut 512) points soient sélectionnés, ou un autre bouton de la souris soit appuyé. L'argument *type* permet de tracer au niveau des points sélectionnés. locator() retourne les emplacements des points choisis sous forme de liste avec deux composants x et y .

locator() est généralement appelé sans aucun argument. Il est particulièrement utile pour la sélection des positions, de manière interactive, pour les éléments graphiques tels que des

légendes ou des étiquettes. Par exemple, pour placer du texte informatif à proximité d'un point, la commande :

```
text(locator(1), "Outlier", adj=0)
```

Pourrait être utile.

identify(x, y, labels) Permet à l'utilisateur de mettre en évidence des points définis par x et y (en utilisant le bouton gauche de la souris) en mettant l'étiquette correspondante (*labels*) à proximité (ou le numéro d'indice du point, si *labels* est absent). Retourne les indices des points sélectionnés quand un autre bouton est appuyé.

8.2.3 Paramètres graphiques permettent de personnaliser le graphique généré par R

Pour changer de manière permanente les paramètres graphiques, on utilise la fonction par().

par() Sans arguments, retourne la liste de tous les paramètres graphiques et de leurs valeurs pour le dispositif actuel.

par(c("col", "lty")) Avec un vecteur de caractères en argument, renvoie uniquement les paramètres graphiques indiqués.

par(col=4, lty=2) Avec des arguments indiqués, définit les valeurs des paramètres graphiques indiqués.

Pour pouvoir restaurer les paramètres graphiques par défaut après modification, il faut penser à enregistrer le résultat de par() lors des modifications. On peut alors les restaurer après.

```
oldpar <- par(col=4, lty=2)
 #... plotting commands ...# par(oldpar)
```

Pour enregistrer et restaurer tous les paramètres graphiques :

```
oldpar <- par(no.readonly=TRUE)
#... plotting commands ...# par(oldpar)
```

8.3 Diagramme en camembert avec légende, valeurs et étiquettes superposées

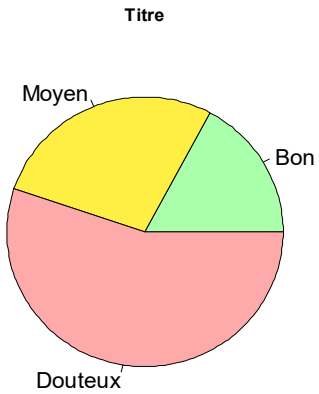
L'exemple ci-dessous permet de faire des diagrammes en camembert (diagramme en secteurs avec le logiciel R) en ajoutant une légende détaillée dans un cadre, en précisant le nom de chaque quartier et la valeur correspondante en pourcentages.

8.3.1 Exemple de diagramme en camembert légendé en détails

```
>valeurs <- c(125,200,400)
```

a) Tracer le diagramme en camembert

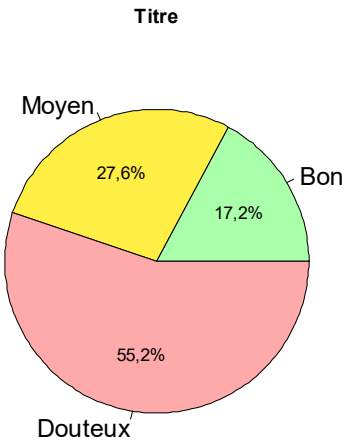
```
>pie(valeurs,col=c("#AAFFAA","#FFEE44","#FFAAAA"),labels=c("Bon","Moyen","Douteux"),main="Titre",cex=1.5)
# labels : ces étiquettes sont une option, dans la mesure où on peut se contenter de la légende
# cex : taille de la police, par défaut ce paramètre est de 1, ici 1.5
```



b) Ajouter les valeurs en % en tant qu'étiquettes au centre

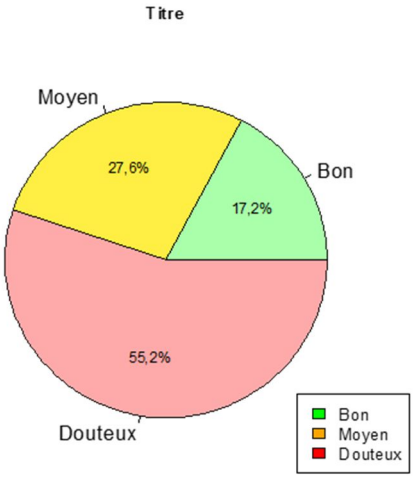
```
# Calculer pourcentages correspondant à chaque valeurs
>total = sum(valeurs)
>pourcentages = valeurs/total*100 ; cat("Les valeurs en % sont de :",pourcentages,"\n")
# Fonction à coller dans R - cette fonction text_pie permet d'ajouter des étiquettes au centre des quartiers
>text_pie = function(vector,labels=c(),cex=1) {vector = vector/sum(vector)*2*pi temp = c()j = 0l = 0for (i in 1:length(vector)) {k = vector[i]/2 j =  j+l+k  l = k
text(cos(j)/2,sin(j)/2,labels[i],cex=cex)
+}
+vector = temp
+}

# Ajouter les étiquettes
```

```
>text_pie(pourcentages,c("17,2%","27,6%","55,2%"),cex=1.1)
# Ces valeurs en % sont à remplacer manuellement
```



c) Ajouter la légende

```
>legend(x="bottomright", legend=c("Bon","Moyen","Douteux"),
cex=1.2,fill=c("green","orange","red"))

#x : position du cadre de légende (ici en bas à droite)
#cex : taille de la police
```



8.4 Tracer un diagramme en barres ou diagramme en bâtons

- Tracer un diagramme en barres
- Annoter un diagramme en barres (traits, légende)
- Ajouter des barres d'erreur à un diagramme en barres (étoiles de confiance)
- Changer le type de diagrammes en barres (barres cumulées, horizontales, multiples axes...)

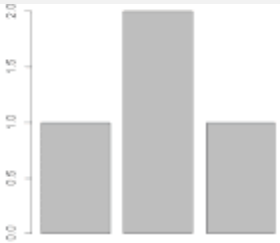
L'essentiel !

Tracer un diagramme sous R se fait avec la commande barplot().
Taper ?barplot dans la console R pour obtenir de l'aide directement à partir de R.
Certains paramètres de barplot sont incontournables tels col (couleur), space (espace entre les barres = largeur des barres), border (bordures des barres), names.arg (noms des barres) et horiz (bâtons horizontaux ou verticaux)...

- Tracer un diagramme en barres

Réaliser un diagramme en barres avec le logiciel R à partir d'une liste de valeurs x

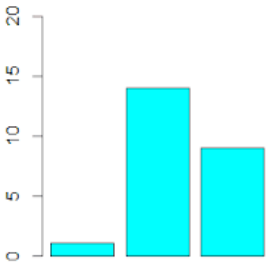
```
>x <- c (1, 2,1)                    # x est donné à titre d'exemple
>barplot (x)
```



Fonction barplot

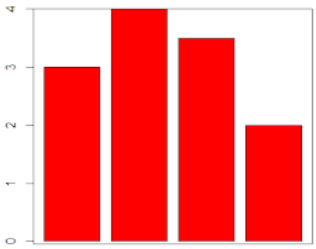
- Changer les limitations de l'axe des ordonnées

```
>x = c (1, 14, 9)                   # x est donné à titre d'exemple
>barplot (x, ylim=c(0,20),col="cyan")
```



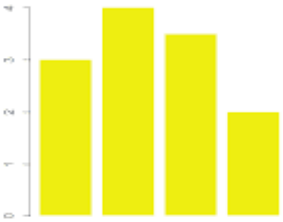
- Encadrer le diagramme en barres

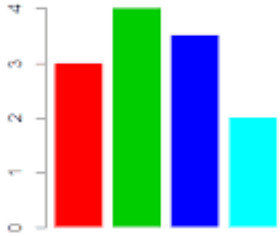
```
>x = c (3,4,3.5,2)
>barplot (x, col="red")
>box ()                             # box() permet d'encadrer le diagramme
```



- Changer la mise en forme : couleur, encadrement des barres

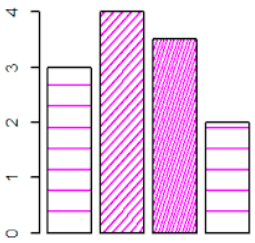
```
>x = c (3, 4, 3.5, 2)
>barplot(x,col="#EEEE11",border=NA)
# col pour la couleur, border = NA si on ne veut pas de bordure sur les bâtons - ici la couleur #EEEE11 est donnée avec précision en hexadécimal
# Cliquer sur ce lien pour plus de couleurs hexadécimales
# Couleurs préprogrammés
>barplot(x,col=c(1,2,3),border=NA)
```





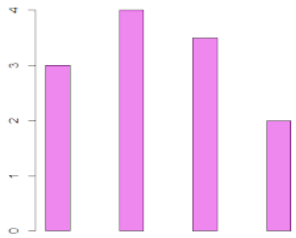
- Changer l'épaisseur du trait de bordures des barres et/ou hachurer les barres

```
>x = c(3,4,3.5,2)
>par(lwd=2)        # doit précéder la fonction barplot pour changer l'épaisseur des traits du diagramme
>barplot(x, density=c(5,15,30),angle=c(0,45,70),col=6)
# density = nombres de traits par unité de surface (densité)
# angle = angle des traits
```



- Changer l'espacement entre les barres

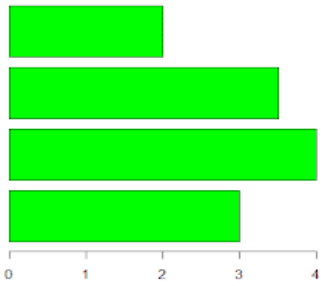
```
>x = c(3,4,3.5,2)
>barplot(x,col="#EE88EE",space=2)    # space permet de jouer sur l'espacement
```



- Changer le type de diagramme en barres : barres horizontales ou barres verticales

exemple :

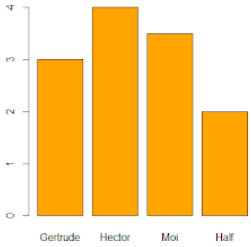
```
>x = c(3,4,3.5,2) ; barplot(x,  horiz = T,col="green")
```



- Changer le nom des barres

exemple :

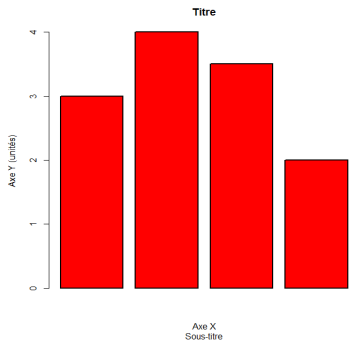
```
>x <- c(3,4,3.5,2)
>noms_barres <- c("Gertrude","Hector","Moi","Half")
>barplot(x,col="orange", names.arg=noms_barres)
```



- Titre du diagramme, sous-titre et titres des axes

exemple :

```
>x <- c(3,4,3.5,2)
>barplot(x,col="red",xlab="Axe X",ylab="Axe Y (unités)",main="Titre",sub="Sous-titre")
```



- **Exemple :** un diagramme en barres à barres horizontales présentant sur un axe le nombre d'habitants et sur l'autre, les pourcentages correspondants.

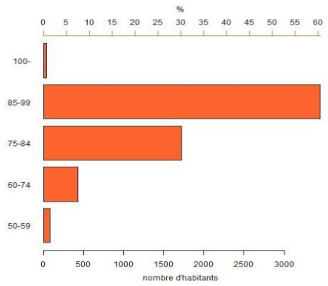


Diagramme à barres horizontales

a. Entrer les données : nombre d'habitants par catégorie

```
>ages = c("<39","40-49","50-59","60-74","75-84","85-99","100-")
>total = c(14,20,76,428,1717,3453,29)
```

b. Tracer le diagramme en barres

```
>barplot(total[3:7],horiz=T,names=ages[3:7], las=1,col="#2D69B3")
>mtext(side=1, "nombre d'habitants",line=2.5)
# side = 1 ; indique qu'il faut tracer l'axe du bas
# line : indique la distance entre le titre de l'axe et l'axe
```

c. Calculer les pourcentages correspondants aux données tracées

```
>somme = sum(total[3:7])
>total_s = c(14,20,76,428,1717,3453,29)/somme*100 ; total_s
```

d. Ajouter la légende

```
>par(new=TRUE)
>barplot(total_s[3:7],axes=F,horiz=T,plot=T,col="#FE642E")
>axis(3,pretty(total_s,10),col="#F5821F")
>mtext(side=3, "%",line=2.5)
```

8.5 Histogrammes

Histogrammes 2D et 3D avec R

Histogrammes annotés par des verticales, horizontales, point précisé ou courbe de densité

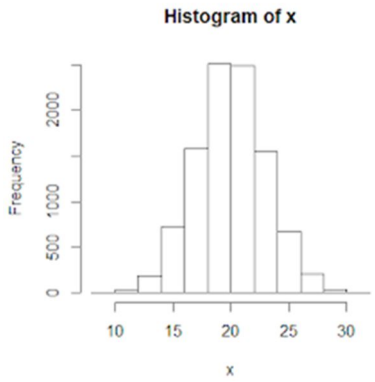
L'essentiel !

Tracer un histogramme avec R, c'est à dire visualiser la répartition d'un effectif se fait avec la commande hist().

On peut aussi réaliser des histogrammes 3D avec la commande hist3D du package plot3D mais il s'agit de diagrammes en barres. Le calcul de la répartition devra être fait par soi-même.

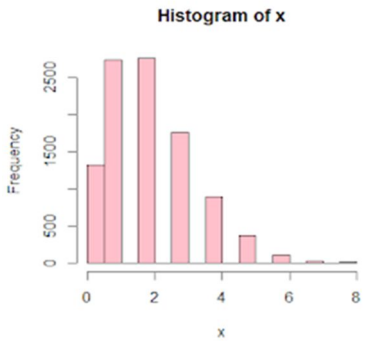
▪ Afficher un histogramme

```
# soit x une liste de données simulées ici
>x <- rnorm(10000,20,3)     # A remplacer
# Tracer l'histogramme
>hist(x)
```



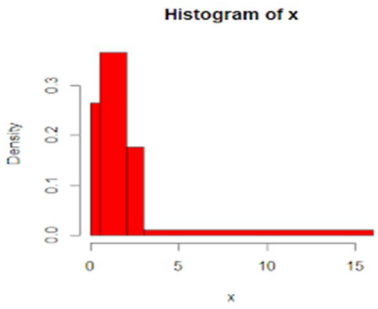
- Fixer le nombre de barres de l'histogramme
  - breaks : argument définissant le nombre de barres

```
>hist(x, breaks=20)
```



- Imposer la découpe de l'histogramme : la fréquence des occurrences de chaque donnée est imposée par catégorie

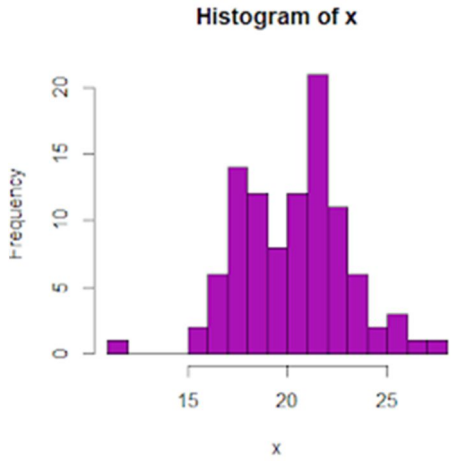
```
# soit x une liste de données simulées ici
>x <- rpois(10000,2)
# ex : 3 classe de 0 à 1, 1 à 2, 2 à 4, 4 à 16 )
>hist(x, br = c(0,0.5,2,3,16),col="red")
```



8.5.1 Mettre en forme un histogramme

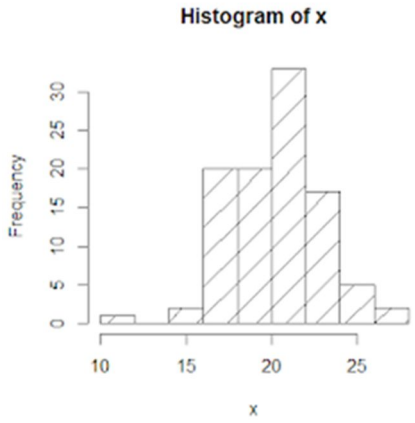
- Changer la couleur des barres
- col : argument codant pour la couleur

```
>hist(x, breaks=20, col="red") # couleurs par leur nom
#ou
>hist(x, breaks=20, col=2) # couleurs par leur code R
#ou
>hist(x, breaks=20, col="#AA12B6") # code hexadécimal
 #  Autres couleurs
```



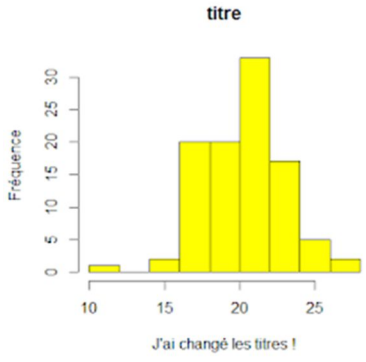
- Changer le remplissage des barres (hachurage)
- density : densité des hachures
- si density = 0, les barres sont vides

```
>hist(x, density=5)
```



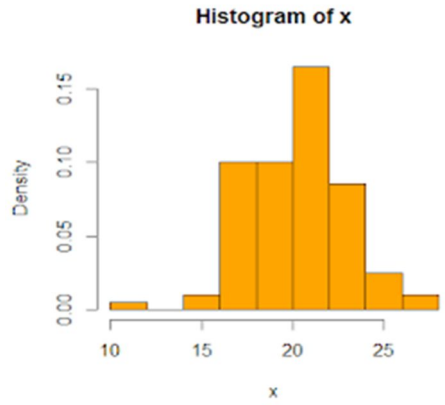
- Tracer l'histogramme avec titre principal et titres aux axes
- main : titre principal
- xlab et ylab : titre des 2 axes

```
>hist(x, main="titre", xlab="Types",ylab="Fréquence")
```



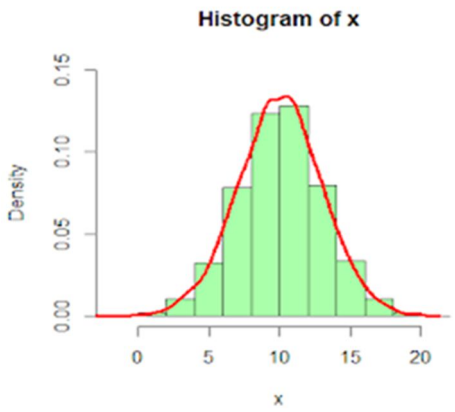
- Afficher la fréquence des occurrences ou le nombre d'occurrences en ordonnées

```
# Affiche   le nombre d'occurrences
>hist(x, freq = T)    # ou hist(x, freq=TRUE)
# Affiche  la fréquence des occurrences
>hist(x, freq = F)    # ou hist(x, freq=FALSE)
```



- Superposer une courbe de lissage traduisant la densité par occurrence (cf. estimation par noyau)

```
x <- rnorm(10000,10,3 ) # x liste de valeurs simulées  à remplacer par les votres
>hist(x,freq=F,col="#AAFFAA",ylim=c(0,0.15)) # il faut que freq=F
>densite <- density(x) # créer une liste des densités
>lines(densite, col = "red",lwd=3) # Superposer la ligne
```



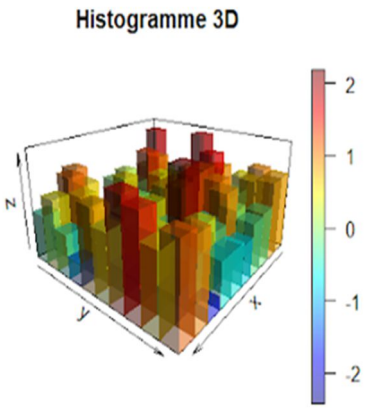
8.5.2 Tracer un histogramme en 3D (histogramme en trois dimensions)

Voici un exemple qui permet de tracer un histogramme 3D. Nous avons un jeu de données de 10 sur 10 dont on a la valeur de chaque barre de coordonnées (x, y)

Attention, ce n'est pas réellement un histogramme 3D, ce serait plutôt un diagramme en barres 3D. Faire réellement un histogramme 3D ici requiert de calculer soi-même la valeur de chaque barre.

```
>install.packages("plot3D")
>library("plot3D")
>ma_matrice <- matrix(rnorm(100),10,10) ; ma_matrice
```

```
>hist3D(z=ma_matrice, expand=.5, alpha=.5, shade=.75, theta=130, phi=20, cex=2,
>main="Histogramme 3D")
```



8.6 Comment faire une boîte à moustache avec R

Vous voulez représenter vos données avec la boîte à moustache de Mr Tukey (boxplot) ? Rien de plus facile avec R.

```
#jeu de données fictif pour exemple
>a<-c(1,1,1,5,5,5,5,6,6,8,8,20,30)
>b<-c(0.5,4,5,6,6,6,6,6,7,7,7,7,8)
```

```
#traçons les boxplots de base avec la fonction boxplot
>boxplot(a)
>boxplot(b)
```

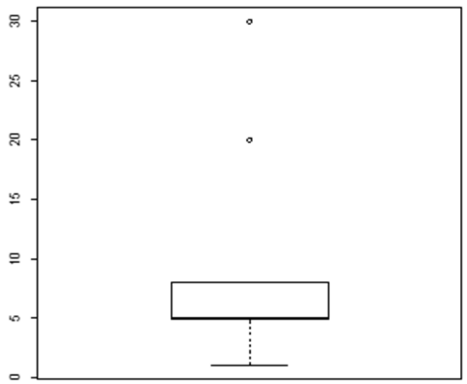


Figure a

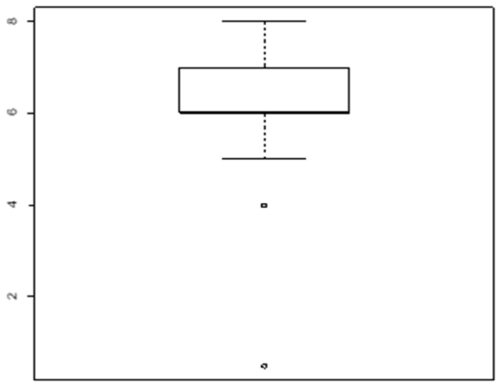


Figure b

```
#on enlève les outliers, en mettant outline=FALSE
>boxplot(a,outline=FALSE)
>boxplot(b,outline=FALSE)
```

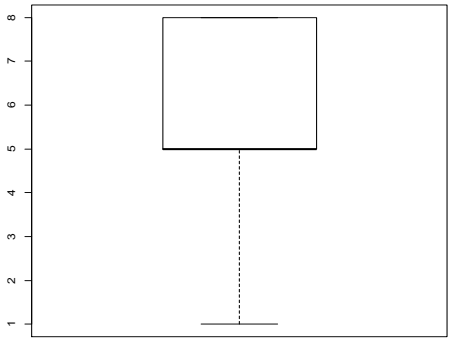


Figure a

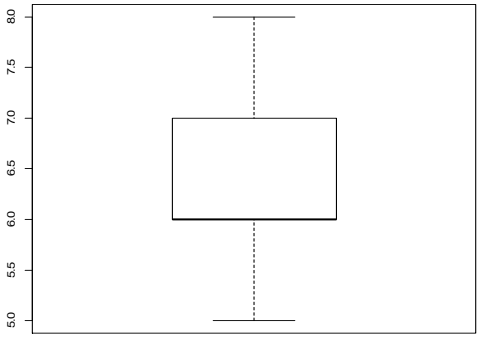


Figure b

```
#pour les mettre à l'horizontal
>boxplot(a,horizontal=TRUE)
>boxplot(b,horizontal=TRUE)
```

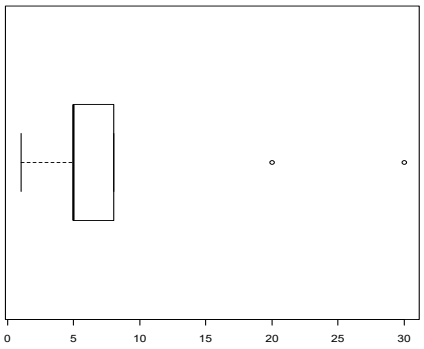


Figure a

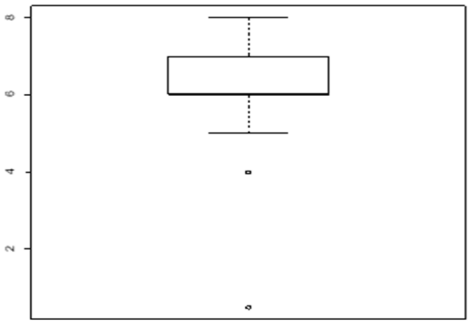


Figure b

```
#changer de couleur
>boxplot(a,border="blue")
>boxplot(b,border="purple")
```

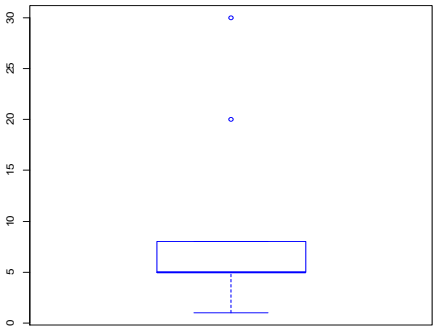


Figure a

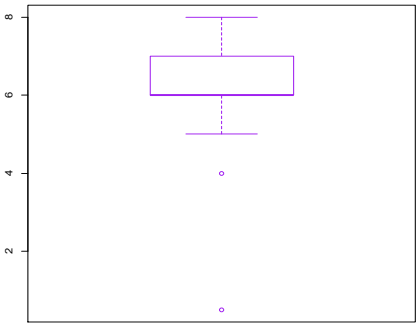


Figure b

```
#nouveau jeu de données plus complexe
>n<-c(1,1,1,5,5,5,5,6,6,8,8,20,30,0.5,4,5,6,6,6,6,6,7,7,7,7,8,3,5,8,8,8,8,8,9,9,9,9,11,12)
>m<-c(rep('A',13),rep('B',13),rep('C',13))
```

```
data<-data.frame(N=n,M=m)
#on visualise le tableau ainsi créé
>data
```

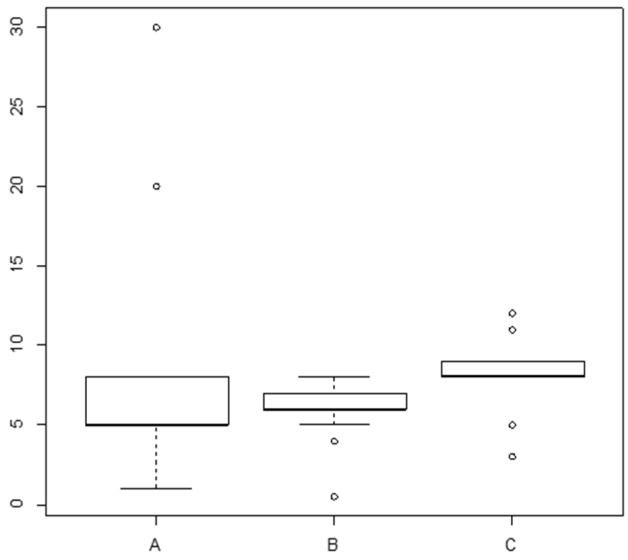
	N	M
1	1.0	A
2	1.0	A
3	1.0	A
4	5.0	A
5	5.0	A
6	5.0	A
7	5.0	A
8	6.0	A
9	6.0	A
10	8.0	A
11	8.0	A
12	20.0	A
13	30.0	A
14	0.5	B
15	4.0	B
16	5.0	B
17	6.0	B
18	6.0	B
19	6.0	B
20	6.0	B
21	6.0	B
22	7.0	B
23	7.0	B
24	7.0	B
25	7.0	B
26	8.0	B
27	3.0	C
28	5.0	C
29	8.0	C
30	8.0	C
31	8.0	C
32	8.0	C

```
33   8.0  C
34   9.0  C
35   9.0  C
36   9.0  C
37   9.0  C
38  11.0  C
39  12.0  C
```

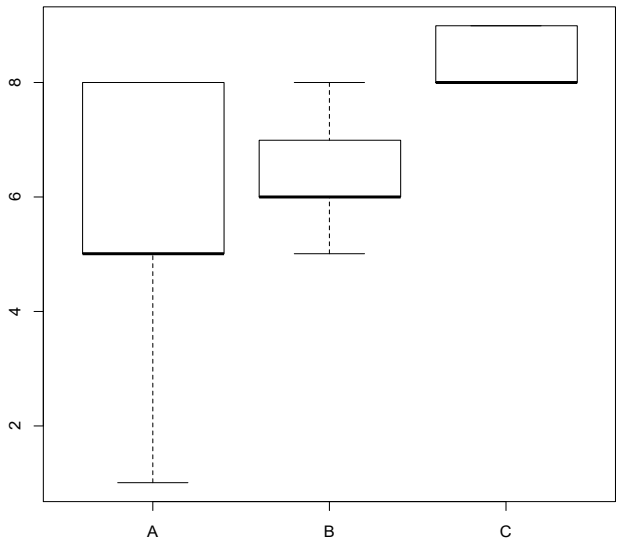
```
>summary(data)
```

```
       N          M
Min.   :  0.500   A: 13
1st Qu.:  5.000   B: 13
Median :  7.000   C: 13
Mean   :  7.269
3rd Qu.:  8.000
Max.   : 30.000
```

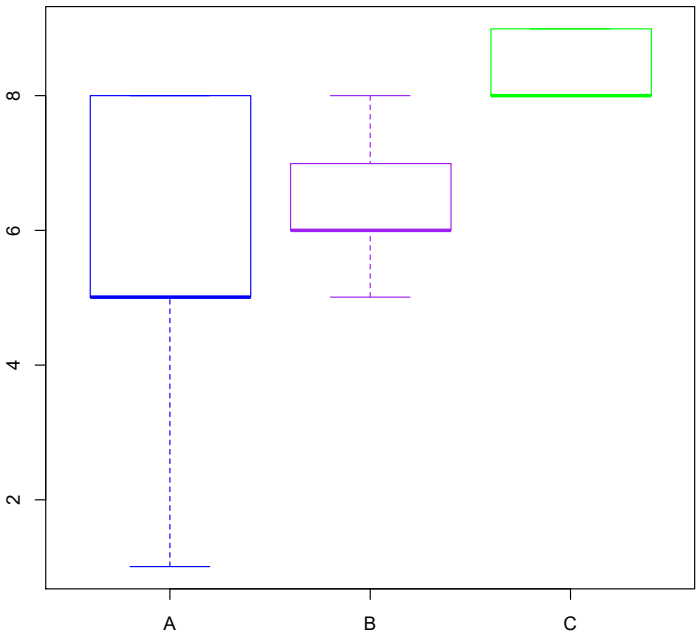
```
#On a 13 mesures pour chaque modalité (A,B,C)
#comment avoir les boxplots pour chaque modalité?
>boxplot(data$N~data$M)
```



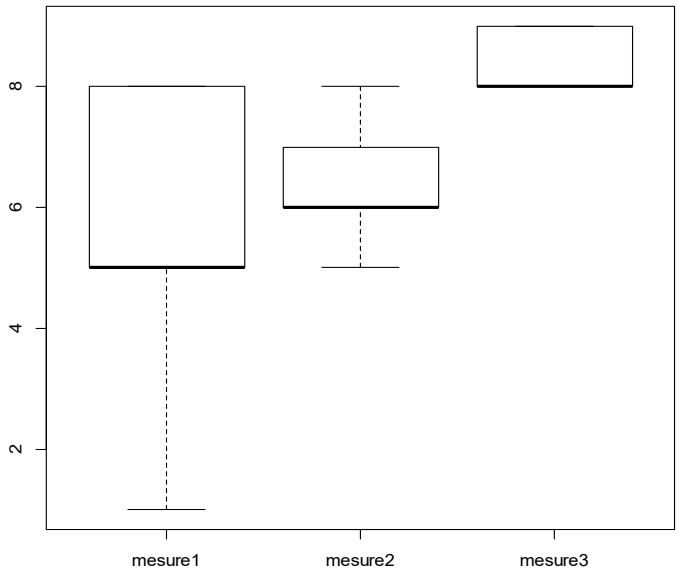

```
#on enlève les outliers
>boxplot(data$N~data$M,outline=FALSE)
```



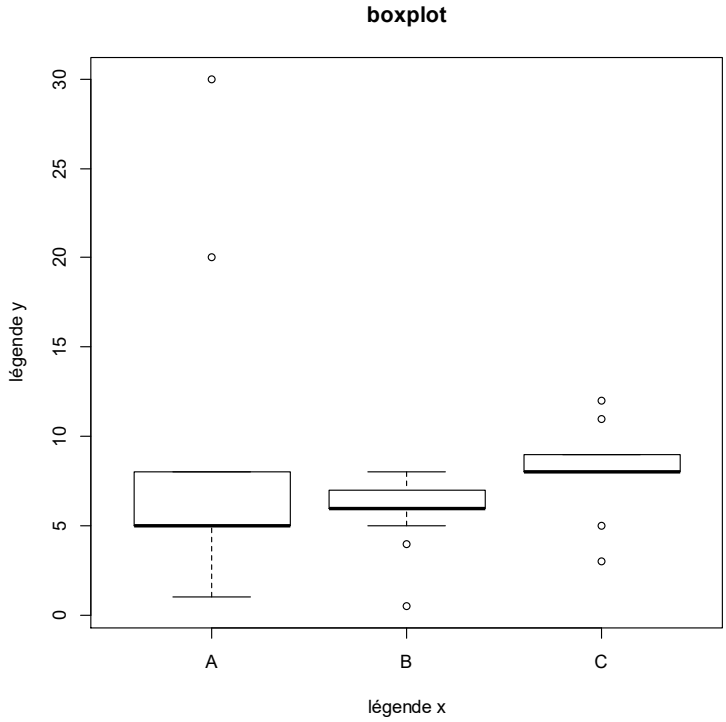

```
#on change les couleurs avec l'argument border
>boxplot(data$N~data$M,outline=FALSE,border=c("blue","purple","green"))
```



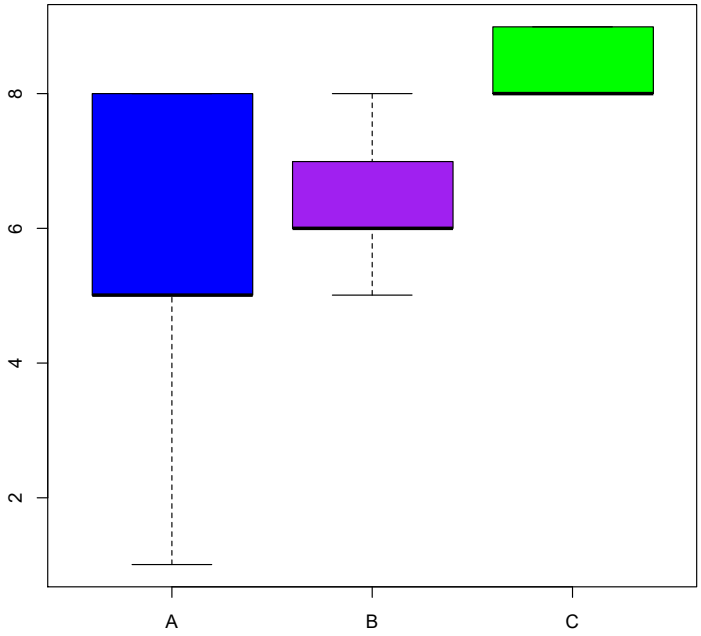

```
#on change les noms avec names: A devient mesure1, B mesure2, C mesure3
>boxplot(data$N~data$M,outline=FALSE, names=c("mesure1","mesure2","mesure3"))
```



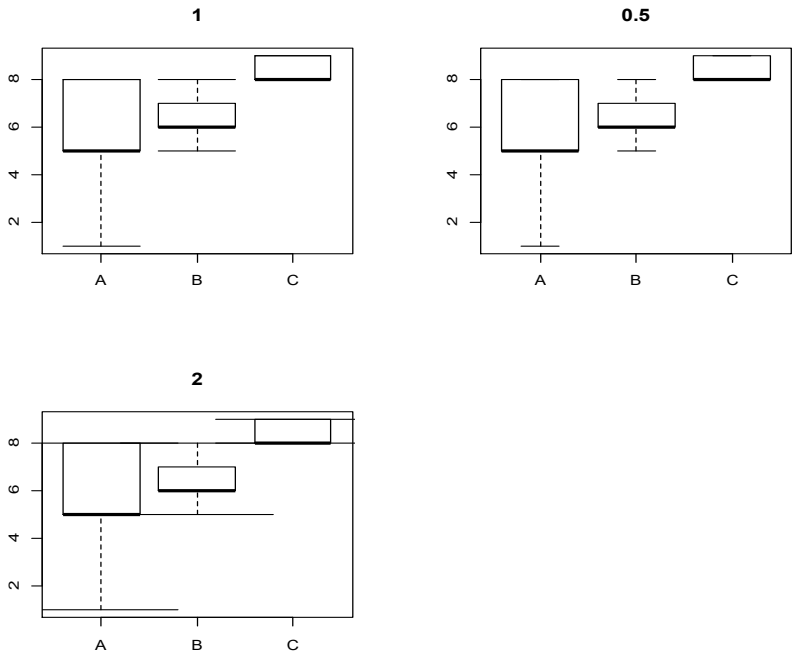

```
#on ajoute les légendes
>boxplot(data$N~data$M,xlab="légende x",ylab="légende y",main="boxplot")
```



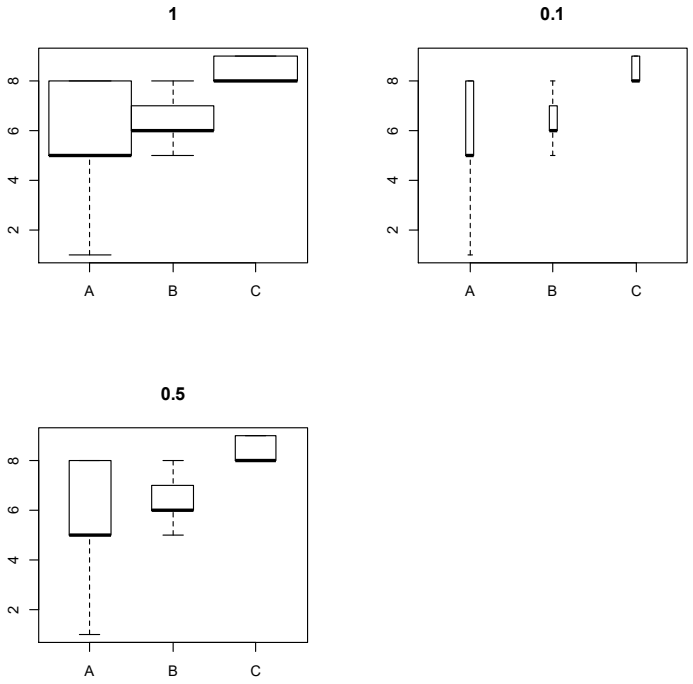

```
#on colore les boîtes avec l'argument col
>boxplot(data$N~data$M,outline=FALSE,col=c("blue","purple","green"))
```



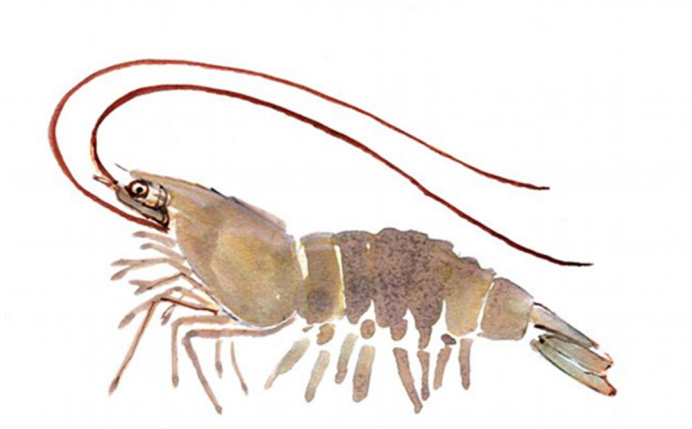

```
#on change la largeur des moustache avec staplewex
#x11() ouvre une autre fenêtre graphique pour éviter d'écraser le graphe précédent,
<x11()
>par(mfrow=c(2,2))
>boxplot(data$N~data$M,staplewex=1,outline=FALSE,main="1")
>boxplot(data$N~data$M,staplewex=0.5,outline=FALSE,main="0.5")
>boxplot(data$N~data$M,staplewex=2,outline=FALSE,main="2")
```



```
#on joue sur la proximité des boîtes avec boxwex
#x11() ouvre une autre fenêtre graphique pour éviter d'écraser le graphe précédent,
>x11()
>par(mfrow=c(2,2))
>boxplot(data$N~data$M,boxwex=1,outline=FALSE,main="1")
>boxplot(data$N~data$M,boxwex=0.1,outline=FALSE,main="0.1")
>boxplot(data$N~data$M,boxwex=0.5,outline=FALSE,main="0.5")
```



REFERENCES BIBLIOGRAPHIQUES



BIBLIOGRAPHIE

[1] Vincent Goulet Introduction à la programmation en R (Québec, avril 2016)

[2] Chambers, J. M. 2008, Software for Data Analysis: Programming with R, Springer, ISBN 978-0-38775935-7.

[3] Becker, R. A. 1994, «A brief history of S», cahier de recherche, AT&T Bell Laboratories. URL http://cm.bell-labs.com/cm/ms/departments/sia/ doc/94.11.ps.

[4] Ihaka, R. et R. Gentleman. 1996, «R: A language for data analysis and graphics», Journal of Computational and Graphical Statistics, vol.5, no 3, p.299– 314.

[5] Emmanuel Paradis, 2005. R pour les débutants (PDF) - Institut des Sciences de l'´Evolution. Université Montpellier II F-34095 Montpellier cédex 05 France

[6] Gaël Millot, 2014. Comprendre et réaliser les tests statistiques à l'aide de R, 3ème édition, 806 pages - tout pour faire des tests statistiques avec R, schéma pp. 259-260 pour choisir un test et la fonction R associée

[7] Anne Philippe, 2004. Notes de cours sur le logiciel R, Université de Nantes, Laboratoire de Mathématiques Jean Leray email : Anne.philippe@math.univ-nantes.fr

[8] Faouzi Lyazrhi, 2005. Une Introduction au langage R, Ecole Nationale Vétérinaire, 23, chemin des Capelles, BP 87614, F-31076 Toulouse cédex email : f.lyazrhi@envt.fr

[9] Kauffmann, Mayeul, 2009. Méthodes Statistiques Appliquées aux Questions Internationales, L'Harmattan, La Librairie des Humanités, 2009, 200 pages.

[10] Tom Short, 2004. R Reference Card, See www.Rpad.org for the source and latest version.

RESSOURCES DOCUMENTAIRES

Livres sur R :

En français

- Gaël Millot, Comprendre et réaliser les tests statistiques à l'aide de R : Manuel de biostatistique

Un excellent aide-mémoire pour les utilisateurs averti et un magnifique tutorial pour les débutants. Livre volumineux (taille d'un annuaire), à avoir sur sa table de chevet

- Pierre-André Cornillon et autres, Statistiques avec R, Presses Universitaires de Rennes

Un livre de base sur le langage R. Les auteurs présentent comment effectuer certaines analyses de base : régression, ACM, analyse de variance. Un livre idéal pour débuter.

En anglais :

- Michael J. Crawley, The R Book, Wiley:
  Un livre très volumineux avec beaucoup de choses. Crawley passe rapidement en revue de nombreuses méthodes statistiques (glm, spatial, survie). Les codes sont souvent basés sur de la programmation basique. Bref un livre à avoir pour s'initier à certaines méthodes statistiques et pour récupérer du code.

- Benjamin M. Bolker ; Ecological Models and Data in R, Princeton University Press:
  C'est LE LIVRE qu'il faut avoir quand on fait de la modélisation en écologie et qu'on utilise R. Idéal pour ceux qui veulent s'initier à la modélisation et à l'inférence statistique avec R et Winbugs. N'hésitez pas à consulter le site internet de Benjamin Bolker qui met beaucoup de choses à disposition de qui veut bien (même une version presque complète de son livre en pdf…)

- Darren J. Wilkinson, Stochastic Modelling for System Biology, Chapman & Hall/CRC Mathematical and Computational Biology Series:
  Très bon livre pour s'initier à la modélisation stochastique. Wilkinson prend beaucoup d'exemples de chimie/biochimie et il est plus facile de rentrer dans le livre quand vous avez des bases de cinétique. De nombreux algorithmes utilisés pour la modélisation stochastique sont présentés et l'auteur a ajouté au livre de nombreuses lignes de code R qui sont les bienvenues…

- Roger S. Bivand, Edzer J. Pebesma, Virgilio Gomez-Rubio, Applied Spatial Data Analysis with R, Springer
  Livre très complet sur les géostatistiques avec R. Parfois difficile à lire c'est un livre qui reste tout à fait intéressant quand on se met au spatial.

- Legendre P. and Legendre L. (2012) Numerical ecology.3rd edition. Elsevier.
- Borcard D., Gillet F. and Legendre P.(2012) Numerical ecology with R. Springer.
- Crawley M.J.(2007) The R Book. John Wiley & Sons, inc.
- Dagnelie P. (2006a) Statistique théorique et appliquée.1.Statistique descriptive et base de l'inférence statistique.3ème édition. De Boeck.

- Dagnelie P. (2006b) Statistique théorique et appliquée.2.Inférence statistique à une et à deux dimensions.3ème édition. De Boeck.

- Millot G. (2008) Comprendre et réaliser les tests statistiques à l'aide de R. 2ème édition. De Boeck.

Ressources francophones

Les ressources en français sont plutôt rares. R est le produit d'un travail collaboratif, intimement lié à la recherche, et un langage de programmation. L'anglais, langue de prédilection pour la communication scientifique et technique, est donc la langue de choix pour tout ce qui touche à R.
Cependant, c'est parfois un obstacle supplémentaire lors de la découverte de R et de son langage de programmation. Comme ce site, ces ressources permettent une plus large diffusion de R auprès d'un public francophone.
Spécial débutant

Le tutoriel R du site du zéro, idéal pour s'initier à la programmation R : Site du zéro
R pour les statophobes de Denis Poinsot (Université de Rennes I) : R pour les statophobes
R pour les débutants d'Emmanuel Paradis : R pour les débutants

RESSOURCES SUR LE WEB

Ressources officielles

La documentation officielle de R est accessible en ligne depuis le site du projet : http://www.r-project.org/
Les liens de l'entrée Documentation du menu de gauche vous permettent d'accéder à différentes ressources.

Les Manuels. Plusieurs manuels sont distribués avec R dans R HOME/doc/manual/ :
- An Introduction to R [R-intro.pdf],
- R Installation and Administration [R-admin.pdf],
- R Data Import/Export [R-data.pdf],
- Writing R Extensions [R-exts.pdf],
- R Language Definition [R-lang.pdf].

Les fichiers correspondants peuvent être sous divers formats (pdf, html, texi, ...) en fonction du type d'installation.

FAQ. R est également distribué avec un FAQ (*Frequently Asked Questions*) localisé dans le répertoire R HOME/doc/html/. Une version de ce RFAQ est régulièrement mise à jour sur le site Web du CRAN : http://cran.r-project.org/doc/FAQ/R-FAQ.html.

Ressources en-ligne. Le site Web du CRAN accueille plusieurs documents et ressources bibliographiques ainsi que des liens vers d'autres sites. On peut y trouver une liste de publications (livres et articles) liées à R ou aux méthodes statistiques[21] et des documents et manuels écrits par des utilisateurs de R[22]. Listes de discussion. Il y a quatre listes de discussion électronique sur R ; pour s'inscrire, envoyer un message ou consulter les archives voir : http://www.R-project.org/mail.html

Listes de discussion. Il y a quatre listes de discussion électronique sur R ; pour s'inscrire, envoyer un message ou consulter les archives voir : http://www.R-project.org/mail.html

La liste de discussion générale "r-help" est une source intéressante d'information pour les utilisateurs (les trois autres listes sont consacrées aux annonces de nouvelles versions, et

aux développeurs). De nombreux utilisateurs ont envoyé sur "r-help" des fonctions ou des programmes qui peuvent donc être trouvés dans les archives. Il est donc important si l'on a un problème avec R de procéder dans l'ordre avant d'envoyer un message à "r-help" et de :

- consulter attentivement l'aide-en-ligne (éventuellement avec le moteur de recherche);
- consulter le R-FAQ ;
- chercher dans les archives de 'r-help' à l'adresse ci-dessus ou en consultant un des moteurs de recherche mis en place sur certains sites Web [23] ;
- lire le « posting guide »[24] avant d'envoyer vos questions.

R News. La revue électronique R News a pour but de combler l'espace entre les listes de discussion électroniques et les publications scientifiques traditionnelles. Le premier numéro a été publié en janvier 2001[25].

Citer R dans une publication. Enfin, si vous mentionnez R dans une publication, il faut citer la référence suivante : R Development Core Team (2005). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL: http://www.R-project.org.

[21] http://www.R-project.org/doc/bib/R-publications.html
[22] http://cran.r-project.org/other-docs.html
[23] Les adresses de ces sites sont répertoriées sur celui du CRAN à http://cran.r-project.org/search.html
[24] http://www.r-project.org/posting-guide.html
[25] http://cran.r-project.org/doc/Rnews/

Groupes d'utilisateurs

Le forum du groupe des utilisateurs du logiciel R : http://forums.cirad.fr/logiciel-R/index.php

Semin-R, un autre groupe d'utilisateurs de R : http://rug.mnhn.fr/semin-r

ANNEXES

TD1

Initiation au logiciel R

Objectif de ce T.D.

Ce T.D. a pour objectif de vous montrer les commandes de base de R (ouverture, fermeture, sauvegarde, aide,...) et de vous faire apprendre les bases du logiciel R.

1 Introduction : Qu'est-ce-que R ?

- R est un logiciel permettant de faire des analyses statistiques et de produire des graphiques.
- Nous allons utiliser R comme une boîte à outils pour faire des analyses statistiques.
- Mais R est également un langage de programmation complet. C'est cet aspect qui fait que R est différent des autres logiciels statistiques.
- Les informations sur R sont disponibles sur la homepage du projet :

http://www.r-project.org/

Si vous avez un ordinateur à la maison, vous pouvez le télécharger à l'adresse : http://cran.r-project.org/mirrors.html

Enfin, R est un clone gratuit du logiciel S-Plus commercialisé par MathSoft et développé par Statistical Sciences autour du langage S (conçu par les laboratoires BELL).

2 Comment se procurer le logiciel R ?

Le logiciel R est gratuit. La page officielle du logiciel est :

http://www.r-project.org/

- Ce logiciel peut être téléchargé à l'adresse précédente et il existe de nombreuses sources d'informations disponibles en ligne. On peut citer entre autre

http://cran.r-project.org/doc/manuals/R-intro.pdf (Eng),

http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf (Fr),

mais aussi des sites web dédiés à la recherche d'information dans R,

http : //www.rseek.org/,

http://en.wikibooks.org/wiki/R_Programming/.

3 Remarques d'ordre général sur R :

- Bien sûr il existe une version française du logiciel R.
- R fonctionne avec plusieurs fenêtres sous Windows. En particulier, nous distinguons la fenêtre **R Console**, fenêtre principale où sont réalisées par d´défaut les entrées de commandes et sorties de résultats en mode texte. A celle-ci peuvent s'ajouter un certain nombre de fenêtres facultatives, telles que les fenêtres graphiques et les fenêtres d'informations (historique des commandes, aide, visualisation de fichier, etc...), toutes appelées par des commandes spécifiques via la console.
- Le menu File ou Fichier contient les outils n´nécessaires a` 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 de commandes, etc.
- Le menu Edit ou Edition contient les habituelles commandes de copier-coller, ainsi que la boite de dialogue autorisant la personnalisation de l'apparence de l'interface.
- Le menu Misc traite de la gestion des objets en mémoire et permet d'arrêter une procédure en cours de traitement.
- Le menu Packages automatise la gestion et le suivi des librairies de fonctions, permettant leur installation et leur mise `a jour de manière transparente au départ du site CRAN (Comprehensive R Archive Network) http://cran.r-project.org/ ou de toute autre source locale.
- Enfin, les menus Windows ou Fenêtres et Help ou 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 de R.
- Ce qui est entré par l'utilisateur figure en rouge, et la réponse de R est en bleu.
- Les nombres entre crochets au début de chaque ligne donnent l'indice du premier nombre de la ligne.
- Quand deux vecteurs ne sont pas de même longueur, le plus court est recyclé.

4 Pour en savoir plus sur R

- Pour un public francophone, un point de départ est le manuel d'Emmanuel Paradis, « R pour les débutants », 81 pages, qui a la particularité d'exister également en version

anglaise « R for Beginners ». Les deux documents sont disponibles ici : http://cran.r-project.org/

- dans la rubrique « Documentation », sous-rubrique « Contributed ».
- Plusieurs milliers de pages d'enseignement en français de statistiques sous R sont disponibles ici : http://pbil.univ-lyon1.fr/R/

5 Pour commencer avec R

Les manipulations décrites ci-dessous s'appliquent à R pour Windows, mais le logiciel existe également pour Mac OS et Linux et les manipulations seront tout à fait similaires si vous utilisez l'un de ces systèmes d'exploitation.

N.B. La version courante de R au moment où ce paragraphe est rédigé est la version 3.5.2. for Windows (79 mégabytes, 32/64 bites). Il est vraisemblable qu'une version plus récente sera disponible lorsque vous installerez R, mais seuls les noms de fichiers devraient être légèrement modifiés.

A l'aide d'un navigateur, affichez le site : http://www.r-project.org Dans la partie gauche de l'écran, allez sur la rubrique Download, packages et cliquez sur le lien CRAN. Choisissez ensuite l'un des sites algériens de téléchargement de R et de ses packages, par exemple : https://cran.usthb.dz/

Cliquez sur le lien : Download R for Windows puis sur le lien : Install R for the first time et enfin sur le lien : Download R 3.5.2 for Windows.

Une fois le téléchargement effectué, allez dans le répertoire où le fichier a été enregistré et exécutez (double clic) le programme d'installation : R-3.5.2-win.exe

Sélectionnez le français comme langue pour l'installateur et les menus de R. Acceptez les choix par défaut de l'installateur pour les autres options. Une fois l'installation terminée, exécutez le logiciel en cliquant sur l'icône R-i386-3.5.2

6 Installer des packages

6.1 Installation de Rcmdr, ade4 et FactoMineR

Avec plus de 13000 packages testés et vérifiés sur le CRAN, R est plus puissant que jamais pour l'analyse de données (ou la data science). R est un logiciel très vivant car de nouvelles fonctionnalités lui sont sans cesse ajoutées au travers de petits ensembles de fonctions regroupées dans un package. Ainsi chaque personne est libre de contribuer au

développement du logiciel en proposant le code R qui permet de réaliser une tâche précise. Ces packages sont regroupés sur un serveur et sont accessibles à tous.

La console de R (fenêtre Rgui) est alors chargée. Utilisez le menu **Packages - Installer les packages.**

Le logiciel demande de choisir dans une liste un site de téléchargement. Sélectionnez : https://cran.usthb.dz/.

Ou encore plus simple, lancer le logiciel R puis taper les commandes suivantes :

```
install.packages("Rcmdr", dependencies=TRUE)
install.packages("ade4", dependencies=TRUE)
install.packages("ade4TkGUI", dependencies=TRUE)
install.packages("FactoMineR", dependencies=TRUE)
```

Sous R, taper : > source("http://factominer.free.fr/install-facto.r")

(sous réserve d'accès à internet)

La (longue) liste des packages s'affiche alors. Sélectionnez le package Rcmdr. Une fois le package installé, essayez de le lancer en saisissant dans la console de R la ligne de commande suivante (en respectant majuscules et minuscules) :

```
>library(Rcmdr)
```

Le logiciel affiche alors une fenêtre d'avertissement indiquant que le fonctionnement correct de Rcmdr exige l'installation de 14 autres packages (car, sem, rgl, relimp, multcomp, lmtest, leaps, Hmisc, e1071, effects, colorspace, aplpack, abind, RODBC) et propose de les installer.

Cliquez sur Installer, puis sur Installer depuis CRAN.

Une fois l'installation de ces packages effectuée, l'interface Rcmdr devrait s'afficher en plus de celle de R.

6.2 R Commander : interface graphique pour réaliser des traitements statistiques avec le logiciel R

R-commander est une interface graphique pour le logiciel R. Elle facilite l'apprentissage de ce langage de programmation en offrant à l'utilisateur la possibilité de réaliser l'importation de données, un certain nombre de traitements statistiques élémentaires ou plus avancés, l'export des résultats de manière interactive tout en indiquant les commandes R correspondantes. Des greffons peuvent être ajoutés pour réaliser d'autres traitements statistiques.

6.2.1 Fonctionnalités générales

- Importation de données :
- depuis un fichier texte ; par copier-coller ; par URL (Windows, Mac, Linux)
- depuis des fichiers Excel, Access, dBase, SPSS, SAS, Minitab, STATA (Windows)
- depuis un paquet R
- Manipulation des données (sélection, réorganisation, édition directe, recodage...)
- Traitements statistiques :
- statistiques descriptives : moyenne, médiane, tableau de contingence
- tests paramétriques (tests t, ANOVA) et non paramétriques (Wilcoxon, Kruskal et Wallis)
- analyse de données : analyse en composantes principales, analyse factorielle, analyse discriminante, classification
- modélisation : régression linéaire, régression logistique simple, multinomiale et ordinale, modèles linéaires généralisés
- Représentations graphiques :
- diagramme en barres, en points, camembert, boîte à moustaches, histogramme, comparaison de quantiles...
- diagnostics de modèles
- les graphiques peuvent être copiés ou exportés dans un format vectoriel pour une meilleure qualité d'impression
- Probabilités, courbes de répartition et données aléatoires à partir de nombreuses distributions.
- Une trentaine de greffons disponibles fournissent d'autres possibilités d'analyses : analyse de durée/survie, analyse de données « à la française », analyse textuelle...

6.2.2 Interopérabilité

Importation depuis plusieurs formats courants ; exportation au format CSV ou TSV.

6.3 Utilisation de FactoMineR sous Rcmdr

6.3.1 Fonctionnalités générales

FactoMineR est un package du logiciel libre R dédié à l'analyse de données.

Il permet de réaliser les méthodes classiques (ACP, AFC, ACM, classification) et avancées (AFM, AFM Hiérarchique, AFM Duale).

De nombreux indicateurs et sorties graphiques sont disponibles.

Toutes les analyses peuvent être effectuées à l'aide d'un menu déroulant convivial.

La gestion des données manquantes en ACP, ACM et AFM est possible grâce à l'utilisation conjointe du logiciel libre R missMDA.

6.3.2 Interopérabilité

- Il peut prendre en entrée des extensions csv, txt, xls.
- Échanges possible avec Excel, SPSS, SAS, langage SQL.

6.3.3 Contexte d'utilisation

FactoMineR est utilisé aussi bien en recherche qu'en enseignement et développement.
Il est utilisé par de nombreux organismes de recherche, des étudiants de différentes filières en France comme à l'étranger (dans plus de 70 pays).
FactoMineR s'adresse à un public aussi bien de statisticiens que de chercheurs ou étudiants d'autres disciplines scientifiques.
Sélectionner le tableau sous Rcmdr puis dans le menu FactoMineR choisir la méthode.

- Vous pouvez choisir les variables et les individus en supplémentaires.
- Les résultats sont disponibles sous R dans le fichier de sortie noté par défaut res.
- Vous pouvez également sauver un fichier excel avec outputs.
- Pour afficher des plans factoriels, sélectionner dans FactoMineR les axes à représenter

Le nombre très important de packages disponibles, la rapidité avec laquelle les nouvelles méthodes statistiques sont proposées sous cette forme font aujourd'hui le grand succès du logiciel R.

N.B. On pourra faciliter le chargement de R en créant un raccourci vers le programme Rgui.exe sur le bureau. On peut alors améliorer le fonctionnement de R Commander en modifiant la ligne de commande associée à ce raccourci.
- Faites un clic droit sur le raccourci et sélectionnez l'item de menu "Propriétés".
- Ajoutez **--sdi** à la fin de la ligne de commande, dans le champ "Cible" :

7 Démarrer R :

Vous lancez le logiciel R en cliquant sur l'icône R. Le symbole > signifie que R est prêt à travailler. Il ne faut pas taper ce symbole au clavier car il est déjà` présent en début de ligne sur la R Console. C'est à la suite de ce symbole > que vous pourrez taper les commandes R. Une fois la commande tapée, vous devez toujours la valider par la touche Entrée.

7.1 R et R Commander : l'interface utilisateur

7.1.1 Fonctionnalités générales

R-commander est une interface graphique pour le logiciel R. Elle facilite l'apprentissage de ce langage de programmation en offrant à l'utilisateur la possibilité de réaliser l'importation de données, un certain nombre de traitements statistiques élémentaires ou plus avancés,

l'export des résultats de manière interactive tout en indiquant les commandes R correspondantes. Des greffons peuvent être ajoutés pour réaliser d'autres traitements statistiques.

7.1.2 Autres fonctionnalités

- Importation de données :
- depuis un fichier texte ; par copier-coller ; par URL (Windows, Mac, Linux)
- depuis des fichiers Excel, Access, dBase, SPSS, SAS, Minitab, STATA (Windows)
- depuis un paquet R
- Manipulation des données (sélection, réorganisation, édition directe, recodage...)
- Traitements statistiques :
- statistiques descriptives : moyenne, médiane, tableau de contingence
- tests paramétriques (tests t, ANOVA) et non paramétriques (Wilcoxon, Kruskal et Wallis)
- analyse de données : analyse en composantes principales, analyse factorielle, analyse discriminante, classification
- modélisation : régression linéaire, régression logistique simple, multinomiale et ordinale, modèles linéaires généralisés
- Représentations graphiques :
- diagramme en barres, en points, camembert, boîte à moustaches, histogramme, comparaison de quantiles...
- diagnostics de modèles
- les graphiques peuvent être copiés ou exportés dans un format vectoriel pour une meilleure qualité d'impression
- Probabilités, courbes de répartition et données aléatoires à partir de nombreuses distributions.
- Une trentaine de greffons disponibles fournissent d'autres possibilités d'analyses : analyse de durée/survie, analyse de données « à la française », analyse textuelle...

7.1.3 Les fenêtres de travail

Au chargement du logiciel, R affiche la console

On dispose alors d'un outil permettant de faire des calculs en mode "ligne de commande".

Essayez par exemple :

```
>5 + 3
>mean(c(1,4,8))
```

Pour faciliter l'utilisation de cet outil, nous utilisons une surcouche logicielle fournissant une interface utilisateur plus conviviale : R Commander.

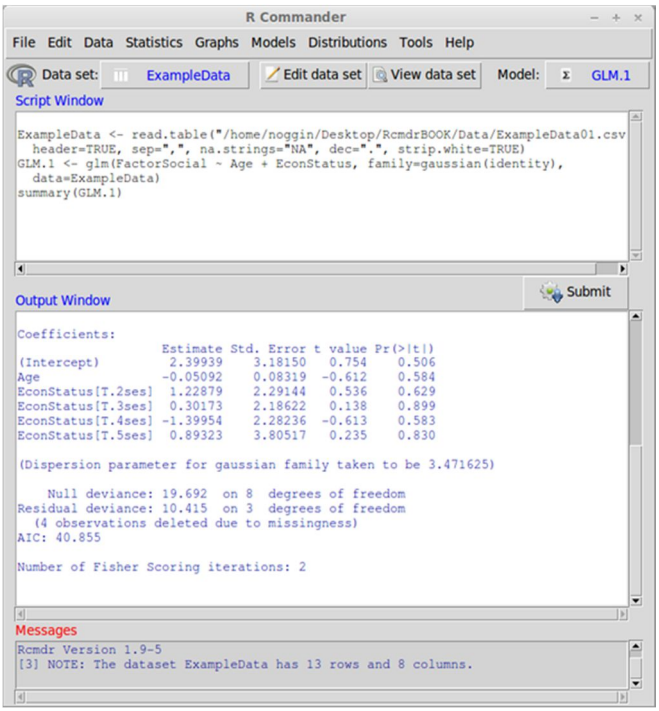
Pour activer R Commander, saisissez dans la console la commande :

```
library(Rcmdr)
```

ou, de manière alternative, utilisez le menu Packages

> Charger le package et sélectionnez Rcmdr dans la liste qui s'affiche.

S'affiche alors une fenêtre disposant d'une barre de menu, d'une barre d'outils, d'une fenêtre de script, d'une fenêtre de sortie et d'une fenêtre de messages.



a) **Sauvegarder sous R :**

Si vous quittez R en choisissant la sauvegarde de l'espace de travail, deux fichiers sont créés :

(i) le fichier. Rdata contient des informations sur les variables utilisées,

(ii) le fichier. Rhistory contient l'ensemble des commandes utilisées.

b) **Quitter R :**

Pour quitter R, vous utilisez la commande

>q()

La question Save workspace image ? [y/n/c] est posée : R propose de sauvegarder le travail effectué. Trois réponses possibles : y (pour yes), n (pour no) ou c (pour cancel, annuler). En tapant c, la procédure de fin de session sous R est annulée. Si vous tapez y, cela permet que les commandes tapées pendant la session soient conservées en mémoire et soient donc « rappelables » (mais vous ne pouvez pas les imprimer).

TD2

1 Syntaxe et manipulation de données

1.1 Commentaires

Dans R, tout ce qui suit le caractère # (= dièse) est un commentaire et n'est pas pris en compte par R :

> # Ceci est du baratin qui n'est pas pris en compte par R !

1.2 Variables

R étant prévu pour faire des calculs statistiques, il ne manipule que des tableaux de données. Ces tableaux sont stockés dans des variables, ce qui permet de leur donner un nom. Le nom des variables doit commencer par une lettre et peut contenir des lettres, des chiffres, des points et des caractères de soulignement (-), mais surtout pas d'espace. L'opérateur = est utilisé pour donner une valeur à une variable ; il peut se lire "prend la valeur de"(NB : on trouve aussi l'opérateur <- ("flèche") qui a exactement la même signification).

Pour afficher la valeur de la variable âge, il suffit de taper le nom de la variable :

```
> âge = 28
```

Pour afficher la valeur de la variable âge, il suffit de taper le nom de la variable :

```
> âge
[1] 28
```

La variable "âge" est ici un tableau avec une seule case, qui contient le chiffre 28. Le 1 entre crochet indique qu'il s'agit de la case n°1. R numérote les cases des tableaux en commençant à 1.

1.3 Types de donnée

R peut manipuler des nombres entiers, des flottants (= nombres à virgule), des chaines de caractère et des booléens (valeur vraie ou fausse) :

```
> âge = 28    # Entier
> poids = 64.5   # Flottant
> nom = "Jean-Baptiste Lamy"   # Chaine de caractère
> enseignant = TRUE     # Booléen
> étudiant = FALSE     # Booléen
```

Enfin, la valeur spéciale NA (non available) est utilisée lorsqu'une donnée est inconnue.

1.4 Tableaux

R définit plusieurs types de tableaux que nous allons voir ; il y a peu de différences entre eux et tous s'utilisent de la même manière.

1.4.1 Vecteurs

Un vecteur est un tableau à une dimension. Toutes les cases du vecteur doivent contenir des données du même type (des entiers, des chaînes de caractère,...). La fonction c() permet de créer un vecteur :

```
> ages=c(28,25,23,24,26,23,21,22,24,29,24,26,31,28,27,24,23,25,27,25,24,21,24,23,25,31)
> ages
[1] 28 25 23 24 26 23 21 22 24 29 24 26 31 28 27 24 23 25 27 25 24 21 24 23 25
[26] 31
```

« [1] » indique que ce qui suit est la première valeur du vecteur, et « [26] » que la deuxième ligne commence à la 26e valeur ; « [1] » et « [26] » ne sont pas des éléments du vecteur.

```
> hbA1c_groupe_temoin = c(75.0, 69.2, 75.4, 87.3)
> hbA1c_groupe_intervention = c(70.5, 64.2, 76.4, 81.6)
> hbA1c = c(hbA1c_groupe_temoin, hbA1c_groupe_intervention)
> hbA1c
 [1] 75.0 69.2 75.4 87.3 70.5 64.2 76.4 81.6
```

La fonction c() permet aussi de concaténer (= mettre bout à bout) des vecteurs.

```
> hbA1c[2]
[1] 69.2
```

Il est possible d'accéder à un élément du vecteur avec des crochets. Par exemple pour accéder au second élément du vecteur poids.

```
> hbA1c[hbA1c > 70.0]

[1]  75.0  75.4  87.3  70.5  76.4  81.6
```

Il est aussi possible d'accéder à l'ensemble des poids répondant à une condition, par exemple l'ensemble des poids supérieurs à 70.0.

```
> length(hbA1c)
[1] 8
> length(hbA1c[hbA1c > 70.0])
[1] 6
```

Enfin, la fonction length() permet de récupérer le nombre d'éléments d'un tableau.

```
> seq(1, 10)
[1] 1 2 3 4 5 6 7 8 9 10
 > seq(1, 10, by = 0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
[16] 8.5 9.0 9.5 10.0
```

Il est possible de créer un vecteur contenant une suite de nombres entiers avec la fonction seq.

1.4.2 Matrices

Une matrice est un tableau à deux dimensions, c'est-à-dire avec des lignes et des colonnes. Comme pour les vecteurs, toutes les cases d'une matrice doivent contenir des données du même type. Une matrice est créée à partir d'un vecteur contenant les valeurs, et d'un nombre de ligne et de colonne.

```
> ma_matrice = matrix(c(1.5, 2.1, 3.2, 1.6, 1.4, 1.5),nr=3, nc=2)
```

```
> ma_matrice
```

```
     [,1] [,2]
[1,]  1.5  1.6
[2,]  2.1  1.4
[3,]  3.2  1.5
```

nr, pour Number of Row nc, pour Number of Column

```
> ma_matrice[1, 1]
 [1] 1.5
```

Les éléments de la matrice peuvent être accédés en donnant entre crochets le numéro de la ligne puis celui de la colonne.

```
> ma_matrice[1,]
[1] 1.5  1.6
```

Il est aussi possible de récupérer une ligne ou une colonne entière, en omettant le numéro correspondant.

1.4.3 Listes

```
> ma_liste = list("JB", 28)
```

Une liste est un tableau à une dimension, qui peut contenir des données de différents types (contrairement au vecteur).

1.4.4 Tableaux de donnée *(data frame en anglais)*

Un tableau de donnée est un tableau où chaque colonne correspond à un attribut différents (âge, taille, poids par exemple) et chaque ligne à un individu différent. Il est possible de créer des tableaux de données dans R, cependant il est beaucoup plus facile de les charger à partir d'un fichier. On utilise pour cela la fonction read.table(). Voici le fichier taille poids.csv :

```
Nom, taille, poids
Jean-Baptiste Lamy, 1.70, 64.0
Bertrand Lamy, 1.80, 63.0
M. X, 1.67, 70.5
M. Y, 1.69, 95.0
M. Z, 1.75, NA
```

Ce fichier peut être chargé ainsi dans R :

```
t = read.table ("taille_poids.csv", sep=",", header=TRUE)
```

sep="," indique que dans le fichiers les différentes colonnes sont séparées par des virgules, et header=TRUE indique que la première ligne du fichier contient les noms des colonnes. Ces valeurs correspondent aux fichiers CSV (Comma-Separated Value file : fichier dont les valeurs sont séparées par des virgules) qui peuvent être généré facilement, par exemple en exportant depuis un tableur comme Open Office ou Excel.

```
>t
                  nom taille poids
1 Jean-Baptiste Lamy   1.70  64.0
2       Bertand Lamy   1.80  63.0
3              M. X    1.67  70.5
4              M. Y    1.69  95.0
5              M. Z    1.75   NA
```

Les noms des colonnes sont disponibles via la fonction names() :

```
> names(t)
[1] "nom" "taille" "poids"
```

Comme pour les matrices, il est possible d'accéder aux cases, lignes et colonnes d'un tableau. Les noms des colonnes peuvent être utilisés `a la place de leurs index :

```
> t[1, 2]   # Première ligne, deuxième colonne : taille du premier individu
[1] 1.7
> t[1, "taille"]   # Première ligne, colonne taille : pareil
[1] 1.7
> t[1,]   # Première ligne
                  nom     taille      poids      IMC
1 Jean-Baptiste Lamy     1.7         64       22.14533
> t[,"taille"]    # Colonne taille
[1] 1.70 1.80 1.67 1.69 1.75
> t$taille     # Notation raccourci pour la colonne taille
[1]   1.70   1.80   1.67  1.69  1.75
```

Il est aussi possible d'indexer avec une condition : par exemple, pour obtenir un tableau avec seulement les individus dont la taille est supérieure à 1m70 (ne pas oublier la virgule, qui sert à indiquer que l'on veut récupérer toutes les colonnes !) :

```
>  t[t$taille > 1.7,]
                nom    taille      poids    IMC
2 Bertrand Lamy    1.7         64       22.14533
5              M. Z    1.75        NA       NA
```

Enfin, la fonction write.table() permet d'enregistrer un tableau de donnée (row.names = FALSE permet de désactivé les numéros de ligne) :

```
write.table(t, "taille_poids_2.csv", sep = ",", row.names = FALSE)
```

Ce fichier pourra ensuite être rechargé avec read.table () comme ci-dessus.

Ce fichier pourra ensuite être rechargé avec read.table () comme ci-dessus.

1.5 Opérateur et calcul

1.5.1 Opérations

R permet de réaliser la plupart des opérations courantes à l'aide des opérateurs suivants : + (addition), - (soustraction), * (multiplication), / (division), ^ (puissance).

```
> 2 * 3 + 1
 [1] 7
```

Les opérations sont réalisées sur chaque élément des tableaux. Par exemple, pour calculer l'Indice de Masse Corporelle (IMC) sur chaque individu du tableau de donnée chargé précédemment, en appliquant la formule IMC = poids/taille^2 :

```
> t$poids / (t$taille ^ 2)
[1]  22.14533  19.44444  25.27878  33.26214  NA
```

R a calculé l'IMC pour chacun des 5 individus ! Notez que la donnée manquante (NA) se "propage", ce qui est logique : si le poids de M. Z est manquant, il n'est pas possible de calculer son IMC, qui est donc manquant lui aussi. Il est possible d'ajouter une quatrième colonne avec l'IMC `a notre tableau de la manière suivante :

```
> t$IMC = t$poids / (t$taille ^ 2)
> t
                  nom  taille  poids    IMC
1 Jean-Baptiste Lamy    1.70     64.0  22.14533
2 Bertand Lamy          1.80     63.0  19.44444
3               M. X    1.67     70.5  25.27878
4               M. Y    1.69     95.0  33.26214
5               M. Z    1.75     NA    NA
```

1.5.2 Comparaison

Les comparaisons se font avec les opérateurs <, >, <= (inferieur ou égal), >= (supérieur ou égal), == (égal), != (différent de). Ils retournent une (ou plusieurs) valeur(s) booléenne(s).

```
> 1 < 3
[1] TRUE
> t$IMC > 25
[1]  FALSE  FALSE  TRUE  TRUE  NA
```

Il est possible de combiner plusieurs comparaisons avec des & (et) ou des | (ou).

TD3

Importation lecture et exportation de données

1 Consulter les données disponibles sur R

Consulter les données disponibles sur les packages chargées en mémoire : data()

- Consulter les données disponibles sur tous les packages du poste :

```
>data(package = packages(all.available = TRUE))
```

- Pour accéder aux données disponibles dans R

```
>data(infert)   #accès aux données infert
```

- Lire les données d'un fichier

R peut lire les données stockées dans des fichiers texte (ASCII) grâce, entre autres, aux fonctions suivantes : read.table (), scan ().

2 Importer depuis des logiciels de statistique

Lorsque les données ont été sauvegardées sous le format propriétaire d'un logiciel statistique tiers, il est nécessaire de disposer d'outils permettant leur transfert vers le système R. Plusieurs extensions existent pour importer des fichiers de données issus d'autres logiciels de statistiques, la librairie foreign offre ces outils pour une sélection des logiciels statistiques les plus courants, à savoir SAS, SPSS et Stata. Installée par défaut avec R et décrite en détails dans le manuel R Data Import/Export disponible sur http://cran.r-project.org/manuals.html. Un des soucis majeurs de cette extension réside dans la manière dont elle traite les métadonnées utilisées en particulier dans les fichiers SAS, SPSS et Stata, à savoir les étiquettes de variable, les étiquettes de valeur et les valeurs manquantes déclarées. En effet, chaque fonction va importer ces métadonnées sous la forme d'attributs dont le nom diffère d'une fonction à l'autre.

Par exemple, la fonction read.spss prend en charge les données enregistrées au moyen des commandes save et export de SPSS. Enfin, foreign ne sait pas toujours importer les différents types de variables représentant des dates et des heures.
SPSS.

Les fichiers générés par SPSS sont de deux types : les fichiers SPSS natifs natifs (extension .sav) et les fichiers au format SPSS export (extension .por).
Dans les deux cas, on aura recours à la fonction read_spss

```
>library(haven)
>données<- read_spss("data/fichier.sav", user_na=TRUE)
```

Si vous préférez utiliser l'extension foreign, la fonction correspondante est read.spss. On indiquera à la fonction de renvoyer un tableau de données avec l'argument to.data.frame = TRUE.
Par défaut, les variables numériques pour lesquelles des étiquettes de valeurs ont été définies sont transformées en variables de type facteur, les étiquettes définies dans SPSS étant utilisées comme *labels* du facteur. De même, si des valeurs manquantes ont été définies dans SPSS, ces dernières seront toutes transformées en NA (R ne permettant pas de gérer plusieurs types de valeurs manquantes). Ce comportement peut être modifié avec use.value.labels et use.missings.

```
>library(foreign)
>donnees <- read.spss("data/fichier.sav", to.data.frame = TRUE, use.value.labels = FALSE, use.missings = FALSE)
```

Il est important de noter que read.spss de l'extension foreign ne sait pas importer les dates. Ces dernières sont donc automatiquement transformées en valeurs numériques.

Par ailleurs, la fonction read.xls du package gdata fournit les outils pour lire des fichiers au format excel. Pour plus d'informations, je vous renvoie au document proposé par le «R development core team» disponible gratuitement sur internet à l'adresse suivante : http://www.r-project.org.
Dans ce TD, nous nous limitons à la lecture des fichiers au format ASCII.

3 Utilisation de la fonction read.table ( )

- Importer dans un objet nommé A le jeu de donné nommé auto2004_original.txt :

```
>A = read.table('chemin/auto2004_original.txt', header = TRUE, sep = '\t')
```

- Importer dans un objet nommé B le jeu de donnée auto2004_sans_nom.txt :

```
>B = read.table('chemin/auto2004_sans_nom.txt', sep = '\t')
```

- Importer dans un objet nommé C le jeu de donnée auto2004_virgule.txt :

>C = read.table ('chemin/auto2004_virgule.txt', header = TRUE, sep = '\t', dec = ',')

- Importer dans un objet nommé D le jeu de donnée auto_don_manquante.txt

>D = read.table ('chemin/auto2004_don_manquante.txt', header = TRUE, sep = '\t')

Importer dans un objet nommé E le jeu de donnée auto_don_manquante(99999).txt :

>E = read.table ('chemin/auto2004_don_manquante(99999).txt', header = TRUE, sep = '\t', na.strings = 99999)

Quel est le mode des objets créer par la fonction read.table ( ) ? :

Le mode des objets crées par la fonction read.table est la data.frame :

>is.data.frame(A)

Remarque : on peut créer des objets d'un autre mode que data.frame en utilisant la fonction scan(). Nous ne traiterons pas cette situation dans le TD.

3.1 Lecture de données tabulaires

Les fonctions read.table() et read.csv() permettent de lire et importer des fichiers *.txt* et *.csv*. Le résultat va se trouver dans une structure de type "data frame". Ces trois fonctions sont identiques, mais possèdent d'autres défauts. La syntaxe de read.table() est assez riche est compliquée. Ci-dessous, on indique juste quelques exemples.

3.1.1 Excel

Une première approche pour importer des données Excel dans R consiste à les exporter depuis Excel dans un fichier texte (texte tabulé ou CSV) puis de suivre la procédure d'importation d'un fichier texte.

Une feuille Excel peut également être importée directement avec l'extension readxl qui appartient à la même famille que haven et readr.

La fonction read_excel permet d'importer à la fois des fichiers .xls (Excel 2003 et précédents) et .xlsx (Excel 2007 et suivants).

```
>library(readxl)
>donnees <- read_excel("data/fichier.xlsx")
```

Une seule feuille de calculs peut être importée à la fois. On pourra préciser la feuille désirée avec sheet en indiquant soit le nom de la feuille, soit sa position (première, seconde, …).

```
>donnees <- read_excel("data/fichier.xlsx", sheet = 3)
>donnees <- read_excel("data/fichier.xlsx", sheet = "mes_donnees")
```

On pourra préciser avec col_names si la première ligne contient le nom des variables.

Par défaut, read_excel va essayer de deviner le type (numérique, textuelle, date) de chaque colonne. Au besoin, on pourra indiquer le type souhaité de chaque colonne avec col_types.

RStudio propose également pour les fichiers Excel un assitant d'importation, similaire à celui pour les fichiers texte, permettant de faciliter l'import.

R peut directement lire un fichier depuis un URL.

On peut indiquer source la fonction file.choose(). Cela permet à l'utilisateur de sélectionner un fichier.

```
# Le fichier data.txt est lu est stocké dans un nouveau objet R nommé Database
>Database <- read.table("data.txt", header = TRUE)

# Le séparateur utilisé dans le fichier délimité est la virgule
>Database <- read.table(file.choose(), header = TRUE, sep = ",")

# Fichier de type CSV. Le séparateur utilisé dans le fichier csv est le point-virgule. On force un encodage UTF-8
>Database <- read.csv(file.choose(), header = TRUE, sep= ";", encoding="UTF-8")

# Fichier de type CSV depuis un serveur web (ce fichier contient des stats de google webmaster tools pour edutechwiki ...)

>Database_webmaster <- read.csv("http://tecfa.unige.ch/guides/R/data/edutechwiki-fr-gw-oct-6-2014.csv", header = TRUE, sep= ",")

# Fichier de type Excel qui contient une simple matrice, la première ligne contient les noms de variables
>library(xlsx)

>Database <- read.xlsx("c:/dks/myexcel.xlsx", 1)

# Pour lire un fichier .txt tabulation contenant des données avec des noms de lignes et des noms de colonnes:

X<-read.table('C:/my_file.txt', sep='\t', header=TRUE, row.names=1)
```

header=TRUE indique que les colonnes du fichier ont une entête (correspondant à la première ligne)

row.names=1 indique les lignes du fichier ont un nom (correspondant à la première colonne)

3.2 Pour copier-coller des tableaux depuis Excel vers R

>read.delim("clipboard", header = T) # commande pour copier et coller des tableaux à partir d'Excel ou d'autres programmes dans R. Si l'argument "header" est défini à FALSE, alors la première ligne du jeu de données ne sera pas utilisée comme colonne titres.

La fonction read.delim est beaucoup plus souple pour importer une table avec des champs vides ou de longues chaines de caractères telle que la description d'un gène.

3.3 Pour éditer des données

>xnew <- edit(xold)

Pour entrer de nouvelles données via l'interface de tableur R, utiliser la commande suivante,

>xnew <- edit(data.frame())

3 Exportation des données de R vers un fichier

Pour exporter les données dans un fichier .txt tabulation

>write.table (mydata, "c:/mydata.txt", sep="\t")

4 Enregistrer des données R

Créer la matrice suivante :

>matrice = matrix(1:25, 5, 5)

Sauver la matrice sous le nom de ‘matrice.txt’ à une adresse valide. Que remarquez-vous ?

write.table (matrice, 'chemin/matrice.txt')

On remarque que par défaut les lignes et les colonnes sont nommées

Ajouter des arguments à la commande précédente pour retirer des noms aux lignes et aux colonnes du fichier crée :

write.table (matrice, 'chemin/matrice.txt', row.names = F, col.names = F)

Sauvegarder la data.frame A dans le fichier nommé auto.txt :

write.table (A, chemin/auto.txt')

Sauver les objets disponibles en mémoire vive à l'adresse '/chemin/Donnees.Rdata'

```
save(list = ls(all = TRUE), file = "/chemin/Donnees.Rdata")
```

Remarque : Si on ne spécifie pas le chemin de sauvegarde, R sauve les données à l'adresse données par la fonction getwd(). On peut modifier le chemin par défaut grâce à la fonction setwd() Par exemple: (setwd("/Documents/donnees").

Sauver les objets disponibles en mémoire vive à l'adresse "/chemin/" grâce à la commande setwd() :

```
>setwd("/chemin/")
>save.image()
```

TD 4

Les graphiques avec R

Faire un graphique pour représenter ses données permet de compléter l'information de la statistique descriptive. La visualisation des données est une information en soi et ouvre des pistes de travail dans l'analyse de ses données. Il existe des représentations graphiques de données très nombreuses et R offre une variété de graphiques remarquables. Pour avoir une petite idée des possibilités offertes, il est possible de taper la commande > demo(graphics) Il n'est pas possible ici de détailler tous ces graphiques et les options dans les fonctions qui les utilisent. Nous nous limiterons aux principales fonctions et aux plus intéressantes.

1. Rappels de vocabulaire

Données univariées lorsqu'il n'y a qu'une seule variable

Données bivariées lorsqu'il y a deux variables

Données multivariées lorsqu'il y a plusieurs variables

Données quantitatives, **données numériques** lorsque ces variables sont des nombres sur lesquels les opérations arithmétiques (addition, soustraction, moyenne,...) ont un sens : par exemple, des températures, des concentrations, des poids, des volumes,... mais pas des numéros de d´département.

Données qualitatives dans le cas contraire : valeurs booléennes (vrai ou faux, oui ou non), ou définies par un ensemble de valeurs possible (par exemple une variable "fréquence" pouvant prendre les valeurs "rare" , "fréquent", "très frèquent"; ou bien une variable "souche d'Aspergillus" pouvant prendre les valeurs "Fumigatus", "Niger" , "Lentullus" ; ou des numéros de département).

Données ordonnées lorsqu'il est possible de classer les valeurs de la variable selon un ordre. Les données quantitatives sont toujours ordonnées, et certaines données qualitatives le sont aussi (par exemple il est possible d'ordonner des fréquences mais pas des souches d'Aspergillus).

2 Graphiques avec R

Faire un graphique pour représenter ses données permet de compléter l'information de la statistique descriptive. La visualisation des données est une information en soi et ouvre

des pistes de travail dans l'analyse de ses données. Il existe des représentations graphiques de données très nombreuses et R offre une variété de graphiques remarquables. Pour avoir une petite idée des possibilités offertes, il est possible de taper la commande :

>demo(graphics).

Il n'est pas possible ici de détailler tous ces graphiques et les options dans les fonctions qui les utilisent. Nous nous limiterons aux principales fonctions et aux plus intéressantes.

2.1 Graphique à 1 variable

2.1.1 Boites à moustaches simple : 1 variable numérique

Pour afficher une boite `a moustache simple :

> boxplot(variable_numérique)

Lorsque l'on a plusieurs variables INDÉPENDANTES, il est possible de placer plusieurs boites `a moustaches les unes à côte des autres, en séparant les variables par des virgules :
> boxplot (variable1_numérique, variable2_numérique, variable3_numérique,...)

2.1.2 Histogramme : 1 variable numérique

C'est une représentation plus classique et très utile ; on l'obtient avec :

> hist(variable_numérique)

Par défaut R applique la loi de Sturges (cf cours) pour d´déterminer le nombre de barres ; il est possible de préciser le nombre de barres manuellement (ici, 3), ou bien les valeurs auxquelles auront lieu les coupures :

> hist(variable_numérique, breaks = 3)
> hist(variable_numérique, breaks = c(0.0, 0.2, 0.5, 0.6, 1.0))

2.1.3 Camembert : 1 variable qualitative

Un camembert se fait avec la commande pie :

> pie(variable_numérique) Pour faire un camembert `a partir d'une donnée qualitative, il faut d'abord stratifier celle-ci avec la commande summary :

> pie(summary(factor(variable_qualitative)))

2.2 Graphique à 2 variables

Il est fréquent d'étudier une variable en fonction d'une (ou plusieurs) autres variables : par exemple la réponse biologique d'un organisme `a une substance en fonction de la dose de substance,...

2.2.1 Boites à moustaches : 1 variable numérique + 1 variables qualitatives

Pour afficher une boite à moustache de la variable 1 pour chaque valeur possible de la variable 2 (la variable 1 est numérique et la variable 2 est qualitative) :

```
> boxplot(variable1_numérique ~ variable2_qualitative)
```

2.2.2 Graphique à deux dimensions (X, Y) : 2 variables numériques

La première variable est placée en X, la seconde en Y. Il est possible d'utiliser des variables numériques ou qualitatives, cependant ce type de graphique est surtout utile avec des variables qualitatives.

```
> plot(variable_numerique1, variable_numerique2)
```

2.2.3 Barre cumulée : 2 variables qualitative

Un diagramme en "barre cumulée" permet d'étudier 2 variables qualitatives :

```
> plot(factor(variable_qualitative1), factor(variable_qualitative2))
```

2.3 Graphique à 3 variables et plus

2.3.1 Boites à moustaches : 1 variable numérique + 2 variables qualitatives

Pour afficher une boite à moustache sur trois variables (ou plus) : ici on affiche une boite à moustache pour la variable 1 pour chaque combinaison des variables 2 et 3 :

```
> boxplot(variable1_numerique ~ (variable2_qualitative + variable3_qualitative))
```

Astuce Le nombre de boites peut vite devenir important ; l'ensemble est souvent plus lisible `a l'horizontal, et en mettant les étiquettes des axes à l'horizontal (las = 2) :

```
> boxplot(variable1_numerique ~ (variable2_qualitative + variable3_qualitative),
horizontal = TRUE, las = 2)
```

2.3.2 Graphique de niveau : 1 variable numérique + 2 variables qualitatives

Avant d'utiliser ce type de graphique, il faut importer le module d'extension "lattice" (il suffit de le faire une seule fois) :

```
> require(lattice)
```

Les graphiques de niveau permettent d'étudier une variable numérique en fonction de deux variables qualitatives :

> levelplot(variable_num´erique1 ~ variable_qualitative2 * variable_qualitative3)

Chaque "case" du graphique obtenue correspond à une valeur de la variable 2 et une valeur de la variable 3 ; la couleur de la case indique la valeur moyenne de la variable 1.

2.3.3 Nuage en 3D : 3 variables numériques

Avant d'utiliser ce type de graphique, il faut importer le module d'extension "lattice" (il suffit de le faire une seule fois) :

> require(lattice)

Les nuages de point en 3D permettent d'étudier le comportement de 3 variables numériques simultanément :

> cloud(variable_num´erique1 ~ variable_num´erique2 * variable_num´erique3)

2.4 Mise en forme des graphiques

Il existe de nombreuses options pour mettre en forme les graphiques. En voici quelques-unes :

main titre du graphique sub sous-titre du graphique

xlab titre de l'axe X

ylab titre de l'axe Y

D'autres options sont disponibles ; pour les obtenir, consultez l'aide, par exemple pour les histogrammes :

> help(hist)

3 Gestion des fenêtres graphiques

Lorsqu'une fonction graphique est exécutée, R ouvrira une fenêtre graphique et y affichera le graphe. On peut spécifier le dispositif de gestion des fenêtres. La liste des dispositifs graphiques disponibles dépend du système d'exploitation.

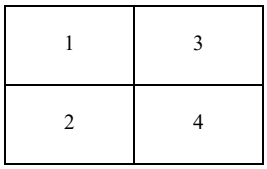
3.1 Partitionner un graphique

R propose deux manières de partitionner les graphiques.

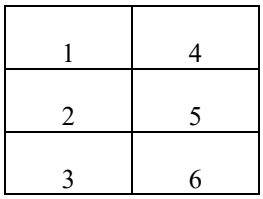
a) La commande split.screen(c(1, 2)) va diviser le graphique en deux parties qu'on sélectionnera avec les commandes sreen(1) et screen(2).

b) La fonction layout() partitionne le graphique actif en plusieurs parties sue lesquelles sont affichés les graphes successivement. Cette fonction a pour argument une matrice de valeurs entières qui indiquent le numéro des sous-fenêtres. Pour visualiser la partition crée, on utilise la fonction layout.show avec, en argument, le nombre de sous fenêtres. Par exemple, si on veut diviser la fenêtre en quatre parties égales, on tapera la commande suivante layout(matrix(1:4, 2, 2)) ; et pour visualiser la partition crée, on utilisera la commande layout.show(4).

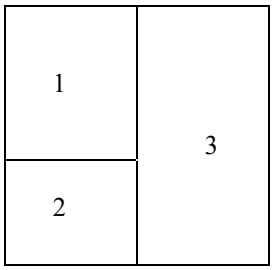
layout(matrix(1:4, 2, 2)) layout.show(4)



layout(matrix(1:6, 3, 2)) layout.show(6)



mat = matrix(c(1:3, 3), 2, 2) layout(mat) layout.show(3)



4 Utilisation de fonctions graphiques de R

4.1 Utilisation de la fonction plot()

Premier exercice

Créer une matrice, mat1, composée de 100 lignes et deux colonnes ; chacune des colonnes des vecteurs aléatoires de loi normale centrée et de variance 1.

```
>mat1 = matrix(rnorm(200, 0, 1), 100, 2)
```

Créer une matrice, mat2, composée de 100 lignes et deux colonnes ; chacune des colonnes étant des vecteurs aléatoires de loi normale de moyenne 1 et de variance 1.

```
<mat2 = matrix(rnorm(200, mean = 1, sd = 0.5), 100, 2)
```

Tracer le graphe bivarié de la première colonne de mat1 sur la deuxième colonne de mat1

```
>plot(mat1)
```

Ajouter au graphe précédent le graphe bivarié de la première colonne de mat2 sur la deuxième colonne de mat2. Que constatez-vous ?

Remarque : On utilisera des couleurs différentes pour distinguer les points de mat1 des points de mat2. Pour connaître la liste des couleurs disponible sur R, tapez la commande colors().

```
>plot(mat1, col = "blue")
>points(mat2, col = "red")
```

On constate que les points de mat2 ne sont afficher que dans le graphique définit par les bornes de mat1.

Utiliser les arguments xlim et ylim pour contourner cette problématique.

```
>mat = rbind(mat1, mat2)
>plot(mat1, col = "blue",  xlim = range(mat[ ,1]),     ylim = range(mat[ ,2]),  main =
"représentation d'un nuage de points",     xlab = "X1", ylab = "X2"     ) points(mat2, col =
"red")
```

4.1.1 Utiliser les options de la fonction plot() pour générer un graphique esthétique

Par exemple :

```
>plot(1,     xlim = range(mat[ ,1]),     ylim = range(mat[ , 2]),     main = "représentation
d'un nuage de points",     xlab = "X1", ylab = "X2",     bty = "l", tcl = -.25     )
>rect(-3, -3, 3, 3, col = "cornsilk")


>points(mat1, col = "blue", pch = 22, bg = "red")
<points(mat2, col = "red", pch = 25, bg = "yellow")
```

Deuxième exercice

Dans le package datasets est disponible le jeu de données iris.

Charger ce jeu de données en mémoire

```
>data(iris)
```

Quel est le mode de ce jeu de données ?

Ce jeu de données est une data.frame

Convertir le jeu de données iris en une matrice nommé matrice_iris

```
>matrice_iris = as.matrix(iris[ , 1:4])
```

Tapez la commande plot(matrice_iris). Que construit la fonction plot() dans le cadre des matrices!?

Dans le cadre de matrices la fonction plot() trace le graphe de la première colonne de la matrice sur la deuxième.

Construire le graphique composé de tous les graphiques bivariés du jeu de données Iris (à partir de matrice_iris). Ajouter une forme et une couleur aux points en fonction de l'espèce Ajouter un titre et un label.

```
>pairs(matrice_iris[ ,1:4], bg = c("red", "green3","blue")[as.numeric(iris[ , 5])],     pch =
c(21, 25, 24)[iris[ ,5]],     main = "Iris de Fisher",     labels = c("Longueur\nSepale",
"Largeur\nSepale",  "Longueur\nPetale",  "Largeur\nPetale" ))
```

4.2 Fonction plot() dans le cadre d'une data.frame

Troisième exercice

Charger en mémoire le jeu de données iris disponible dans le package datasets.

Ce jeu de données est une data.frame. Tapez la commande plot(iris). Que constatez-vous ?

On constate que dans le cadre des data.frame la fonction plot trace toutes les combinaisons possible de graphes bivariés.

4.2.1 Faire un graphique esthétique du jeu de données iris (couleur, titre, forme, labels, etc…)

Par exemple :

```
>plot(iris[ ,1:4],  bg = c("red", "green3","blue")[iris[ ,5]],  pch = c(21, 25, 24)[iris[ ,5]],
main = "Iris de Fisher",  labels =  c("Longueur\nSepale",  "Largeur\nSepale",
"Longueur\nPetale",  "Largeur\nPetale" ))
```

4.3 Utilisation de la fonction hist()

Quatrième exercice

Intéressons-nous aux liens entre le temps d'attente entre deux éruptions et la durée des éruptions pour le «old faithful geyser» du parc national du Yellowstone (Wyoming, EtatsUnis). Ce jeu de données nommé faithful est disponible dans le package datasets. Lorsqu'on tape la commande ? faithful, on obtient un descriptif du jeu de données.
Visualiser le jeu de données faithful par l'intermédiaire de la fonction plot.

Définissez un seuil critique d'attente au-delà duquel, la probabilité que la prochaine éruption soit longue, soit forte.

Posons le seuil critique arbitrairement à 63.

Construire un histogramme de la durée d'éruption. Ajouter un titre, labéliser l'axe des abscisses, colorer les barres de l'histogramme en vert, colorer les traits de l'histogramme en rouge et représenter l'histogramme en terme de fréquence plutôt qu'en termes d'effectifs.

```
>hist(faithful$eruptions, col = "green", border = "red", proba = TRUE,  main =
"histogramme du temps des  eruptions",  xlab = "le old faithful geyser")
```

Augmenter la taille du pas de l'histogramme à 20 et ajuster une loi normale à cet histogramme. Que remarque-t-on ?

```
>hist(faithful$eruptions, col = "green", border = "red", proba = TRUE, main =
"histogramme du temps des eruptions", xlab = "le old faithful geyser", breaks = 20)
>x = seq(from = 1.5, to = 5, length = 500)

>y = dnorm(x, mean = mean(faithful$eruptions),    sd=sd(faithful$eruptions))
>lines(x, y, col = "brown") mtext("Ajustement à une loi normale") L'ajustement à une loi
normale n'est pas adéquat.
```

Utilisation des estimateurs locaux de la densité

Superposer à l'histogramme les estimateurs locaux de la densité associés aux paramètres d'ajustement 0.1, 0.3, 0.8 et 1.

```
>hist(faithful$eruptions, col = "yellow", border = "green", proba = TRUE, main =
"histogramme du temps des  ruptions", xlab = "le old faithful geyser", breaks = 20 )
>lines(density(faithful$eruptions, adj = .1), type='l', col='red', lwd=1)
>lines(density(faithful$eruptions, adj = .3), type='l', col='green', lwd=1)
>lines(density(faithful$eruptions, adj = .8), type='l', col='blue', lwd=1)
>lines(density(faithful$eruptions, adj = 1), type='l', col='black', lwd=1)
```

On s'aperçoit que l'estimateur local de la densité associé à un coefficient d'ajustement de 0.1 dépasse de la fenêtre graphique.

Imposer donc des contraintes à la taille de l'axe des ordonnées.

```
>hist(faithful$eruptions, col = "yellow", border = "green", proba = TRUE, main =
"histogramme du temps des  ruptions", xlab = "le old faithful geyser", breaks = 20, ylim =
c(0, 1) ) lines(density(faithful$eruptions, adj = .1), type='l', col='red', lwd=1)
>lines(density(faithful$eruptions, adj = .3), type='l', col='green', lwd=1)
>lines(density(faithful$eruptions, adj = .8), type='l', col='blue', lwd=1)
>lines(density(faithful$eruptions, adj = 1), type='l', col='black', lwd=1)
```

4.4 Utilisation de la fonction boxplot ()

Partitionner votre graphique en deux dans le sens horizontal.

```
>vecteur = t(1:2)
>layout(vecteur)
```

Dans le premier cadre de la partition, tracer les boîtes à moustache associé aux durées d'éruption conditionnellement aux temps d'attente entre deux éruptions

```
>boxplot(faithful$eruption ~ faithful$waiting)
```

Dans le second cadre de la partition, utiliser le seuil calculé précédemment pour construire deux boites à moustache associé aux durées d'éruption conditionnellement aux temps d'attente entre deux éruptions. L'une codant le séisme court et l'autre codant le séisme long.

Pour ce faire créer une variable égale à court si waiting est inférieur au seuil et à long si waiting est supérieur au seuil.

```
>Y = character(272)
>Y[faithful$waiting > 63] = "long"
>Y[faithful$waiting <= 63] = "court"
>Y = as.factor(Y) boxplot(faithful$eruption ~ Y)
```

Décorer le graphique (Couleur, label, titre) et rajouter un indicateur de densité.

```
>boxplot(faithful$eruption ~ Y, col = c("yellow", "red"),  main = "boxplot en fonction de
la duree du seisme",  xlab = "court vs long" )
>rug(faithful$eruption, side = 2)
```

Utilisation de la fonction pie()

On souhaite visualiser, via les boîtes à moustache, la proportion de séisme dont la durée est supérieur à 3 minutes.

```
>Y = numeric(272)
>Y[faithful$eruption > 3] = 1
>Y[faithful$eruption <= 3] = 0
>Pourcentage_court = length(Y[Y == 0])/nrow(faithful)
>Pourcentage_long = length(Y[Y == 1])/nrow(faithful)
>Seisme = c(Pourcentage_court, Pourcentage_long) names(Seisme) = c("seisme long",
"seisme court") pie(Seisme, col = c("yellow", "red"), main = "duree des seismes", border =
NA)
```

TD5

1 Création, interrogation et destruction des objets.

R est un langage basé sur des objets (vector, matrix, data.frame, list, factor,...). Nous reviendrons sur cette notion dont la compréhension est nécessaire pour la bonne maîtrise de ce langage. Pour commencer, créons un objet simple de type "vecteur" qui contiendra une valeur numérique.

```
> x <- 15
> x
[1] 15
```

L'opérateur d'assignation"="peut se substituer `a l'operateur"<-"

```
> x = 22
> x
[1] 22
```

Cependant l'opérateur "<-" est généralement préféré car il permet d'avoir un code plus lisible (pour des raisons que nous expliciterons par la suite). Le code pourra être commenté à l'aide du caractère "#". Toute commande à la suite de ce caractère ne sera pas interprétée.

```
> #x <- 23
> x
[1] 22
```

Les instructions peuvent être séparées par un retour à la ligne ou par le caractère " ; ".

```
> x <- 12 ; y<- 13
```

Les objets créés sont stockés dans la mémoire vive de l'ordinateur (ils ne sont pas à ce stade, stockés sur le disque). On peut lister les objets disponibles en mémoire avec la fonction ls. Si vous quittez R ces objets seront détruits. Une autre méthode pour les détruire consiste à utiliser la fonction rm (remove).

```
> ls()
[1] "x" "y"
> rm(x) > rm(y)
> ls()
character(0)
```

2 Fonctions. Informations de base sur les objets

Nous venons de construire des vecteurs ("vector" pour R) contenant des données numériques. Nous avons, par ailleurs utilisé les fonctions ls et rm. Dans R, on peut appeler des fonctions pour interroger des objets et réaliser des actions à partir de ceux-ci. Les fonctions se présentent sous la forme suivante :

```
> nomdelafonction(arg1= a, arg2= b,...)
```

Arg1 et arg2 (...) correspondent aux noms des arguments tandis que a et b correspondent aux objets que l'on souhaite passer à la fonction pour qu'elle effectue une tâche.

Dans l'exemple suivant on utilise la fonction mean pour calculer une moyenne puis une moyenne tronquée (on élimine 10% des valeurs faibles et 10% des valeurs fortes, soit 20%).

Notez que les noms des arguments peuvent être abrégés voire omis. Dans ce dernier cas il faudra veiller à passer les arguments dans l'ordre.

```
> y <- c(-50, 1:10, 700)
> y
[1] -50 1 2 3 4 5 6 7 8 9 10 700
> is(y)
[1] "numeric" "vector"
> length(y)
[1] 12
> mean(y)
[1] 58.75
> mean(y, trim = 0.2)
[1] 5.5
> mean(y, tr = 0.2)
[1] 5.5
> mean(y, 0.2)
[1] 5.5
```

3 Les modes

Il existe quatre types de modes : "numérique", "caractère", "logique", "complexe". On peut stocker ces variables dans des vecteurs en utilisant la fonction c (combine). Attention, un vecteur n'accepte qu'un seul type de mode.

```
> noms = c("celine", "alain", "robert")
> noms
[1] "celine" "alain" "robert"
> is(noms)
[1] "character" "vector"
> length(noms)
[1] 3
```

```
> logic = c(TRUE, FALSE, TRUE)
> logic
[1] TRUE FALSE TRUE
> is(logic)
[1] "logical" "vector"
```

Les variables logiques peuvent s'écrire sous la forme TRUE et FALSE ou sous la forme T et F.

```
> logic = c(T, F, T)
```

4 Les objets dans R

Nous traiterons dans cette section des objets de type vector, factor matrix, list et data.frame.

R est particulièrement doué dans la gestion des valeurs numériques qui se présenteront le plus fréquemment sous la forme de vecteurs (une dimension) ou matrice (deux dimensions) de données.

4.1 Les objets de type vecteur

4.1.1 Création de vecteurs

Les fonctions c, rep et seq Comme nous l'avons vu, R peut stocker des données dans des objets de type vecteur. On peut créer des vecteurs à l'aide des fonctions c (pour combine), rep (pour repeat), seq (pour sequence) ou de l'opérateur " :". Comme nous l'avons souligné dans la section 3 le vecteur n'accepte qu'un type de mode.

```
> x <- 1:10 > x
[1] 1 2 3 4 5 6 7 8 9 10
> x <- c(2, 5, 7)
> x
[1] 2 5 7
> y <- rep(1, times = 5)
> y
[1] 1 1 1 1 1
> y <- rep(x, times = 3)
 > y
[1] 2 5 7 2 5 7 2 5 7
> x <- seq(from = 1, to = 20, by = 2)
> x
[1] 1 3 5 7 9 11 13 15 17 19
> x <- seq(from = 1, to = 20, length = 10)
> x
```

[1] 1.000000 3.111111 5.222222 7.333333 9.444444 11.555556 13.666667 [8] 15.777778 17.888889 20.000000

4.1.2 Création de vecteurs contenant des variables pseudo-aléatoires

Il est souvent très pratique en statistique de pouvoir générer des valeurs aléatoires. Les fonctions rnorm, runif et rpois, par exemple, permettent de générer des données aléatoires (random) selon une loi normale, uniforme ou de poisson, respectivement.

```
> x <- rnorm(10000, mean = 10, sd = 4)
> hist(x, col = "blue", breaks = 100)
> x <- runif(10000, min = 0, max = 10)
> hist(x, breaks = 20, col = "red")
```

Ici la fonction hist () est utilisée pour générer un histogramme de fréquences à partir de 10 000 valeurs aléatoires suivant une loi normale. Notez que l'argument breaks contrôle le nombre d'intervalles (axe des abscisses).

4.1.3 Création de vecteurs contenant des caractères alphabétiques

R contient des variables prédéfinies pour faciliter la vie de l'utilisateur. On peut les utiliser pour créer des vecteurs contenant des caractères alphabétiques :

```
> x <- letters [1:10]
> x
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> x <- LETTERS [1:10]
> x
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

4.1.4 Opérations de tri et de randomisation sur les vecteurs

On pourra faire des opérations simples de tri ou de randomisation sur les vecteurs.

Notez que, dans l'exemple suivant, quand l'argument replace de la fonction sample est positionné sur TRUE, le tirage aléatoire est effectué avec remise.

```
> x <- 1:10
> sort(x, decreasing = T)
[1] 10 9 8 7 6 5 4 3 2 1
> sample(x)
[1] 8 2 9 3 4 5 6 7 10 1
> sample(x, replace = T)
[1] 5 2 5 4 8 8 4 6 10 5
```

4.1.5 Indexation des vecteurs

Pour interroger les ´éléments d'un vecteur on utilisera (i) un vecteurs d'indices (ie; de positions), ou un vecteur logique ou les noms des éléments du vecteur.

Vecteur d'indices Il s'agit, dans ce cas, d'indiquer par une ou plusieurs valeur(s) numérique(s) les positions que l'on souhaite extraire dans le vecteur. On peut utiliser la fonction which qui renvoie les positions pour lesquels une condition est vérifiée.

```
> x <- c(-2, 4, -5, 1, 7)
> x[3]
[1] -5
> x[3:5]
[1] -5   1   7
> x[c(3, 5)]
[1] -5   7
> ind <- which(x > 0)
> ind
[1] 2 4 5
> x[ind]
[1] 4   1   7
```

On pourra aussi utiliser l'indexation de positionnement pour déliter les éléments d'un vecteur.

```
> x[-ind]
[1] -2 -5
```

Indexation par un vecteur logique

Le vecteur logique est une suite de booléens (TRUE ou FALSE). On peut le créer très facilement en utilisant des opérateurs de comparaisons (==, ! =,>, <, <= ,>=) qui renvoie TRUE ou FALSE. On peut combiner ces comparaisons avec des opérateurs logiques (&,|).

```
> x > 0
[1] FALSE TRUE FALSE TRUE TRUE
> x[x > 0]
[1] 4 1 7
> x > 0 & x < 5
[1] FALSE TRUE FALSE TRUE FALSE
> x[x > 0 & x < 5]
[1] 4 1
```

Ici, seules les valeurs de x pour lesquels la condition est vraie sont renvoyés. Il s'agit en fait d'une boucle implicite. Cette syntaxe est extrêmement économique en terme de code et, par conséquent, elle est très employée dans le langage R.

```
> y <- 1:5
> cbind(x, y)
      x  y
[1,] -2  1
[2,]  4  2
[3,] -5  3
[4,]  1  4
[5,]  7  5
> x[x > 0 & y > 3]
[1] 1  7
> x[x > 1 | y > 3]
[1] 4  1  7
```

4.1.6 Opération mathématique sur les vecteurs

On peut effectuer des opérations mathématiques sur les vecteurs `a l'aide d'opérateurs classiques : +, −, ∗. Pour élever à la puissance on utilise l'accent circonflexe. Ces opérations ne nécessitent pas d'utiliser des boucles "for" comme c'est le cas dans de nombreux langages et permettent donc d'effectuer en quelques lignes de code des opérations très complexes. C'est l'une des caractéristiques principales de R : la vectorisation. La structure vectorielle du langage rend les boucles implicites dans les expressions. Pour additionner deux vecteurs on écrira plutôt :

```
> x <- 2:5
> x
[1] 2 3 4 5
> y <- seq(1, 10, length = 4)
> y
[1] 1  4  7  10
> x + y
[1] 3 7 11 15
> x - y
[1] 1  -1  -3  -5
> x * y
[1] 2  12  28  50
> x^2
[1] 4  9  16  25
> x^0.5
[1] 1.414214  1.732051  2.000000  2.236068
```

4.2 Les facteurs

La fonction factor permet de créer des variables catégoriques `a partir de vecteurs de modes caractères ou numériques. On pourra en effet, choisir d'analyser les individus ou les variables selon leur appartenance à une catégorie donnée. La fonction levels permet de connaitre le nom des catégories que l'on pourra modifier comme dans l'exemple suivant.

```
> u <- sample(letters[1:3], size = 10, replace = T)
> u
[1] "b" "a" "c" "b" "b" "c" "a" "c" "a" "c"
> is(u)
[1] "character" "vector"
> u <- factor(u)
> levels(u)
[1] "a" "b" "c"
> levels(u) <- c("chat", "chien", "lapin")
> u
[1] chien chat lapin chien chien lapin chat lapin chat lapin
Levels : chat chien lapin
```

NB : La fonction gl est aussi fr´equemment utilis´ee pour g´en´erer des objets de type factor.

Les facteurs constituent aussi un outil très efficace pour transformer des variables de types caractères en variables numériques via la fonction as.numeric.

```
>as.numeric(u)
[1] 2 1 3 2 2 3 1 3 1 3 # Un vecteur (!= facteur)
```

4.3 Les matrices

C'est simplement un tableau à 2 dimensions. C'est un cas particulier de la structure array qui accepte k dimensions.

4.3.1 Création des matrices

On produira une matrice avec la fonction matrix. Tout comme le vecteur, une matrice n'accepte qu'un seul type **de mode**. L'argument data de la fonction matrix est un vecteur qui indique ce que l'on souhaite mettre dans la matrice, l'argument byrow indique si on range les données en lignes ou en colonnes et les arguments nrow et ncol permettent d'indiquer le nombre de lignes et de colonnes respectivement. Comme vous pouvez le constater dans l'exemple qui suit, cet objet contient des noms pour les lignes et des noms pour les colonnes.

```
> mat <- matrix(1:20, nc = 5, nr = 4, byrow = T)
> dim(mat)
```

```
[1] 4 5
> colnames(mat) <- LETTERS[1:5]
> row.names(mat) <- letters[1:4]
> mat

   A  B  C  D  E
a  1  2  3  4  5
b  6  7  8  9 10
c 11 12 13 14 15
d 16 17 18 19 20
```

Par ailleurs, on pourra agglomérer (bind) des lignes (rows) ou des colonnes avec les fonctions cbind et rbind pour générer des matrices.

```
> x <- cbind(1:10, 21:30)
> is(x)
[1] "matrix" "structure" "array" "vector" "vector"
> x
      [,1] [,2]
 [1,]    1   21
 [2,]    2   22
 [3,]    3   23
 [4,]    4   24
 [5,]    5   25
 [6,]    6   26
 [7,]    7   27
 [8,]    8   28
 [9,]    9   29
[10,]   10   30

> x <- rbind(1:10, 21:30)
> is(x)
[1] "matrix" "structure" "array" "vector" "vector"
> x
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    2    3    4    5    6    7    8    9    10
[2,]   21   22   23   24   25   26   27   28   29    30
```

4.3.2 Indexation des matrices

Pour l'indexation, on utilisera, comme dans le cas du vecteur, des vecteurs contenant des indices de positionnement ou des vecteurs logiques. Cependant, il faudra considérer une dimension supplémentaire. Le premier vecteur correspondra aux lignes, le deuxième aux colonnes.

```
> mat[1, ]
A B C D E
1 2 3 4 5
```

```
> mat[1:3, ]
   A  B  C  D  E
a  1  2  3  4  5
b  6  7  8  9 10
c 11 12 13 14 15
> mat[c(1, 3), ]
   A  B  C  D  E
A  1  2  3  4  5
c 11 12 13 14 15
> mat[-c(1, 3), ]
   A  B  C  D  E
B  6  7  8  9 10
d 16 17 18 19 20
> mat[, 1:3]
   A  B  C
a  1  2  3
b  6  7  8
c 11 12 13
d 16 17 18
> mat[1:3, 1:3]
   A  B  C
a  1  2  3
b  6  7  8
c 11 12 13
> ind <- mat[, 1] > 10
> ind
    a     b     c     d
FALSE FALSE  TRUE  TRUE
> mat[ind, ]
   A  B  C  D  E
c 11 12 13 14 15
d 16 17 18 19 20
```

4.4 Les listes

L'objet list, est l'objet le plus flexible des objets basiques de R. Au contraire de la matrice, il ne possède pas de contraintes de tailles (dans la matrice toutes les lignes et toutes les colonnes ont la même taille par définition). De même, il ne contient pas de contraintes de modes et on pourra même y stocker des objets très divers. Etant donné cette flexibilité, les fonctions renvoient souvent les résultats sous cette forme (exemple : une liste contenant des vecteurs, des matrices et autres objets divers et variés issus du calcul). Chaque ´élément de la liste possède un nom. Ces noms sont accessibles via la fonction names.

4.4.1 Création des listes

```
> L1 <- list(A = 1:5, B = matrix(8:13, nr = 2), C = LETTERS[10:15])
> L1
$A
[1] 1 2 3 4 5

$B
     [,1] [,2] [,3]
[1,]    8   10   12
[2,]    9   11   13
$C
[1] "J" "K" "L" "M" "N" "O"
> names(L1)
[1] "A" "B" "C"
```

4.4.2 Indexation des listes

Attention, dans le cas des listes, la fonction d'indexation ("[") renvoie une liste, alors qu'on souhaiterait généralement avoir accès au contenu (dans l'exemple, une vecteur numérique).

```
> L 1[1]
$A
[1] 1 2 3 4 5
> is (L1[1])
[1] "list" "vector"
```

Pour avoir accès au vecteur contenu dans la liste, il faut utiliser la fonction d'indexation spécifique des liste ("[[") et lui passer un numérique ou le nom d'un ´élément de la liste comme argument.

```
> L1[[1]]
[1] 1 2 3 4 5
> is(L1[[1]])
[1] "integer" "vector" "numeric"
> names(L1)
[1] "A" "B" "C"
> L1[["A"]]
[1] 1 2 3 4 5
```

On pourra, de plus utiliser l'op´erateur $ pour indexer la liste.

```
> L1$A
[1] 1 2 3 4 5
```

NB : On ne peut indexer qu'un seul élément de la liste `a la fois. C'est l`a une limite de cet objet.

4.5 L'objet data.frame

Le data.frame est un tableau de données à deux dimensions composé d'un ou plusieurs vecteurs ayant tous la même longueur mais pouvant être de modes différents. C'est un intermédiaire entre la liste et la matrice. Comme la matrice il peut posséder des noms de lignes et des noms de colonnes accessibles par les fonctions row.names et colnames. Ses dimensions peuvent être interrogées par la fonction dim (comme dans le cas des objets matrix).

4.5.1 Création des objets data.frame

On pourra cr´eer des objets data.frame en utilisant la fonction data.frame. Cette fonction prend comme arguments différents vecteurs qui constitueront les colonnes du data.frame. Attention, les vecteurs de caractères sont transformés par défaut en objet de type factor. On peut éviter cet effet en traitant les vecteurs de caractères à l'aide de la fonction I (voyez l'aide sur cette fonction si vous souhaitez en savoir plus).

```
> df <- data.frame(x = 10:1, y = 1:10, z = letters[10:19])
> df
   x  y  z
1  10 1  j
2  9  2  k
3  8  3  l
4  7  4  m
5  6  5  n
6  5  6  o
7  4  7  p
8  3  8  q
9  2  9  r
10 1  10 s
> is(df[, 3])
[1] "factor" "oldClass"e
> dim(df)
[1] 10 3
```

4.5.2 Indexation des objets data.frame

A nouveau, on se situe entre la liste et la matrice. Pour l'indexation, on peut proc´eder (i) comme pour les matrices, (ii) se r´ef´erer aux noms des colonnes en utilisant le caract`ere "$" (iii) ou encore utiliser la fonction d'indexation ("[[") (cf : la section 4.4.2 sur l'indexation des listes).

```
> df[, 1]
[1] 10 9 8 7 6 5 4 3 2 1
> df$x
22
[1] 10 9 8 7 6 5 4 3 2 1
> df[["x"]]
[1] 10 9 8 7 6 5 4 3 2 1
```