R

Université FUN 2016/2017

Introduction à la statistique avec R

Bruno Falissard, Christophe Lalanne

Table des matières

Première partie

Université FUN

Chapitre 1

Deuxième partie

Université de Nantes

Troisième partie

Représentations graphiques

Chapitre 2

Graphiques de base - Pr Jean R. Lobry

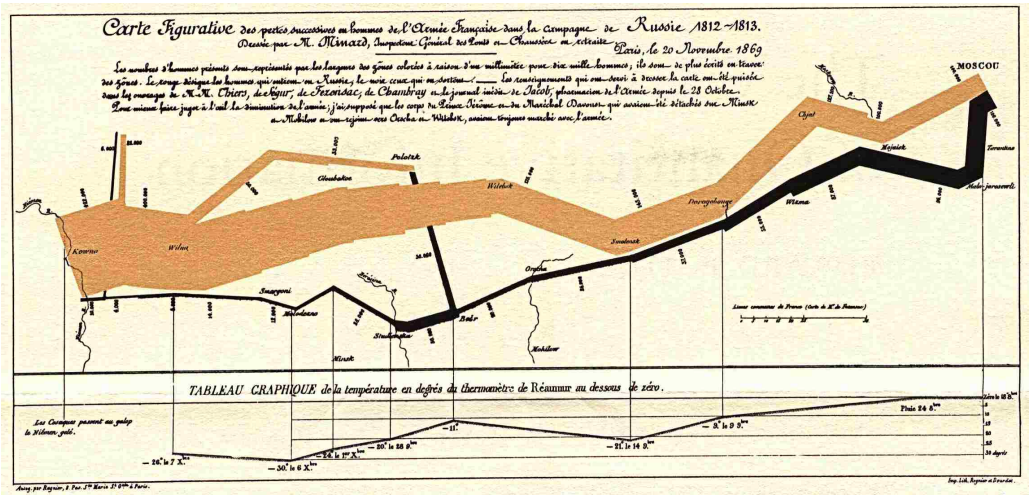
2.1 Introduction

2.1.1 Le graphique de Charles Minard

Un bon croquis vaut mieux qu'un long discours. - **Napoléon Bonaparte**

Charles Joseph Minard (né le 27 mars 1781 à Dijon, Côte-d'Or, et mort le 24 novembre 1870 à Bordeaux, Gironde) est un ingénieur civil français célèbre pour ses inventions dans le domaine de la traduction graphique et cartographique appliquée au génie civil et aux statistiques. Plus méconnus mais néanmoins réels sont sa réflexion et son apport sur l'utilité collective et son analyse de la tarification des équipements publics (péage).

Par une ironie de l'histoire, le graphique statistique considéré par beaucoup comme étant le meilleur jamais produit illustre la désastreuse campagne de Russie conduite par Napoléon en 1812.
Ce graphique est de l'ingéenieur français Charles Minard (1781-1870).
Le graphique représente le nombre de survivants de l'armée par l'épaisseur des bandes sur la carte de la campagne, à l'aller et au retour. La température pendant la retraite est indiquée au bas de la figure.
Carte figurative des pertes successives en hommes de l'armée française dans la campagne de Russie 1812-1813.



Exemple : Reprises du graphique de Charles Minard.

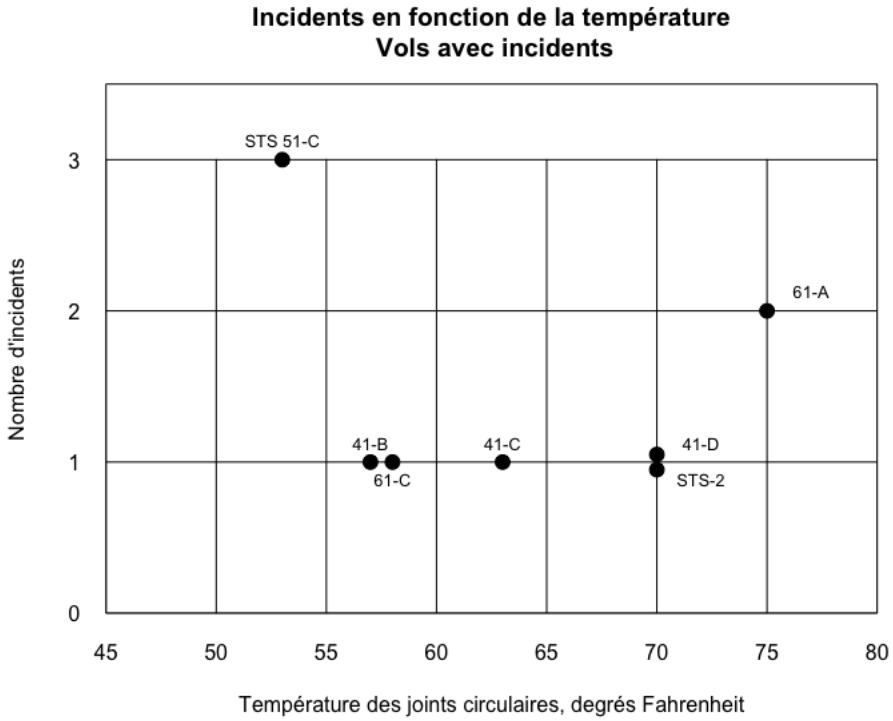
2.1.2 Importance des représentations graphiques

Un mauvais graphique peut avoir des conséquences catastrophiques comme par exemple, l'explosion de la navette spatiale Challenger

On peut (aussi) faire de mauvais graphiques sous R :

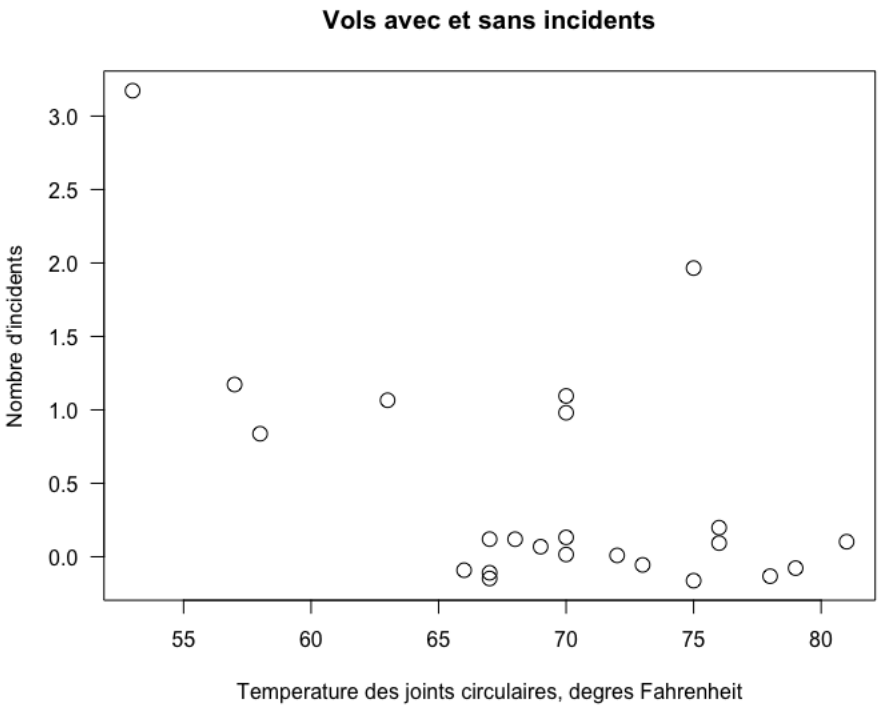
```
orf <- read.table("http://pbil.univ-lyon1.fr/R/donnees/ORingFailure.txt", header = TRUE)
str(orf)
'data.frame': 23 obs. of  2 variables:
 $ Temperature: int  66 70 69 68 67 72 73 70 57 63 ...
 $ Failures   : int  0 1 0 0 0 0 0 0 1 1 ...
  head(orf,3)
  Temperature Failures
1          66        0
2          70        1
3          69        0
```

```
 plot(
+   orf[orf$Failure > 0 & orf$Temperature != 70, ],
+   pch = 19,xlim = c(45,80),ylim = c(0,3.5),cex = 1.5,
+   las = 1,
+   xlab = "Température des joints circulaires, degrés Fahrenheit",
+   ylab = "Nombre d'incidents", xaxs = "i",yaxs = "i",xaxt = "n",yaxt = "n",
+   main = "Incidents en fonction de la température\n Vols avec incidents"
+   )
 points(c(70,70), c(0.95,1.05), pch = 19, cex = 1.5)
 axis(1, at = seq(45,80,by = 5), tick = FALSE)
 axis(2, at = 0:3, las = 1, tick = FALSE)
 abline(h = 0:3)
 for( i in seq(45,80,by = 5)) segments(i,0,i,3)
 text(53, 3, "STS 51-C", pos = 3, cex = 0.8)
 text(77, 2, "61-A", pos = 3, cex = 0.8)
 text(57, 1, "41-B", pos = 3, cex = 0.8)
 text(58, 1, "61-C", pos = 1, cex = 0.8)
 text(63, 1, "41-C", pos = 3, cex = 0.8)
 text(72, 1, "41-D", pos = 3, cex = 0.8)
 text(72, 1, "STS-2", pos = 1, cex = 0.8)
```



Les options par défaut des fonctions graphiques de **R** sont étudiées pour donner de bons résultats.

```
plot(orf$Temperature, jitter(orf$Failure), cex = 1.5,
     las = 1, xlab = "Temperature des joints circulaires, degres Fahrenheit",
     ylab = "Nombre d'incidents",
     main = "Vols avec et sans incidents")
```



2.1.3 Les grandes familles de fonctions graphiques

R est un très bon environnement pour produire de façon reproductible des graphiques statistiques de haute qualité. On peut classer les fonctions graphiques en plusieurs catégories :

— Les fonctions liées format de sortie des graphiques.
— Les fonction permettant d'interagir avec les graphiques.
— Les fonctions graphiques de bas niveau pour retoucher un graphique existant.
— Les fonctions graphiques de haut niveau.

Les fonctions de format de sortie des graphiques Il existe de nombreuses fonctions pour ouvrir un nouveau périphérique graphique (*e.g.* pdf(), jpeg(), postscipt(), x11(), png(), gnome(), quartz(), xfig(), bitmap(), pictex()). Elles ne sont pas toutes disponibles pour tous les systèmes d'exploitation.
Pour en savoir plus voir *?Devices*. Le dispositif utilisé par défaut est donné par *getOption("device")*.
Exemple d'utilisation pour sauvegarder un graphique dans un fichier au format PDF :

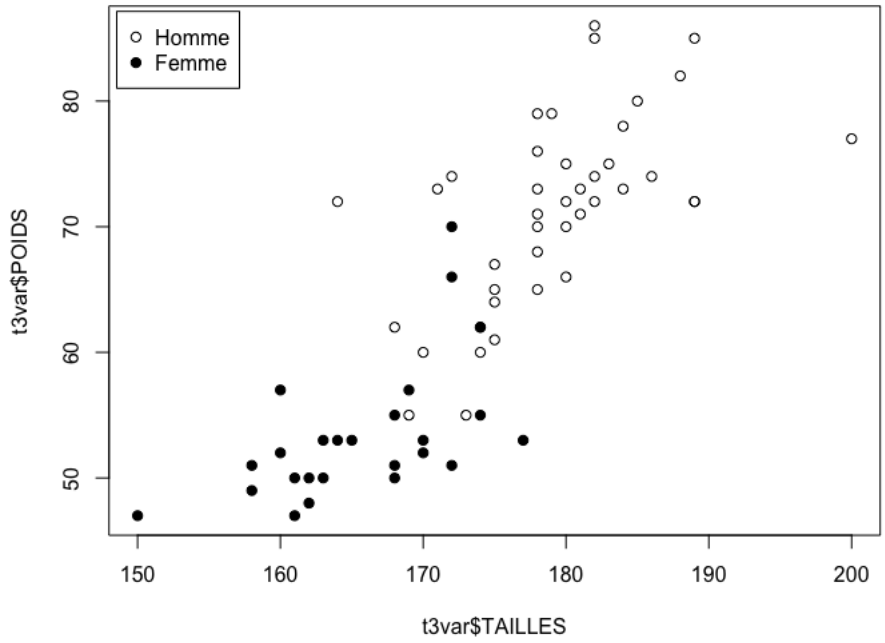
```
pdf("monfichier.pdf")
plot(0)
dev.off()
```

Les fonctions interactives Ces fonctions permettent de retoucher " à la main " un graphique, tout en conservant le résultat pour sa reproductibilité ultérieure.

— *locator()* permet de récupérer les coordonnées des points lorsque en cliquant dessus.
— *identify()* permet d'identifier des points. Donne le rang des points dans le jeu de données.

Les fonctions graphiques de bas niveau Ces fonctions permettent de retoucher un graphique déjà existant (*e.g.* points(), abline(), arrows(), lines(), seqments(), polygon(), rect(), box(), axis(), title(), rug(), grid(), legend(), text(), mtext()).
Par exemple pour ajouter une légende :

```
t3var <- read.csv2("data.csv",header = TRUE,sep="")
str(t3var)
'data.frame': 66 obs. of  3 variables:
 $ SEXE   : Factor w/ 2 levels "f","h": 2 1 1 1 1 1 1 2 1 2 ...
 $ POIDS  : int  60 57 51 55 50 50 48 72 52 64 ...
 $ TAILLES: int  170 169 172 174 168 161 162 189 160 175 ...
head(t3var,3)
  SEXE POIDS TAILLES
1    h     60     170
2    f     57     169
3    f     51     172
plot(t3var$TAILLES,t3var$POIDS, pch = ifelse(t3var$SEXE == "h", 1, 19))
legend("topleft", inset = 0.01, c("Homme","Femme"), pch = c(1, 19))
```



Les fonctions graphiques de haut niveau Ce sont celles que l'on utilise le plus souvent parce qu'elles donnent un graphique complet. Elles sont très nombreuses (*e.g.* plot(), hist(), dotchart(), stripchart(), pie(), barplot(), boxplot(), curve(), sunflowerplot(), symbols(), pairs(), stars(), assocplot(), mosaicplot(), coplot(), contour(), image(), persp()).
Nous allons envisager ci-après quelques fonctions graphiques de haut niveau trés utilisés en analyse exploratoire des données.

2.2 Variables numériques

2.2.1 Variables discrètes et variables continues

On parle également de variables **quantitatives**, elles sont représentées par une valeur numérique (numeric()).
On distingue parfois :

— les variables quantitatives **discrètes**, ne pouvant prendre qu'un nombre fini de valeurs (par exemple le nombre de jambes d'un individu).

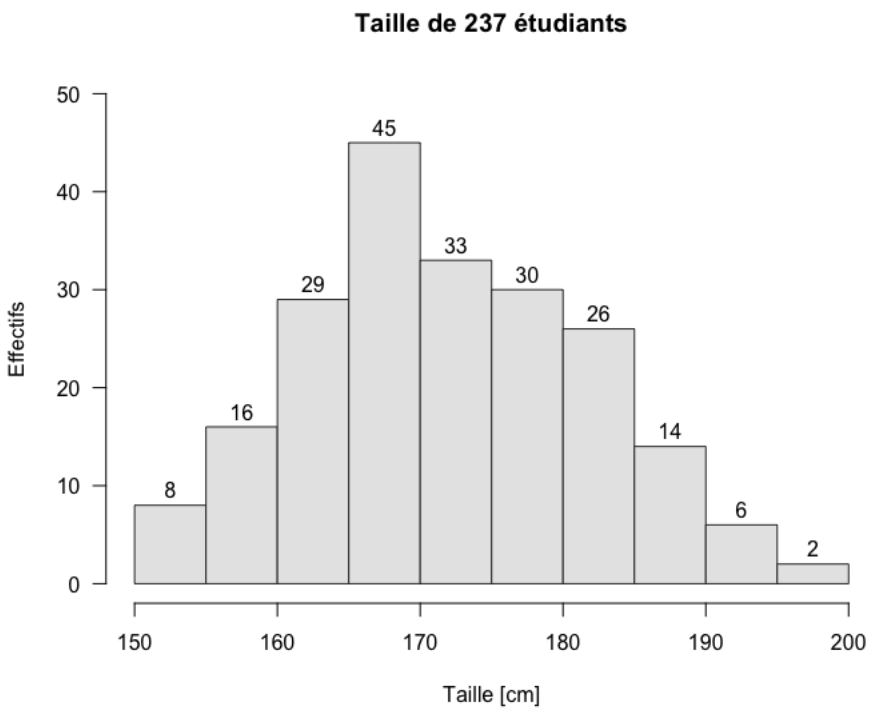
— les variables quantitatives **continues**, pouvant prendre un nombre infini de valeurs (par exemple la taille d'un individu).

Cette distinction est un peu artificielle puisque les variables continues *stricto sensu* n'existent pas à cause de la précision limitée des intruments de mesure.
Illustrons ce point.

Taille de 237 étudiants

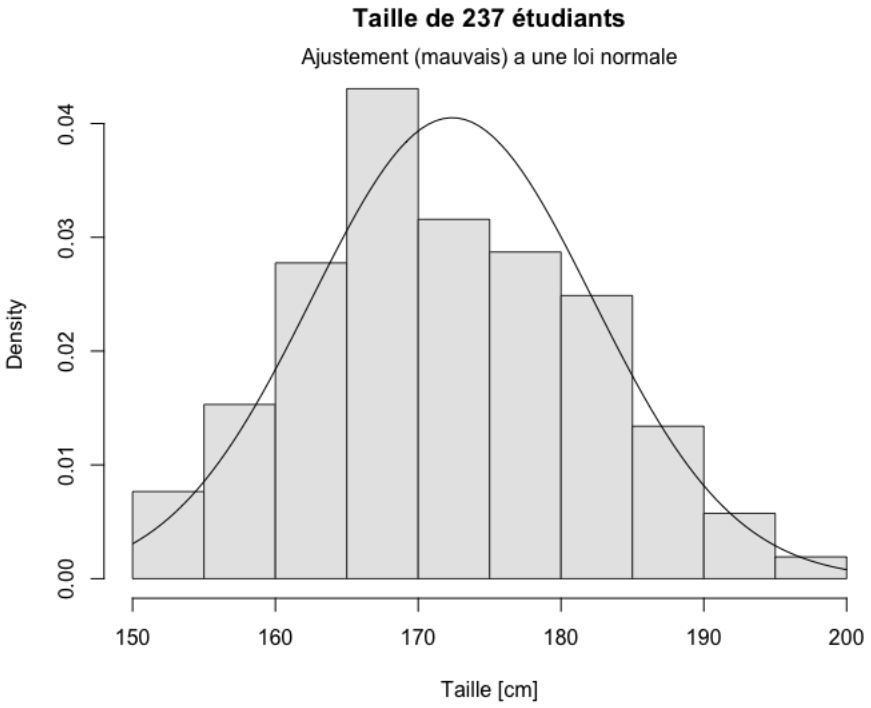
Intéressons nous à la taille de 237 étudiants disponibles dans le jeu de données *survey* de la bibliothèque *MASS*. Utilisons un histogramme pour représenter ces données.

```
library(MASS)
data(survey)
names(survey)
 [1] "Sex"    "Wr.Hnd" "NW.Hnd" "W.Hnd"  "Fold"   "Pulse"  "Clap"   "Exer"   "Smoke"
[10] "Height" "M.I"    "Age"
hist( survey$Height, col = grey(0.9), border = grey(0.2),
      main = paste("Taille de", nrow(survey), "étudiants"),
      xlab = "Taille [cm]",
      ylab = "Effectifs",
      labels = TRUE, las = 1, ylim = c(0, 50))
```



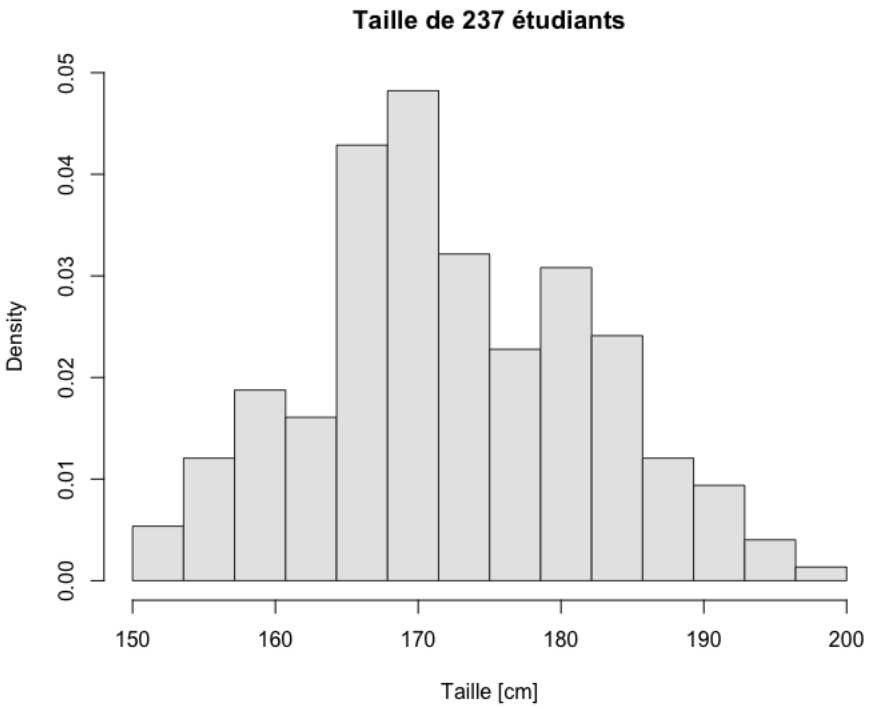
Nous avons utilisé ici des fréquences absolues, on préfère généralement utiliser des fréquences relatives *(proba = TRUE)* pour pouvoir superposer facilement des distributions de référence, par exemple :

```
hist( survey$Height, col = grey(0.9), border = grey(0.2),
      main = paste("Taille de", nrow(survey), "étudiants"),
      xlab = "Taille [cm]",
      proba = TRUE)
x <- seq(from = min(survey$Height, na.rm=T), to = max(survey$Height, na.rm=T),
         length = 100)
lines(x, dnorm(x, mean(survey$Height, na.rm = TRUE), sd(survey$Height,
                                                        na.rm = TRUE)))
mtext("Ajustement (mauvais) a une loi normale")
```



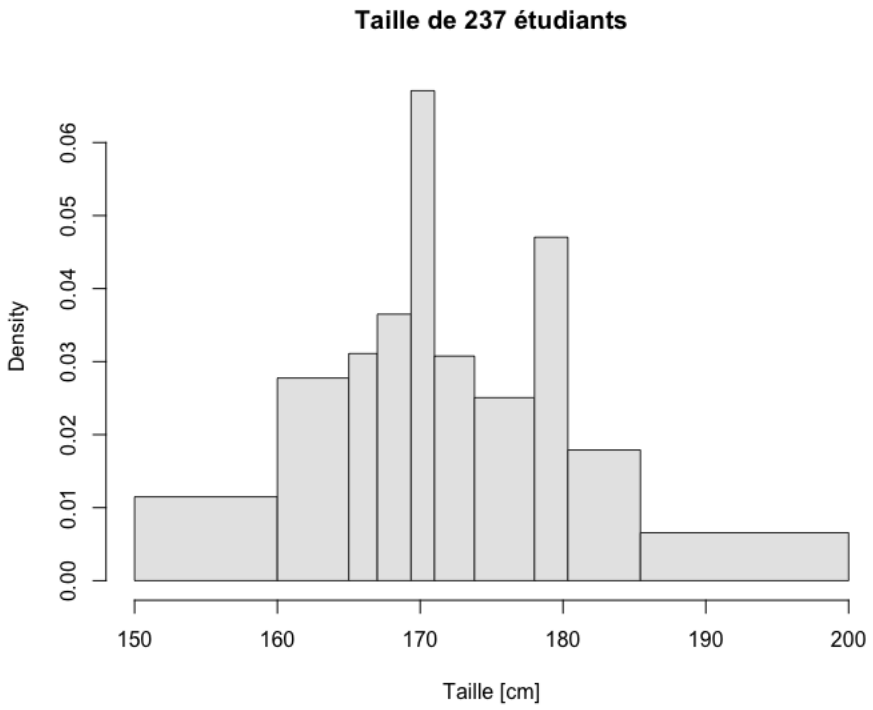
Le problème des histogrammes est que le choix du découpage en intervalles est assez arbitraire. On peut le contrôler avec le paramètre *break* de la fonction *hist()*, par exemple :

```
hist( survey$Height, col = grey(0.9), border = grey(0.2),
      main = paste("Taille de", nrow(survey), "étudiants"),
      xlab = "Taille [cm]",
      proba = TRUE, breaks = seq(from = 150, to = 200, length = 15))
```



On a plus l'impression dans ce dernier cas que la distribution est bimodale. Le choix du découpage en intervalle est un problème délicat qui risque de biaiser fortement notre perception des données. Avec des intervalles de même effectifs on aurait :

```
isohist <- function(x, nclass, ...){
  breaks <- quantile(x, seq(from = 0, to = 1, length = nclass + 1),
                     na.rm = TRUE)
  invisible(hist(x, breaks = breaks, ...))
}
isohist(survey$Height, 10, col = grey(0.9), border = grey(0.2),
        main = paste("Taille de", nrow(survey), "étudiants"),
        xlab = "Taille [cm]",
        proba = TRUE)
```



Chapitre 3

Sites internet

Site internet (source) : Aide à l'utilisation de R
Exemples de graphiques réalisés avec R : Représentations et scripts

Quatrième partie

Fonctions usuelles et aide mémoire

Chapitre 4

Commandes usuelles de R - Aide mémoire R

J. Chiquet - octobre 2015 & M. Kauffmann - mars 2009

Cette liste de commandes (non exhaustive) est une adaptation de la ref-card R de *Tom Short*. 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.

4.1 Aide et fonctions de base

help(topic) : ?topic affiche l'aide relative à topic ;

apropos("topic") : ? ?topic recherche par mot-clé (version courte)

help.search("topic") : ? ? ?topic recherche par mot-clé (version longue)

help.start() : lance la version HTML de l'aide

str(a) : affiche la structure de l'objet a

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

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

summary(a) : propose un «résumé» de a, la plupart du temps un résumé statistique.

search() : affiche l'itinéraire de recherche.

ls() : affiche tous les objets présents dans l'itinéraire de recherche.

ls.str() : applique str() à chaque variable présente dans l'itinéraire de recherche (liste de recherche).

dir() : affiche les fichiers présents dans le répertoire courant.

library(x) : charge la bibliothèque x.

attach(x) : place l'objet x dans l'itinéraire de recherche ; x peut être une liste, un tableau de données ou un objet créé à l'aide de la fonction save.

search() pour montrer la liste de recherche.

detach(x) : ôte l'objet x de l'itinéraire de recherche.

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 fonction.

if, while, repeat, etc. : voir help(if").

example(function) : exécute l'exemple donné en bas de la page d'aide de la function indiquée

<- 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)

methods(a) afficher les méthodes S3 [1] 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).

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

4.2 Entrées / Sorties

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 de la session

load(file) : charge un objet précédemment enregistré à l'aide de save.

data(x) : charge le jeu de données x.

read.table(file), read.csv, read.delim : lit un fichier stocké sous la forme d'un tableau et crée un objet data.frame ; le séparateur par défaut est le caractère espace pour read.table, la virgule ou le point virgule pour read.csv, la tabulation pour read.delim ; utiliser l'option header=TRUE pour que la première ligne soit considérée comme définissant le nom des colonnes. Plus en détails :

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.

cat(..., file="", sep=" ") : fonction d'impression bas niveau. 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.

1. S3 refers to a scheme of method dispatching. If you've used R for a while, you'll notice that there are print, predict and summary methods for a lot of different kinds of objects.
In S3, this works by :
— setting the class of objects of interest (*e.g.* : the return value of a call to method glm has class glm)
— providing a method with the general name (*e.g.* print), then a dot, and then the classname (*e.g.* : print.glm)
— some preparation has to have been done to this general name (print) for this to work, but if you're simply looking to conform yourself to existing method names, you don't need this (see the help I refered to earlier if you do).

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.

4.3 Variables réservées

NULL : l'objet nul (objet réservé).

NA : absense de données/valeur manquante.

TRUE/FALSE : vrai et faux logiques.

Inf : valeur infinie.

"abc" une chaîne de 3 caractères

4.4 Création de données

vector(mode, size) : initialise un vecteur de mode mode de taille size.

logical(size), numeric(size), double(size), character(size) : spécialisation de code aux modes élémentaires..

c(nom1=, nom2=, ...) : fonction générique combinant une suite d'éléments en un vecteur (possibilité d'attribuer des noms). 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)

from :to : génère une séquence ; priorité de l'opérateur « :» 1 :4 + 1 vaut 2,3,4,5.

seq(from,to) : génère une séquence ; by= et length= spécifient l'incrément et/ou la longueur..

seq_along, seq.intseq : variante de seq.

rep(x,times) : répète x un nombre times de fois ; utiliser each= pour répéter chaque élément each fois ; each peut être un vecteur.

rep.int, rep_len : variantes de rep. 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 tableau de données ; les vecteurs courts sont répétés jusqu'à correspondre à la taille des vecteurs les plus longs. 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) ;

vector(list, size) : crée une liste de taille size.

array(x,dim=) : crée un tableau multidimensionnel x ; les éléments de x sont répétés si la taille ne correspond pas aux dimensions spécifiées.

matrix(x,nrow=,ncol=) : crée une matrice ; les éléments de x sont répétés si la taille ne convient pas.

factor(x,levels=) : crée un vecteur de facteurs. Transforme un vecteur x en factor (les niveaux sont indiqués par levels=)

expand.grid() : génère un tableau de données contenant les combinaisons des vecteurs spécifiés en arguments.

rbind(),cbind() : pour combiner les éléments d'un objet par ligne et par colonne.

4.5 Extraction de données

4.5.1 Indexation des listes

x[n] : une liste avec les éléments de n. 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]] : le n^e élément de la liste.

x$name : l'élément "name".

x[["name"]] : l'élément "name".

4.5.2 Indexation des vecteurs

x[n] : n^e élément du vecteur.

x[-n] : tous les éléments sauf le n^e.

x[1 :n] : n premiers éléments.

x[-(1 :n)] : tous les éléments sauf les n premiers. Les éléments de $n+1$ à la fin.

x[c(1,4,2)] : éléments 1,4 et 2.

x["name"] : élément(s) de nom "name".

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

x[x > 3 & x < 5] : tous les éléments compris entre 3 et 5.

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

4.5.3 Indexation des matrices

x[i,j] : élément de la i^e ligne et j^e colonne.

x[i,] : i^e ligne.

x[,j] : j^e colonne.

x[,c(1,3)] : colonnes 1 et 3.

x["name",] : lignes intitulées "name".

x[rowSums(x)>10,] : lignes dont la somme est supérieure à 10.

4.5.4 Indexation des data.frame

Comme pour les matrices plus ce qui suit

x[["nom"]] : la colonne nommée "nom"

x$nom : la colonne nommée "nom"

4.6 Variables et attributs

as.array(x), 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)

is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.character(x), ... : teste le type d'un objet, renvoie TRUE ou FALSE ; pour une liste complète, faites methods(is)

Les fonctions suivantes s'utilisent pour récupérer ou spécifier un attribut.

length(x) : nombre d'éléments de x.

dim(x) : nombre de dimensions d'un objet. Récupère ou définit (dim(x) <- c(3,2)) les 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)/NROW(x), ncol(x)/NCOL(x) : nombre de lignes et de colonnes. NROW(x) et NCOL(x) considère un vecteur comme une matrice

class(x) : classe de l'objet x. Récupère ou définit la classe de x ; class(x) <- "maclasse"

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.

4.7 Manipulation et sélection de données

which.max(x),which.min(x) : retourne l'indice du plus grand (resp. plus petit) élément de x.

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

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

order() : renvoie une série d'indices permettant de permuter un tableau afin de le mettre dans l'ordre selon les valeurs de certaines colonnes ; *e.g.*,trier par ordre alphabétique de prénom le tableau suivant : x<-data.frame(prenom=c("Bernard","Charles", "Annie"),age=c(10,20,30)) ; x[order(x$prenom),].

cut(x,breaks) : découpe x en intervalles (factors) définis par breaks. breaks est le nombre de cas ou un vecteur de cloisons.

split(x,index) : renvoie une liste découpant x selon le facteur index.

unlist(l, recursive) : mise à plat de la liste l (récursivement par défaut).

match(x, y) : renvoie un vecteur de la même taille que x dont l'élément i vaut x[i] si x[i] appartient à y et NA sinon.

which(x==a) : renvoie les indices des éléments de x vérifiant x==a. renvoie les indices de x pour lesquels le résultat de l'opération logique est vrai (TRUE), dans cette exemple les valeurs de i pour lesquelles x[i]==a (l'argument de cette fonction doit être une variable de type « logique » (vrai ou faux)).

choose(n, k) : calcule les combinaisons de k éléments parmi n.

tabulate(x,nbin=length(x)) : compte les occurennces de tous les entiers jusqu'à nbin de x.

table(x) : généralisation de tabulate à des facteurs et tableaux de données. renvoie une table avec le décompte de chaque valeur différente de x ; table(x,y) renvoie un tableau de contingence

na.omit(x) : supprime les observations manquantes (notées NA). supprime les lignes correspondantes si x est une matrice ou un data.frame)

na.fail(x) : renvoie une erreur si x contient aumoins un NA.

any(x) : teste si x contient aumoins un élémént TRUE.

anyNA(x) : teste si x contientau moins un élémént NA.

unique(x) : supprime les doublons d'un vecteur ou d'un tableau. (pour un data.frame, ne renvoie que des lignes uniques)

table(x) : renvoie un tableau avec le nombre des différentes valeurs.

subset(x, ...) : renvoie un sous ensemble de x défini par levels(f), nlevels(f), is.ordered(f).

levels(f), nlevels(f), is.ordered(f) : manipulations des niveaux du facteur f .

sample(x, size) : crée un échantillon aléatoire de taille size parmi les éléments de x.

4.8 Mathématiques

abs, sqrt, sin, cos, tan, asin, acos, atan, atan2, log, log10, exp, %\%, %%, exp. . . : fonctions mathématiques élémentaires.

max(x), min(x), range(x), sum(x), diff(x), prod(x), mean(x), median(x), sd(x) : maximum (des éléments de x), minimum (des éléments de x), amplitude (c(mix(x),max(x))), somme (des éléments de x), différences (différence entre chaque élément de x et son prédécesseur), produit (des éléments de x), moyenne (des éléments de x), médiane (des éléments de x), écart-type (des éléments de x).

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

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

var(x), cov(x) : variance empirique corrigée ; si x est une matrice, renvoie la matrice de variance-covariance.

cor(x) : matrice de corrélations de x.

var(x, y), 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.

cor(x, y) : idem pour la corrélation linéaire.

rank(x) rang des éléments de x

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

floor(x), ceiling(x) : arrondissent à l'entier relatif supérieur ou inférieur.

scale(x) : centre et réduit les données x ; pour centrer et/ou réduire uniquement, utiliser les scale et/ou center.

pmin(x,y,...), pmax(x,y,...) : un vecteur dont le i^e élément est le minimum (resp. maximum) entre x[i] et y[i].

cumsum(x) : un vecteur dont le i^e élément est la somme des i premiers éléments de x.

cumprod(x), cummin(x), cummax(x) : idem pour le produit, le min, le max.

union(x,y), intersect(x,y), setdiff(x,y), setequal(x,y), all.equal(x,y) et is.element(el,set) : fonctions de définition d'ensembles. setdiff(x,y) trouve les éléments de x qui ne sont pas dans y

Re(x), Im(x), Mod(x),Arg(x), Conj(x) : partie réelle, partie imaginaire, module, argument et conjugué d'un nombre complexe.

convolve(x,y) : calcule de convolution [2] entre deux séquences.

fft(x), mvfft(x) : transformation de Fourier d'une matrice,resp des colonnes d'une matrice.

filter(x,filter) : application d'un filtre linéaire à chaque élément d'une suite x. 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

4.9 Matrices

rowSums(x), colSums(x), rowMeans(x), colMeans(x) : somme et moyenne de chaque ligne, resp. chaque colonne de x.

t(x) : transposée de x.

diag(x) : renvoie ou spécifie la diagonale de x.

upper.tri(A), lower.tri(A) : selection du triangle supérieur/inférieur de A.

%*% : multiplication matricielle.

crossprod(x,y), t(x)%*% y : produit scalaire de x par y.

det(x) : déterminant de x.

svd(x) : décomposition en valeurs singulières.

eigen(x) : diagonalisation d'une matrice.

chol(x) : décomposition de Cholesky [3].

---

2. Le produit de convolution de deux fonctions réelles ou complexes f et g, est une autre fonction, qui se note généralement $f * g$ et qui est définie par :

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(x-t)g(t)\,\mathrm{d}t = \int_{-\infty}^{+\infty} f(t)g(x-t)\,\mathrm{d}t$$

ou encore, pour des suites (en remplaçant la mesure de Lebesgue par la mesure de comptage) :

$$(f * g)(n) = \sum_{m=-\infty}^{\infty} f(n-m)g(m) = \sum_{m=-\infty}^{\infty} f(m)g(n-m)$$

(mais dans ce qui suit, nous n'utiliserons que la version continue).
On peut considérer cette formule comme une généralisation de l'idée de moyenne mobile.
Pour que cette définition ait un sens, il faut que f et g satisfassent certaines hypothèses ; par exemple, si ces deux fonctions sont intégrables au sens de Lebesgue (c'est-à-dire qu'elles sont mesurables et que l'intégrale de leur module est finie), leur produit de convolution est défini pour presque tout x et est lui-même intégrable.

3. **Factorisation de Cholesky d'une matrice :** Si A est une matrice symétrique définie positive, il existe une matrice réelle triangulaire inférieure L telle que :

$$A = L \times L^T$$

On peut également imposer que les éléments diagonaux de la matrice L soient tous positifs, et la factorisation correspondante est alors unique.
Par exemple : La matrice symétrique A est égale au produit de la matrice triangulaire L avec sa transposée L^T :

$$A = L \times L^T \Leftrightarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 \\ 1 & 5 & 14 & 14 \\ 1 & 5 & 14 & 15 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

qr(x) : décomposition QR[4].

solve(a,b) : résout a % * % x = b.

solve(a) : calcule l'inverse de a.

chol2inv(x) : Inversion à partir d'une décomposition de Cholesky.

Quelques fonctions du package Matrix

Matrix(x, sparse=) : définition d'un objet de classe matrice.

sparseMatrix(i, j, p, x=) : définition d'un matrice creuse.

bdiag(...) : création d'une matrice diagonal par blocs à partir d'une série de matrices.

bandSparse(...) : matrice creuse définie par ses termes super/sous diagonales.

Diagonal(n, x=) : création d'une matrice diagonale creuse.

Les objets de type Matrix possèdent les méthodes associées aux factorisations et décomposition usuelles (SVD, Cholesky, QR)

4.10 Traitements avancés

apply(x,INDEX,FUN=) : renvoie un vecteur ou une liste de valeurs obtenues en appliquant la fonction FUN aux éléments de la dimension INDEX de x. apply(X,MARGIN,FUN=, ...) applique une fonction FUN aux marges de X (MARGIN=1 pour les lignes, MARGIN=2 pour les colonnes) ; les paramètres ... sont passés à la fonction FUN.

lapply(x,FUN) : applique FUN aux éléments d'une liste. lapply(X,FUN) applique une fonction FUN à chaque élément de X

sapply(x,FUN) : applique FUN aux éléments d'une liste et simplifie la sortie.

tapply(x,INDEX,FUN=) : applique FUN à chaque groupe du tableau X défini par les indices INDEX.

rowsum(x,INDEX) : spécialisation de tapply pour la fonction sum (très performant).

by(data,INDEX,FUN) : applique FUN au tableau de données data découpé via INDEX.

ave(x,...,FUN) : applique FUN à chaque sous-ensemble de x définis par des facteurs.

merge(a,b) : fusion de deux tableaux de données portant les mêmes noms de ligne ou de colonne. merge(x,y) fusionne 2 data frames en utilisant leurs noms de colonnes en commun (ou en les désignant avec by.x et by.y)

aggregate(x,by,FUN) : découpe le tableau x en sous ensembles auxquels sont appliqués la fonction FUN et renvoie le résultat ; by est une liste définissant les sous-ensembles de x. aggregate(x,by,FUN) divise le data frame x en groupes, calcule la fonction FUN pour chacun ; by est une liste d'éléments de regroupement, chaque élément aussi long que le nombre de ligne de x

stack(x, ...), unstack(x, ...) : transforme un tableau ou une liste x en un vecteur colonne, et réciproquement. transforme un tableau en plusieurs colonnes en tableau à 1 colonne, en indiquant d'où vient chaque valeur ; *e.g.* : stack(data.frame(a=1 :3,b=4 :6)).

unstack(x, ...) inverse de stack().

4. En algèbre linéaire, la **décomposition QR** (appelée aussi, factorisation QR ou décomposition QU) d'une matrice A est une décomposition de la forme

$$A = QR$$

où Q est une matrice orthogonale ($Q^TQ = I$), et R une matrice triangulaire supérieure.
Ce type de décomposition est souvent utilisée pour le calcul de solutions de systèmes linéaires non carrés, notamment pour déterminer la pseudo-inverse d'une matrice.

combn(x, m, func) : applique la fonction func à toutes les combinaisons de m éléments parmi les élément de x.

replicate(n, expr, ...) : repète une opération faisant intervenir de l'aléa et renvoie un tableau multidimensionnel résultant de ces opérations.

do.call(func, list) : appelle la fonction func qu'elle applique aux arguments défini par les élements de list.

reshape(x, ...) fonction avancée (et compliquée) réorganisant en largeur ou en longueur une data.frame (*e.g.* : un tableau de 2 variables avec 3 années pour 4 pays contient 24 données, organisées en 2x12 ou 6x4 8x3 ; reshape convertit entre ces formats)

4.11 Chaînes de caractères

paste(...) : concaténation de vecteurs après conversion en caractères. sep= les sépare (par défaut : espace)

substr(x,start,stop) : extraction ou spécification d'une sous-chaîne de x.

strsplit(x,split) : découpe x selon la sous-chaîne "split".

grep(pattern,x) : recherche le motif pattern dans la chaîne x. renvoie les indices des éléments de x dans lesquels on trouve le patron pattern, *e.g.* : grep("b", c("ab", "cd", "bz"))

tolower(x), toupper(x) : conversion en minuscules, resp. en majuscules.

match(x,table) : un vecteur renvoyant les positions où les éléments de x ont été pour la première fois rencontrés dans table. pour chaque élément de x, renvoie NA si l'élément n'est pas trouvé dans table, sinon renvoie la position où il se trouve dans table

x % in% table : identique, mais renvoie un vecteur de booléens. pour chaque élément de x, renvoie TRUE si l'élément est trouvé dans table, sinon renvoie FALSE

pmatch(x,table) : appariement partiel des éléments de x parmi table.

nchar(x) : nombre de caractères de x.

4.12 Dates et heures

La classe Date enregistre des dates. POSIXct enregistre date, heure et fuseau horaire. Les comparaisons ($>$, $<$ ...), seq() ence, et écart de temps (difftime()) sont utiles. On peut enlever ou ajouter des jours à un objet Date (+, -).

as.Date(x) convertit une chaîne de caractères en date ; as.Date("2009-12-31")+1 renvoie le 1er janvier 2010.

format(x) l'inverse ; on peut choisir la représentation voulue (cf. help(strftime))

4.13 Graphiques et figures

x11(), windows() : ouvre une nouvelle fenêtre graphique x11 sous linux/mac, windows sur windows.

pdf() , png(), jpeg(), bitmap(), xfig(), pictex(), postscript() : pilote graphique produisant des sorties dans des fichiers plutôt qu'à l'écran. 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 le pilote de sortie graphique pour clore le fichier de sortie. cf. aussi dev.cur, dev.set

4.13.1 Commandes graphiques haut niveau

plot(x) : trace les valeurs contenues dans x sur l'axe des y ; s'adapte à la classe de l'objet x.

plot(x, y) : graphe bivarié (x sur l'axe des x, y sur l'axe des y) - nuage de points.

hist(x) : histogramme des fréquences de x.

barplot(x) : histogramme des valeurs de x - diagramme en barre

curve(expr) : trace la fonction définie par l'expression expr.

dotchart(x) : si x est un tableau de données, trace les données par nuages de points groupés par ligne en ordonnées puis par colonne en abscisses.

pie(x) : graphe en camembert.

boxplot(x) : boîte à moustaches. diagramme en boîte ; 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)

interaction.plot (f1, f2, y) : si f1 et f2 sont des facteurs, trace les moyennes de y en fonction des valeurs de f1 et f2 sur deux courbes différentes.

matplot(x,y) : graphe bivarié traçant la première colonne de x vs. la première colonne de y, puis la deuxième colonne de x vs. la deuxième colonne de y, etc..

assocplot(x) : graphe d'association indiquant à quelle point les colonnes et lignes du tableau de contingence x dévient de l'hypothèse d'indépendance.

mosaicplot() : graphe mosaïque des résidus d'un modèle loglinéaire.

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

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.

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

pairs(x) : trace tous les graphes bivariés possibles entre les colonnes du tableau de données x.

plot.ts(x) : si x est de classe "ts" (time-serie), trace x en fonction du temps.

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

qqnorm(x) : fractiles de x en fonction des valeurs attendues sous l'hypothèse gaussienne. 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) : fractiles de y en fonction des fractiles de x. De y en fonction des quantiles de x

contour(x, y, z), image(x, y, z), persp(x, y, z) : variantes pour tracer les données de la matrice z en fonction des vecteurs x et y.

symbols(x, y, ...) : trace aux coordonnées spécifiées par x et y des cercles, carrés, rectangles, étoiles, boîtés àmoustaches, etc..

termplot(mod.obj) : trace les termes d'un modèle de régression en fonction des prédicteurs .

4.13.2 Paramètres récurrents des fonctions graphiques

add=FALSE : si TRUE superpose le graphe au précédent.

axes=TRUE : si FALSE ne trace pas d'axes.

type="p" : spécifie le type de tracé : "p" pour points, "l" pour lignes, "b" pour points liés par des lignes, "o" pour lignes superposées aux points, "h" pour lignes verticales, "s" ou "S" pour fonction en escaliers.

xlim=, ylim= : spécifie les limites des axes x et y.

xlab=, ylab= : annotation des axes x et y.

main= : titre du graphe en cours.

sub= : sous-titre du graphe en cours.

4.13.3 Commandes graphiques bas-niveau

points(x, y) : ajoute des points aux coordonnées x et y (type= peut être utilisé).

lines(x, y) : trace y en fonction de x.

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

rug(x) : ajoute les occurrences des points en abscisses. ajoute près de l'axe des abscisses une petite barre pour chaque valeur de x

mtext : (text, side=3, ...) ajoute le texte text dans la marge side.

segments(x0, y0, x1, y1) : trace des lignes des points (x0,y0) aux points (x1,y1).

arrows(x0, y0, x1, y1) : identique mais avec des flèches.

abline(a,b) : trace une droite de pente b et de décalage a par rapport à l'axe des x. $y = ax + b$

abline(h=y) : trace une ligne horizontale à l'ordonnée y.

abline(v=x) : trace une ligne verticale à l'abscisse x.

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

rect(x1, y1, x2, y2) : trace un rectangle défini par x1, x2, y1 et y2.

polygon(x, y) : trace un polygone en liant les points de coordonnées définies dans les vecteurs x et y.

legend(x, y, legend) : ajoute une légende au point (x,y) spécifié par legend.

title() : ajoute un titre et éventuellement un sous-titre.

axis(side) : fonction de bas niveau pour gérer les axes de la figure. 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() : trace un cadre autour de la figure courante.

locator(n) renvoie les coordonnées des clics de la souris après n clics sur le graphique

4.13.4 Paramètres graphiques de bas de niveau

Ces paramètres sont définis à l'aide de la commande par(...) ou directement par passage à la fonction graphique d'appel

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)

adj : contrôle la justification du texte.

bg : spécifie la couleur de fond.

bty : contrôle le type de cadre tracé autour de la figure.

cex : contrôle la taille du texte et des symboles.

col : contrôle la couleur des symboles et des courbes (entier ou chaîne de caractères). pour créer des vecteurs de 5 couleurs, faire suivre col= de gray(0 :5/5), rainbow(5) ou terrain.colors(5)

font : un entier contrôlant le style de la police.

las : un entier contrôlant l'orientation des annotations des axes.

lty : contrôle le type de ligne (entier ou chaîne de caractère).

lwd : dcontrôle l'épaisseur des lignes.

mar : contrôle l'espace entre le tracé et les bordures de la fenêtre.

mfc : un vecteur de la forme c(nr,nc) qui partitionne la fenêtre graphique en nr lignes et nc colonnes, les graphes étant tracés par colonne.

mfrow : identique mais les graphes sont tracés par ligne.

pch : contrôle le type de symbole.

ps : un entier contrôlant la taille du texte et des symboles.

4.13.5 Groupes de graphiques conditionnels

Pour accéder à ces fonctions, il faut faire avant *library(lattice)*
La formule $y \sim x$ trace y en fonction de x. On peut faire un graphique $y \sim x$ par sous groupe de données en indiquant l'appartenance à tel ou tel groupe par le vecteur $g1$: $y \sim x|g1$;
pour toutes les combinaisons des séries de groupes $g1$ et $g2$: $y \sim x|g1 * g2$

xyplot(y∼x) : nuages de points

barchart(y∼x) : diagrammes en barre

histogram(∼x) : histogrammes

bwplot(y∼x) : boîtes à moustache

stripplot(y∼x) : graphique à une dimension, x doit être un nombre, y peut être un facteur

4.14 Statistiques

4.14.1 Distributions

Toutes les fonctions suivantes peuvent s'utiliser en remplaçant la lettre r avec d, p ou q pour obtenir, respectivement, un tirage de n réalisations d'une variable aléatoire, la densité de probabilité, la fonction de répartition, et la valeur des fractiles.

rnorm(n, mean=0, sd=1) : gaussienne.

rexp(n, rate=1) : exponentielle.

rgamma(n, shape, scale=1) : Gamma.

rpois(n, lambda) : Poisson.

rweibull(n, shape, scale=1) : Weibull.

rcauchy(n, location=0, scale=1) : Cauchy.

rbeta(n, shape1, shape2) : Beta.

rt(n, df) : Student.

rf(n, df1, df2) : Fisher-Snedecor (F).

rchisq(n, df) : Pearson (χ^2).

rbinom(n, size, prob) : binomiale.

rgeom(n, prob) : geometrique.

rhyper(nn, m, n, k) : hypergeometrique.

rlogis(n, location=0, scale=1) : logistique.

rlnorm(n, meanlog=0, sdlog=1) : lognormale.

rnbinom(n, size, prob) : binomiale negative.

runif(n, min=0, max=1) : uniforme.

rwilcox(nn, m, n), rsignrank(nn, n) : Statistique de Wilcoxon.

4.14.2 Modèles

Toutes les fonctions suivantes peuvent s'utiliser en remplaçant la lettre r avec d, p ou q pour obtenir, respectivement, un tirage de n réalisations d'une variable aléatoire, la densité de probabilité, la fonction de répartition, et la valeur des fractiles.

density(x) : estimateur à noyaux de la densité de x.

lm(formula) : ajuste un modèle linéaire ; formula est typiquement de la forme response $\sim termA + termB + \ldots$. formula=$y \sim a + b$ estime le modèle $y = ax + by + c$ (mettre - 1 dans la formule pour enlever la constante c) ; summary(lm(...)) donne des informations utiles

glm(formula,family=) : ajuste un modèle linéaire généralisé. *e.g.* family= binomial(link = "logit") pour un modèle logit (cf. ?family). Après la formule, on peut en général préciser le nom du data.frame (data=) et le sous-ensemble de données (subset=suivi d'un vecteur de valeurs logiques)

nls(formula) : estimateur non-linéaire des moindres carrés des paramètres d'unmodèle non-linéaire.

Les fonctions ci-dessus renvoient un objet modèle dont l'ajustement dépend de la méthode utilisée. Certains des attributs de cet objet sont évalués à l'aide des commandes suivantes.

aov, anova : fonction d'analyse de la variance.

df.residual(fit) : renvoie le nombre de degrés de liberté résiduels de fit.

coef(fit) : renvoie les coefficients estimés de fit.

residuals(fit) : renvoie les résidus du modèle fit.

predict(fit, ...) : prédiction à partir d'un modèle ajusté fit. Calcule également les intervalles de confiance et de prédiction.

fitted(fit) : retourne les valeurs prédites par le modèle.

logLik(fit) : calcule la log-vraisemblance du modèle et le nombre de paramètres.

AIC(fit) : calcule le critère AIC (Akaike information criterion).

Quelques fonctions liées au modèle linéaire.

step : régression stepwise sur critère AIC/BIC.

regsubsets : du package leaps, régression exhaustive.

rstandard(fit), rstudent(fit) : résidus standardisé ou studentisé associés à un modèle.

cooks.distance(fit) : calcul de la distance de cook.

lm.influence(fit) : diverse fonctions d'influence.

4.14.3 Tests

Toutes les fonctions suivantes peuvent s'utiliser en remplaçant la lettre r avec d, p ou q pour obtenir, respectivement, un tirage de n réalisations d'une variable aléatoire, la densité de probabilité, la fonction de répartition, et la valeur des fractiles.

t.test(x,y=) : test de Student pour une ou deux population.

pairwise.t.test : test de Student apparié.

power.t.test : calculs de puissance associée à un test de Student.

chisq.test : test du χ^2 de contingence ou d'adéquation.

var.test : test de Fisher d'égalité des variance.

fisher.test : test exact de Fisher d'indépendance.

ks.test : test de Kolmogorov-Smirnov d'adéquation, une ou deux populations.

shapiro.test : test de normalité de Shapiro-Wilk.

binom.test : test du paramètre d'une loi binomiale.

prop.test : test d'égalité de proportion.

Utiliser help.search("test") pour voir l'ensemble des tests statistiques disponibles

4.15 Optimisation

optimize(fn,interval) : méthode d'optimisation pour les fonctions unidimensionnelles.

optim(par, fn) : méthode d'optimisation générique minimisant la fonction fn en partant de la valeur par des coefficients.

nlm(f,p) : minimise la fonction f à l'aide d'un algorithme type

approx(x,y=) : interpolation linéaire.

spline(x,y=) : interpolation par splines[5] cubiques.

5. Une **courbe spline** est une fonction polynomiale par morceaux définie sur un intervalle $[a, b]$ divisé en sous intervalles $[t_{i-1}, t_i]$ tels que :

$$a = t_0 < t_1 < \cdots < t_{k-1} < t_k = b$$

on la note donc $S : [a, b] \to \mathbb{R}$.
Sur chaque intervalle $[t_{i-1}, t_i]$ on définit un polynôme :

$$P_i : [t_{i-1}, t_i] \to \mathbb{R}$$

, Cela nous donne, pour une spline à k intervalles :

$$\begin{aligned} S(t) &= P_1(t) \ , \ t_0 \leq t < t_1, \\ S(t) &= P_2(t) \ , \ t_1 \leq t < t_2, \\ &\vdots \\ S(t) &= P_k(t) \ , \ t_{k-1} \leq t \leq t_k. \end{aligned}$$

Le degré de la spline est défini comme celui du polynôme P_i de plus haut degré. Si tous les polynômes ont le même degré, on dit que la spline est uniforme. Dans le cas contraire, elle est non uniforme.

Continuité : Sachant que la dérivabilité d'un polynôme est infinie, la dérivabilité d'une spline dépend de la continuité au niveau de la jointure des courbes polynômes.
Si pour tout i $0 < i < k$ et pour tout j tel que $0 \leq j \leq n$ l'égalité suivante est vérifiée :

$$P_i^{(j)}(t_i) = P_{i+1}^{(j)}(t_i)$$

Alors la spline est de continuité n, notée C_n.
La continuité définit les caractéristiques de la jonction entre chaque intervalle. Cela correspond au degré de correspondance entre deux polynômes successifs aux points de jonction.

4.16 Programmation

Fonctions permettant d'enchaîner des opérations de manière structurée. Pour avoir de l'aide sur ces fonctions, saisir leur nom entre guillemets ; *e.g.* help("if").

function(*arglist* **)** {*expr*}

— arglist est une liste d'arguments,
— expr est une expression exécutée ;
e.g. : **mafonction** <-function(a, b) a+2*b ; **mafonction(1,2)** renvoie 5.

return(value) mis dans expr lors d'une définition de fonction, indique que la fonction doit renvoyer ce résultat(si return est absent, la fonction renvoie la dernière valeur calculée dans expr)

if(cond) {expr} si cond est vrai (TRUE), évaluer expr

== != < > <= >= opérateurs de comparaison, dans l'ordre : égal, différent, inférieur, supérieur, inférieur ou égal, supérieur ou égal ; *e.g.* 1==1 vaut TRUE ou T ; 1 !=1 vaut FALSE ou F ; dans les opérations avec des nombres, T est converti en 1 et F en 0 (T-1==0 est vrai)

if(cond) {cons.expr} else {alt.expr} si cond est vrai évaluer cons.expr sinon évaluer alt.expr

for(var in seq) {expr} exécute l'expression pour chaque valeur de var prises dans une sequence

while(cond) {expr} exécute l'expression tant que la condition est vraie

repeat {expr} répète expr en boucle ; penser à l'arrêter avec if(...) {break} (ou avec les touches Ctrl+C)

break arrête une boucle for, while ou repeat

next arrête l'itération en cours et reprend la boucle (dans le cas de for, avec la valeur suivante de la sequence)

ifelse(test, yes, no) pour chaque ligne/cellule de test, renvoie la valeur yes si le test

---

— C_0 est la continuité minimum : les polynômes successifs passent bien par les points de jonction.
— C_1 indique une continuité des tangentes : les polynômes successifs ont des dérivées premières égales aux points jonction.
— C_2 indique une continuité de la courbure : les polynômes successifs ont en plus des dérivées secondes égales aux points jonction. Le cas le plus courant des splines est la spline cubique. Elle est uniforme et définie par des polynômes de degré 3. Un polynôme de degré 3 s'écrivant ,

$$P(t) = a + bt + ct^2 + dt^3$$

il nécessite 4 contraintes (a,b,c,d) pour être défini.
Ces 4 contraintes par intervalle vont nous permettre d'interpoler des courbes splines passant par un ensemble de points donnés.