1 Introduction à R

R est un logiciel pour l'analyse statistique des données. Il fournit les procédures usuelles (t-tests, anova, tests non paramétriques...) et possède des possibilités graphiques performantes pour explorer les données. Pouvant être utilisé aussi bien en mode interactif qu'en mode batch, R est un logiciel libre, dont le code source est disponible et qui peut être recopié et diffusé gratuitement. Des versions compilées de R sont disponibles pour Linux, Windows et Mac OS X.

Au moment de la rédaction de ce document (Octobre 2004), la version courante de R est la 2.0.

1.1 Installation du système de base

Le site principal du logiciel R est www.r-project.org.

Le téléchargement de R se fait à partir d'un des sites du "*Comprehensive R archive Network*" (CRAN), par exemple cran.cict.fr (cf. Fig. 1.1).

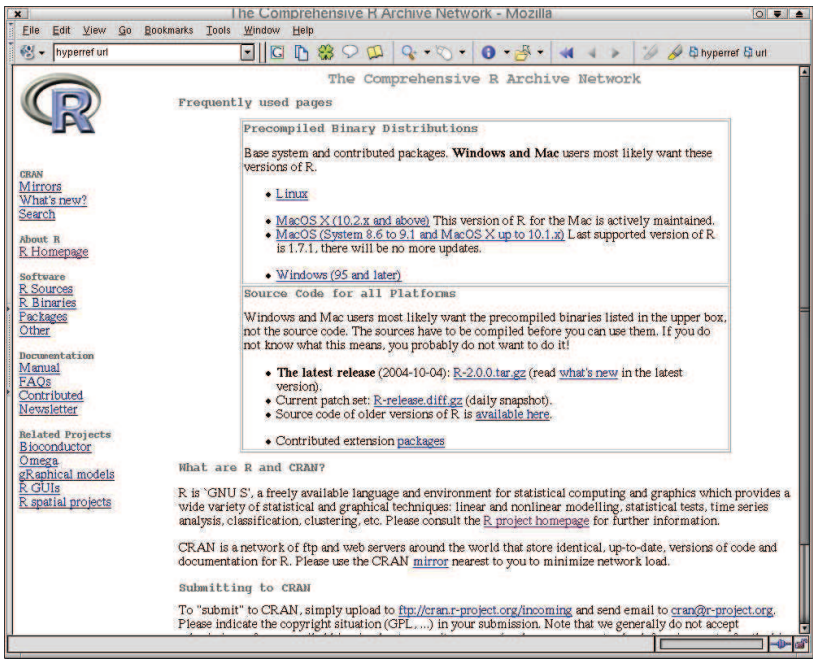

Fig. 1.1 – Site de téléchargement de R

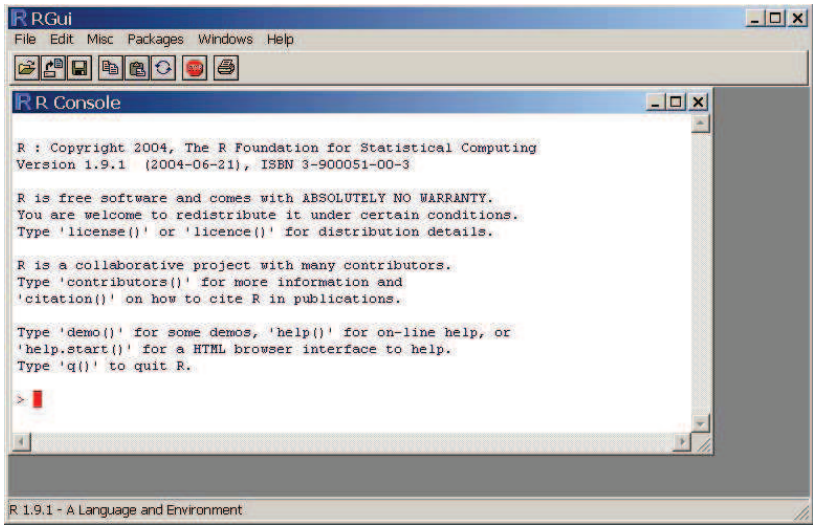

FIG. 1.2 – RGui : l'interface graphique de R sous Windows

Installation sous Windows : Le programme d'installation pour Windows est accessible en suivant les liens "Windows", puis "Base". Le nom de ce programme dépend de la version, il s'agit, par exemple, de "`Rw2000.exe`" pour la version 2.0. Téléchargez ce fichier sur votre disque, puis cliquez-le pour installer le logiciel. Si vous acceptez l'option par défaut "Create a desktop icon", une icône représentant une lettre "R" en bleu est ajoutée sur le bureau. Cliquez dessus, pour voir apparaître la fenêtre 'RGui' ("R Graphical User Interface", voir figure 1.2)

Installation sous Mandrake Linux : Pour une installation sous Linux, vérifiez s'il existe un paquetage "rpm" adapté à votre distribution et prêt à être installé. Si tel est le cas, télécharger-le et installez le, en tant qu'administrateur, avec la commande "`rpm -i R*.rpm`".

Si vous utilisez Mandrake Linux 10.x, R fait partie de la distribution de base (il est sur les CD), et il suffit de taper "`urpmi R-base`" pour l'installer.

En l'absence de binaire précompilé, il vous faudra récupérer le code source (`R-2.0.0.tar.gz`) et le compiler avec une commande '`configure && make && make install`' (en tant qu'utilisateur 'root'). Cela ne doit pas poser de problème mais nécessite que les outils de compilation soient bien installés sur votre système (notamment le compilateur fortran g77).

Pour lancer R sous Linux, il suffit de taper "`R`" dans un terminal (cf. Fig. 1.3).

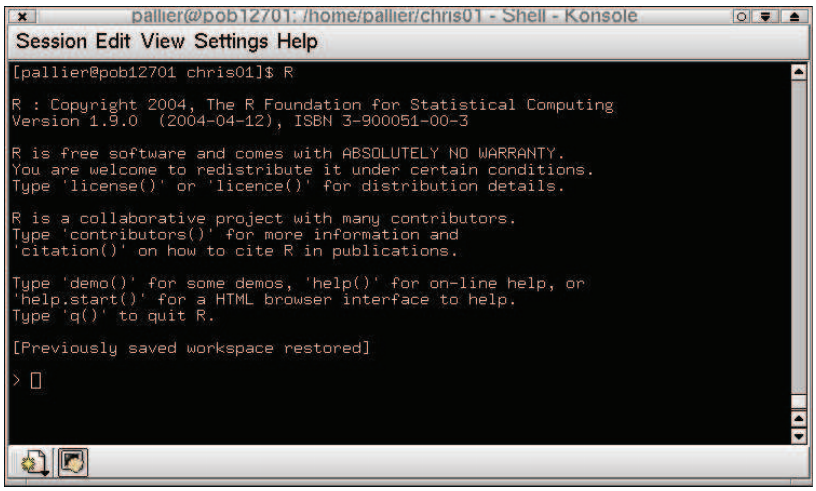

FIG. 1.3 – R dans un terminal sous Linux

1.2 Modules additionnels

Après avoir installé le système de base, vous pouvez installer des modules supplémentaires, parfois appelés "paquetages" (packages), qui ajoutent des fonctions à R.

Pour l'analyse des données d'expériences, les paquetages `car`, `gregmisc`, `vcd`, `psy`, `multcomp` fournissent des fonctions supplémentaires intéressantes. Par exemple, "`multcomp`" fournit diverses procédures pour effectuer des comparaisons multiples (Dunnett, Tukey, Sequen, AVE, Changepoint, Williams, Marcus, McDermott, Tetrade).

Ces modules sont disponibles sur les sites CRAN dans la section "*Contributed extension packages*".

Pour installer un module sous Windows, dans RGui, utiliser le menu 'Package/Install package from CRAN' (il faut être connecté a Internet).

Pour installer un module sous Linux, il faut d'abord télécharger le fichier `package.tar.gz` du CRAN, puis, en tant que root, exécuter :

```
R CMD INSTALL package.tar.gz
```

1.3 Interfaces graphiques

Rest un programme avec lequel on communique en tapant des commandes plutôt qu'en cliquant dans des menus ou sur des icônes.

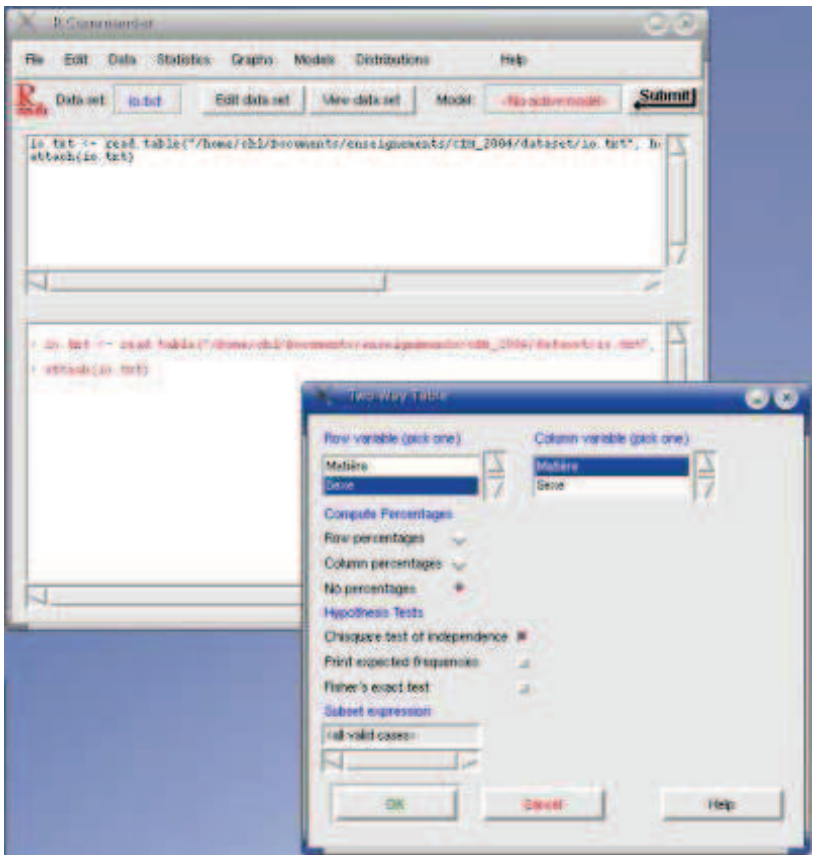

FIG. 1.4 – L'interface Rcommander sous Linux

Il existe cependant des systèmes à bases de menu et d'icônes (des "cliquodrômes") qui gèrent l'interaction avec R, et permettent, plus ou moins, d'éviter de taper des commandes. Citons, entre autres, les interfaces graphiques "Rcommander" (Linux + Windows, cf. figure 1.4), et "SciViews" (Windows).

Néanmoins, il nous paraît qu'apprendre les commandes de R permet de mieux comprendre ce qu'on fait et autorise finalement plus de flexibilité. Pour ces TPs, nous avons fait le choix de vous enseigner les rudiments du langage R.

1.4 Documentation

De nos jours, beaucoup de gens trouvent naturel de pouvoir utiliser les logiciels sans lire de documentation. Si cela est raisonnable pour les logiciels qui réalisent des opérations assez simples, c'est dangereux avec les logiciels qui effectuent des opérations conceptuellement compliquées.

Dans le cas de R, qui comprend de nombreuses commandes, il est illusoire d"envisager utiliser ce logiciel sans lire un minimum de documentation. Notre expérience est que les premières heures d'analyse de données avec R nécessitent de fréquents recours aux documentations, mais lorsqu'on est devenu à l'aise, alors il n'y a pratiquement plus besoin de s'y référer.

Il est donc utile de savoir où chercher l'information a propos de R.

Pour les débutants, on trouve sur Internet un bon nombre de documents sur R, notamment dans la section "Documentation/Contributed" du site www.r-project.org. Mentionnons en particulier :

- R pour les débutants par Emmanuel Paradis.
- Introduction au système R par Yves Brostaux.
- le site *Statistiques avec R*, réalisé par Vincent Zoonekynd, à l'adresse suivante : http ://zoonek2.free.fr/UNIX/48_R/all.html.
- Introduction to analysis of variance with "R", (qui bien qu'inachevé, vous sera sans doute utile).
- Notes on the use of R for psychology experiments and questionnaires par Jonathan Baron and Yuelin Li.

R possède aussi une documentation officielle, sous forme de fichiers pdf et html, qui est copiée sur votre disque dur lors de l'installation du logiciel. Dans l'interface graphique sous Windows, les manuels au format pdf sont accessibles dans les menus `Help/Manuals`. Il est fortement conseillé de parcourir, au minimum, les deux documents "*An Introduction to R*" et "*R Data Import et Export*".

Les manuels sont également accessibles sous forme html, dans le menu `Help/Html help` sous Windows, et en tapant `help.start()` sous Linux. Cela ouvre votre navigateur Internet sur une page web locale qui contient divers liens, entre autres vers ces manuels. Par exemple, le lien `Packages/base` liste les commandes de bases de R.

Il existe plusieurs livres publiés qui traitent de R. Pour les débutants, les deux livres suivants peuvent offrir une aide utile :

- *Introductory statistics with R* par Peter Dalgaard édité par Springer-Verlag.
- *An R and S-plus companion to applied regression* par John Fox, édité par Sage publications.

Pour un niveau plus avancé :

- *Modern Applied Statistics with S-PLUS* par Venables et Ripley.
- *Mixed-Effects Models in S and S-PLUS* par Pinheiro et Bates

2 Premiers pas

L'interaction avec R se fait en tapant des commandes dans la fenêtre `R Console`.

2.1 Entrer des commandes dans la console R

Pour commencer, vous pouvez utiliser R comme une calculatrice. Cliquez dans la fenêtre 'R Console', puis tapez :

```
2+3
```

Le résultat, '5', doit s'afficher.

Poursuivez avec :

```
a=5
a+8
```

RGui doit se présenter comme sur la Figure 2.1 page suivante.

Le principe de R est le suivant : vous entrez une ligne de commande, et quand vous tapez sur 'Entrée', R lit cette ligne et effectue l'opération demandée.

Essayez maintenant les commandes suivantes :

```
a=1:10
a
b=rnorm(10)
plot(a,b)
plot(a,b,pch=16,col=2)
```

La commande `plot` provoque l'affichage d'une fenêtre graphique (Fig. 2.2).

Cliquez à nouveau dans la fenêtre `R Console`, puis tapez :

```
a=c(3,4,6,7,8,9)
a
length(a)
b=c('alpha','beta')
b
length(b)
```

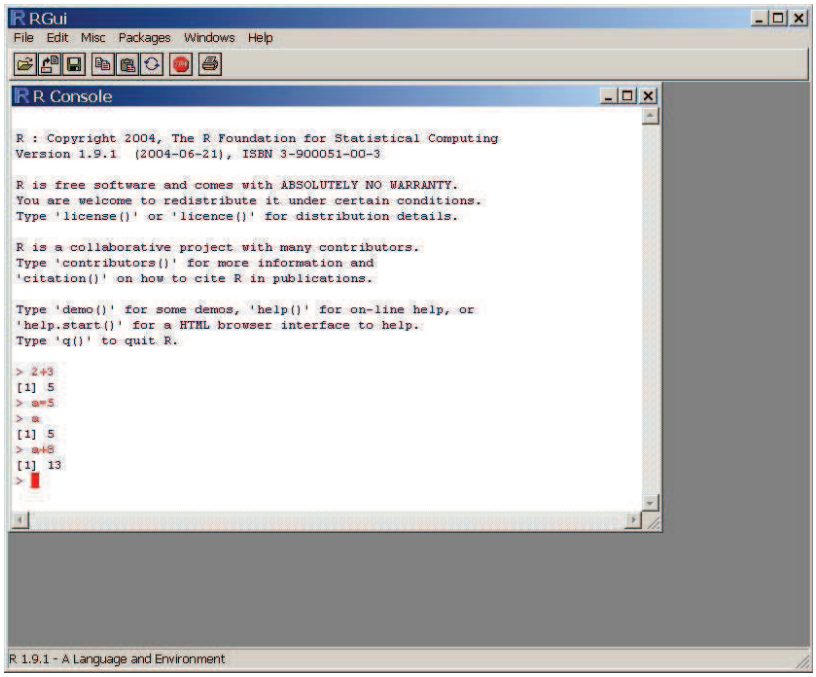

FIG. 2.1 – De simples additions

La variable 'a' contient un vecteur numérique à six éléments.

La variable 'b' contient un vecteur contenant deux chaînes de caractères.

Les concepts de *vecteur* et de *variable* sont essentiels dans R. On y reviendra plus tard ; pour le moment, retenez que :

- un vecteur n'est rien d'autre qu'un suite d'items qui ont tous le même type (numérique, chaîne de caractères, ...). C'est l'objet de base dans R.
- une variable contient un objet, et permet de le retrouver sans le ré-écrire en entier.

Comme on l'a déjà vu, la liste des variables peut être affichée par "`ls()`.", et un variable peut être détruite par la commande "`rm(nom)`".

Entrez les commandes suivantes, pas à pas, et observez le résultats :

```
a=rnorm(20,mean=55,sd=10)
mean(a)
sd(a)
max(a)
summary(a)
hist(a)
boxplot(a)
```

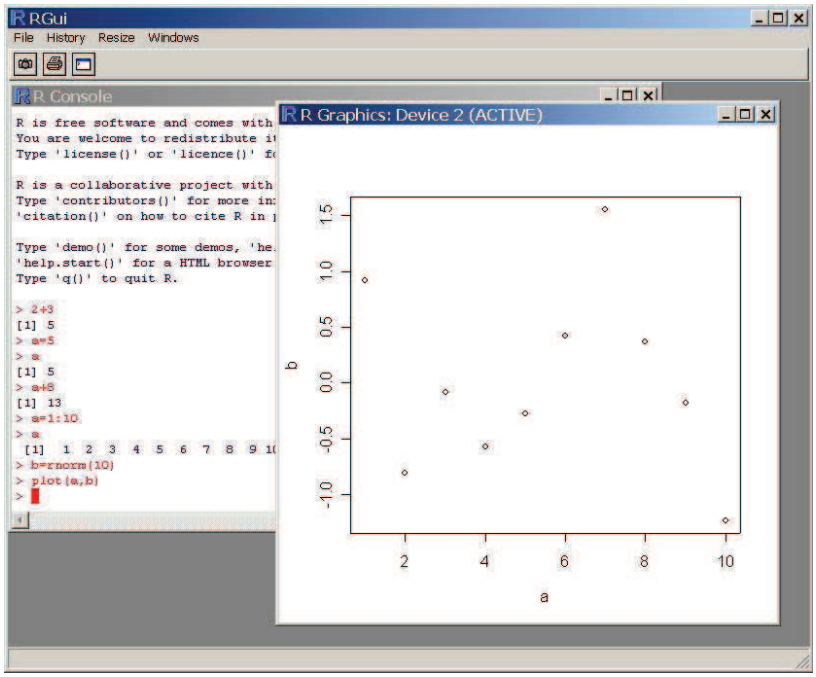

FIG. 2.2 – Fenêtre graphique

```
stripchart(a)
stripchart(a,pch=16,cex=2,col=2,method='jitter',vertical=T)

x1=rnorm(10,mean=100,sd=10)
x2=rnorm(10,mean=110,sd=10)
boxplot(x1,x2)
t.test(x1,x2)
plot(x1,x2)
summary(lm(x2~x1)
```

2.2 Aide en ligne

A tout moment, une aide en ligne est disponible à l'aide de la commande *help.search('mot clé')*. La description détaillée d'une commande s'obtient en tapant ' *?nom_de_la_commande*'.

Essayez :

```
?t.test
help.search("test")
help.start()
```

2.3 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 laissée 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.

2.4 Sauvegarder les commandes dans un script

Tapez la commande `history()`. Une fenêtre s'affiche listant les dernières commandes que vous avez tapées (voir figure 2.3 page suivante).

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".

[1] Pour 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.

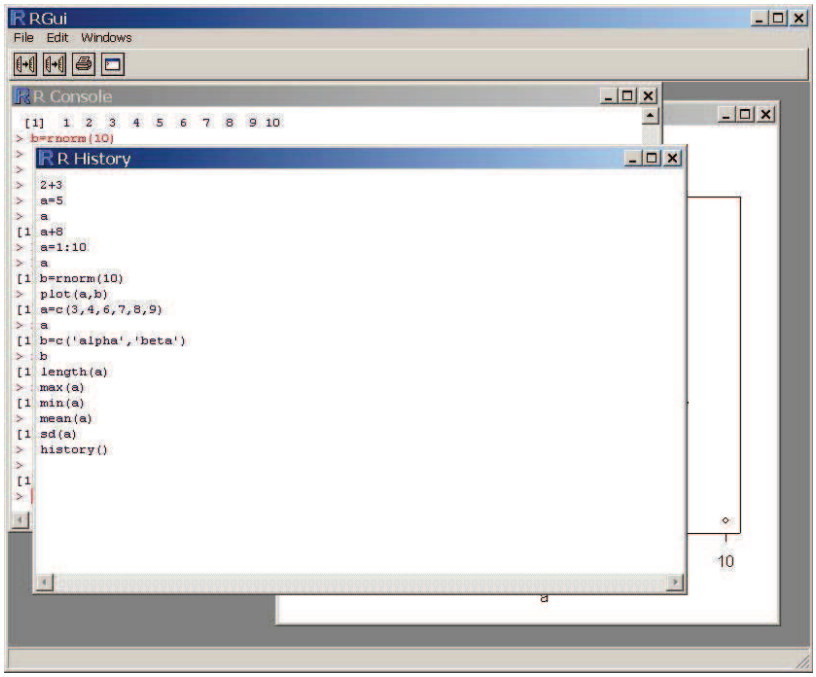

FIG. 2.3 – Historique des commandes affiché par `history()`

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-executer 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)
```

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'.

2.5 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-automotique.

2.6 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).

3 Manipulations de base

3.1 Objets

L'objet de base en R est le vecteur. Un vecteur peut contenir des valeurs numériques, des valeurs de vérité (True or False), des chaînes de caractères... Les fonctions les plus utilisées pour créer des vecteurs sont `c`, `rep` et `seq` :

```
c(1,2,3,4,5,6)
c(T,T,F,F)
c('a','b')
rep(55,10)
rep(c(1,2),10)
rep(c('a','b'),c(2,7))
seq(1,10,by=.1)
```

Un type de vecteur particulièrement utile est le type *factor*. Les facteurs sont des vecteurs utilisés pour *classifier* les valeurs d'autres vecteurs (les facteurs sont des "variables indicatrices"). Par exemple, étant donné 100 scores provenant de plusieurs groupes de sujets, une variable facteur peut désigner ces sous-groupes.

```
(a=factor(c(rep('alpha',10),rep('beta',10))))

(b=gl(3,4,48,labels=c('a','b','c')))

(x=rnorm(48))
tapply(x,b,mean)
boxplot(x~b)
stripchart(x~b,method='jitter')
stripchart(x~b,method='jitter',vertical=T)
```

On peut créer un facteur à partir d'un vecteur grâce à la fonction `factor`, ou directement avec la fonction "`gl`".

3.2 Accéder aux éléments d'un vecteur

```
(a=rnorm(50))
a[1]
a[2]
a[c(1,3,5)]
```

```
a>0
a[a>0]

(b=gl(2,25,labels=c('g1','g2')))
a[b=='g1']
```

Une particularité de R est que les éléments d'un vecteur peuvent avoir des noms :

```
v=c(1,2,3,4)
names(v)=c('alpha','beta','gamma','delta')
v['beta']
```

Cela s'avère très utile pour créer des dictionnaires. Par exemple, un vecteur '`freq`' donnant la fréquence d'usage des mots peut avoir les mots comme 'names' ; il suffit alors de taper "`freq['aller']`" pour obtenir la fréquence du mot 'aller'.

```
mots=c('aller','vaquer')
freq=c(45,3)
freq
freq[mots=='aller']
names(freq)=mots
freq
freq['aller']
```

3.3 Arrays, listes et data.frames

D'autres objets de R sont les les listes, les arrays (vecteurs multidimensionnels) et les data.frames.

Les data.frames sont des listes de vecteurs qui ont tous la même longueur. Les data.frames sont très bien adaptés pour stocker des données présentées sous forme de tableau bi-dimensionnel.

```
(a=array(1:20,dim=c(4,5)))
a[2,4]

(b=list(alpha=1:3,
        beta=c('a','b','c','d')))
names(b)
b$alpha
b$beta

(c=data.frame(a=gl(2,5,10),b=1:10,x=rnorm(10)))
c$a
c$b
c$x
c[1:2,]
```

3.4 Variables

Les objets peuvent être enregistrés dans des variables avec l'opérateur = (ou <-). Pour voir le contenu de l'objet représenté par une variable, il suffit de taper le nom de celle-ci.

```
a<-c(1,2,3)
a
ls()
rm(a)
ls()
```

Les vecteurs contenus dans une liste ou dans un data.frame sont accessibles avec le symbole $. Un data.frame peut être "attaché" pour que ses vecteurs soient directement accessibles.

```
mydata<-data.frame(a=gl(2,5,10),b=1:10,x=rnorm(10))
names(mydata)
mydata$a
mydata$b
mydata$x
attach(mydata)
a
b
x
detach(mydata)
```

3.5 Lire des données

Quand les données sont très peu nombreuses, on peut les entrer directement dans un vecteur (comme on l'a fait jusqu'ici) avec la fonction 'c'.

Les fonctions `scan` et `read.table` permettent de lire des données enregistrées dans des fichiers textes.

`scan` lit une suite de données dans un vecteur.

Avec un éditeur de texte, créez un fichier `datafile1.txt` contenant :

```
3.4 5.6 2.1 6.7 8.9
```

Puis, dans R, entrez :

```
scores<-scan('datafile1.txt')
```

On peut également entrer des données directement en ligne de commande :

```
scores<-scan('')
```

La fonction *read.table* lit des données présentées sous forme tabulaire (par ex. les fichiers .csv enregistrés par Excel) et renvoie un data.frame.

Créez un fichier `datafile2.txt` contenant :

```
sujet groupe score
s1    exp    3
s2    exp    4
s3    exp    6
s4    cont   7
s5    cont   8
```

Puis importez le dans R :

```
a<-read.table('datafile2.txt',header=T)
a
```

R dispose d'un éditeur de data.frame très limité :

```
scores<-edit(data.frame(a))
```

scan et read.table ne lisent que des fichiers textes, Le package 'foreign' permet de lire directement certains fichiers de données binaires provenant de SPSS, SAS, ...

```
library(help='foreign')
```

Mentionnons également l'existence de packages permettant d'accéder à des informations stockées dans des bases de données (MySQL, Oracle...).

4 Statistiques élémentaires

Cette section a pour but d'illustrer quelques concepts fondamentaux de la statistique inférentielle, et de présenter les principales fonctions de R pour le traitement statistique des données recueillies lors d'un protocole expérimental.

4.1 Manipulation des distributions de probabilités

4.1.1 Distributions univariées

Différentes fonctions permettent de générer des nombres aléatoires suivant une certaine distribution de probabilité :

```
runif(10) # distribution uniforme
rnorm(10) # distribution normale
rnorm(10,mean=100)
rbinom(10,size=1,prob=.5) # distribution binomiale
```

La fonction `rnorm` génère des nombres aléatoires distribués selon une loi normale. En augmentant le nombre d'échantillons générés (de 10 à 10000), on constate que la distribution des valeurs obtenues se rapproche de plus en plus d'une distribution normale continue :

```
s1=rnorm(10,mean=2)
summary(s1)
s2=rnorm(100,mean=2)
summary(s2)
s3=rnorm(10000,mean=2)
summary(s3)

par(mfrow=c(3,3))       # organisation des graphiques selon une matrice 3 x 3

hist(s1)                # histogrammes
hist(s2)
hist(s3)

# graphes en évitant le chevauchement des points de même coordonnées
stripchart(s1,method='jitter',vert=T,pch=16)
stripchart(s2,method='jitter',vert=T,pch=16)
stripchart(s3,method='jitter',vert=T,pch='.')


plot(density(s1))       # fonction de densité
```

```
x=seq(-5,5,by=.01)      # vecteur de coordonnées normées pour les abscisses
lines(x,dnorm(x,mean=2),col=2)
plot(density(s2))
lines(x,dnorm(x,mean=2),col=2)
plot(density(s3))
lines(x,dnorm(x,mean=2),col=2)
```

En première approximation, la distribution théorique de la taille des individus de sexe masculin, français, et dans la tranche d'âge 20-35 ans,suit une loi normale de moyenne 170 et d'écart-type 10.

On peut donc non seulement situer un individu, ou un groupe d'individus, dans cette distribution, mais également évaluer la probabilité qu'un individu choisi au hasard parmi la population entière mesure moins de 185 cm, ou plus de 198 cm, ou ait une taille comprise entre 174 et 186 cm.

Lorsque l'on ne dispose pas des tables de lois normales $N(\mu;\sigma^2)$ (il y en a une infinité puisqu'il y a 2 paramètres libres), on utilise la loi normale centrée-réduite $N(0;1^2)$ (encore appelée loi Z), dont la table est disponible la plupart des manuels ou bien sur le web. Cependant R fournit directement les tables des lois normales, par l'intermédiaire de la commande `pnorm`, qui prend en arguments la valeur repère, la moyenne et l'écart-type théoriques.

```
taille=seq(130,210,by=1)
plot(taille,dnorm(taille,mean=170,sd=10),type='b',col="red")
pnorm(185,mean=170,sd=10)
abline(v=185,col=4)
text(185,.012,paste("P(X<185)=",signif(p,3)),col=4,pos=2,cex=.6)
p=pnorm(198,mean=170,sd=10)
abline(v=198,col=4)
text(198,.002,paste("P(X>198)=",round(1-p,3)),col=4,pos=4,cex=.6)
```

La probabilité qu'un individu choisi au hasard parmi la population entière mesure moins de 185 cm ($P(X < 185)$) est de 0.933 (obtenu par `pnorm(185,mean=170,sd=10)`). La probabilité qu'un individu mesure plus de 198 cm est de 0.003 ($1 - P(X < 198)$, et la probabilité que sa taille soit comprise entre 174 et 186 est 0.290 ($P(X < 186) - P(X < 174)$).

On constate que la probabilité qu'un individu choisi aléatoirement dans une population de moyenne 170 ± 10 mesure plus de 198 cm est très faible. C'est sur la base de ce calcul de probabilités que repose le test de typicalité, ou "test Z" : un groupe d'individus (i.e. un échantillon) sera déclaré atypique ou non représentatif de la population parente dont il est issu, lorsqu'il a une position au moins aussi extrême qu'une certaine position de référence, correspondant en général à la probabilité 0.05.

R permet également de générer d'autres distributions de probabilités, notamment la loi binomiale, les lois statistiques telles que le t de Student, le F de Fisher-Snedecor, le chi-deux (χ^2), etc. On peut ainsi voir dans l'exemple qui suit que la distribution du t de Student tend vers la loi normale lorsque la taille de l'échantillon est suffisamment grande (dans cet exemple, on a manipulé le degré de liberté `df`, donné en argument de la fonction `dt`).

```
?pnorm
?pt
?pbinom
help.search('distribution')
pnorm(2)
pt(3,df=10)             # fonction de répartition de la loi du t de Student
qnorm(.99)              # donne la valeur associée au 99ème centile d'une distribution normale
t<--50:50/10
plot(dnorm(t),type='l',col='red')
par(new=T)              # le prochain graphe sera superposé au précédent
plot(dt(t,df=5),type='l')
```

Imaginons que vous disposiez d'une pièce dont vous vous demandez si elle est baisée. Vous prévoyez de la lancer 10 fois à pile ou face. A partir de quelle proportion relative d'essais face/pile (ou l'inverse) considérerez- vous que la pièces est truquée ?

Si la pièce n'est pas truquée, le nombre de "pile" suit une loi binomiale.

```
plot(dbinom(0:10,rep(10,11),prob=1/2),type='h')
hist(rbinom(100,10,.5))
hist(rbinom(1000,10,.5))
hist(rbinom(10000,10,.5))
```

Supposez que vous tiriez à pile ou face 10 fois de suite, et que la pièce retombe 8 fois sur 'pile'. Quelle la probabilité d'observer cela si la pièce n'est pas biaisée ?

```
binom.test(8,10)
prop.test(8,10,1/2) # test approché
```

4.1.2 Distributions conjointes

Si l'on reprend l'exemple précédent des tailles de la population française masculine (20-25 ans), on a une distribution similaire (i.e. suivant une loi normale de moyenne 70 et d'écart-type 7) pour les poids. On peut bien évidemment se poser les mêmes questions que précédemment, mais on peut également s'intéresser à la relation entre ces deux variables quantitatives. En représentant le poids en fonction de la taille, on peut évaluer la liaison linéaire entre ces deux variables à l'aide du coefficient de corrélation de Bravais-Pearson.

Pour illustrer cela, nous allons utiliser les données issues d'une population d'enfants de sexe masculin âgés de 11 à 16 ans.

```
taille<-scan('')        # saisie manuelle des données
1: 172 155 160 142 157 142 148 180 167 165
11:
```

```
Read 10 items           # indicateur de fin d'entrée-sortie généré par R
poids<-scan('')
1: 50.5 38.1 57.3 39.3 46.1 37.1 45.9 66.3 60 50.5
11:
Read 10 items
plot(poids~taille)
r<-lm(poids~taille)     # modèle linéaire (x,y)
summary(r)                      # diagnostic de la régression
abline(r)                       # tracé de la droite de régression
-55.1963626 + 175 * 0.6568411   # "prédiction" pour taille=175 cm
predict(r,list(taille=c(175))
```

Ensuite, à partir de la connaissance de cette liaison linéaire, on peut se demander quelle serait le poids théorique (non observé) d'un individu dont on ne connaît que la taille : c'est le domaine de la régression linéaire. L'affichage des paramètres de la droite de régression donne la relation poids = 0.657 x taille - 55.196. Ainsi, on peut *prédire* que le poids d'un enfant mesurant 175 cm sera de 59.8 kg.

4.2 Résumés numériques et représentations graphiques

4.2.1 Résumés numériques

Le résumé statistique des principaux indicateurs descriptifs de position et de dispersion peut être obtenu à l'aide des fonctions `mean`, `sd`, `median` ; la fonction `summary` donne un résumé plus complet – par exemple, lorsqu'il s'agit d'un vecteur, elle indique la moyenne et la médiane, ainsi que l'étendue et les valeurs des premier et troisième quartiles.

```
a<-rnorm(100)
mean(a)
sd(a) # écart-type corrigé
summary(a)
boxplot(a)
mean(a,trim=.1) # moyenne sans les 10 % d'observations en fin de vecteur
```

4.2.2 Représentations graphiques

Les fonctions graphiques standard en 2D – `boxplot`, `plot`, `hist` – ont été vues dans les sections précédentes. La création de graphiques personnalisés sous R est facilitée par son extrême souplesse quant au paramétrage des graphiques (positionnement, symboles et type de tracés, etc.). L'utilisation de l'aide en ligne est vivement recommandée.

Pour les graphiques en trois dimensions (z étant une matrice de dim 3), on pourra utiliser les fonctions `image` et `contour` :

```
x=1:10
y=1:10
z=outer(x,y,"*")
persp(x,y,z)
image(z)
contour(z)
```

4.3 Définition de fonctions

Il est possible de définir ses propres fonctions sous Ret d'enrichir ainsi le langage.

Par exemple, Rne possède pas de fonction pour calculer l'erreur-type ($\sigma/\sqrt{(N)}$). On peut en définir une de la manière suivante :

```
se <- function (x) { sd(x)/sqrt(length(x)) }
```

L'exemple suivant permet de calculer la moyenne arithmétique après suppression des valeurs atypiques, i.e. supérieures à 2 écart-types de la moyenne :

```
clmean <- function (x) {
  m<-mean(x)
  d<-sqrt(var(x))
  threshold<-2
  mean(x[(x-m)/d<threshold])
}

a<-c(rnorm(100),5)
mean(a)
clmean(a)
```

On peut lire le code des fonctions existantes :

```
clmean
ls
t.test
methods(t.test)
getAnywhere(t.test.default)
```

5 Tests statistiques

Ce chapitre a pour but de présenter de manière non exhaustive certains tests statistiques employés fréquemment en statistique inférentielle.

Comme on l'a vu précédemment (voir section 4.1), la détermination des seuils de significativité (p) se fait grâce aux fonctions associées à chaque distribution (voir section 4.1).

```
1-pnorm(167,mean=150,sd=10)
1-pbinom(8,10,0.5)
```

5.1 Test du khi-deux

Soit le tableau de contingence A x B suivant à analyser :

	A1	A2	A3
B1	13	24	20
B2	10	7	18

Le calcul du test du χ^2 associé à ce tableau s'effectue de la manière suivante :

```
a<-scan('')
1: 13 24 20
4: 10 7 18
7:
Read 6 items
chisq.test(matrix(a,2,3,byrow=T))
```

5.2 Estimation de la moyenne d'un groupe

L'intervalle de confiance de la moyenne peut être obtenu à l'aide de la fonction `t.test` :

```
a<-10+rnorm(10,sd=10)
t.test(a,conf.level=.01)
```

Si l'hypothèse de normalité n'est pas soutenable, le test de Wilcoxon (non-paramétrique) peut être utilisé à l'aide de la fonction `wilcox.test` : ce test des signes permet de déterminer si la médiane du groupe peut être considérée comme significativement différente de 0.

5.3 Comparaison de deux groupes

Ce sont les mêmes fonctions – `t.test` (test paramétrique) et `wilcox.test` (test non paramétrique) – qui permettent la comparaison entre deux groupes ; dans ce cas, on passe en arguments les deux groupes :

```
a<-rnorm(10)
b<-rnorm(10,mean=1)
t.test(a,b)
wilcox.test(a,b)

c<-c(a,b)
x<-gl(2,10,20)
t.test(c~x)
wilcox.test(c~x)
```

5.4 Analyse de variance sur un facteur

Lorsque l'on est en présence d'un ensemble de k observations indépendantes (un seul facteur inter-sujets), on peut comparer leurs moyennes respectives à l'aide de la fonction `aov` (ou selon un modèle linéaire général, avec la fonction `lm`).

```
x<-rnorm(100)
a<-gl(4,25,100)
plot(x~a)
r<-aov(x~a)
anova(r)
pairwise.t.test(x,a)
t.test(x[a==1],x[a==2])
```

5.5 Anova sur deux facteurs

Avec deux facteurs inter-sujets, le principe d'analyse est le même, mais on étudie également l'interaction entre les deux facteurs.

```
x<-rnorm(100)
a<-gl(2,50,100)
b<-gl(2,25,100)
plot(x~factor(a:b))
interaction.plot(a,b,x)
l<-aov(x~a*b)
anova(l)
```

5.6 Anova sur des protocoles de mesures répétées

Avec un seul facteur intra-sujet, on procèdera ainsi :

```
subject<-gl(10,3,30)
cond<-gl(3,1,30)
x<-rnorm(30)
interaction.plot(cond,subject,x)
summary(aov(x~cond+Error(subject/cond))
```

Avec deux facteurs intra, la démarche est à peu près identique :

```
subject<-gl(10,4,40)
cond1<-gl(2,1,40)
cond2<-gl(2,2,40)
table(cond1,cond2)
x<-rnorm(40)
plot(x~factor(cond1:cond2))
interaction.plot(cond1,cond2,x)
interaction.plot(cond1,subject,x)
interaction.plot(cond2,subject,x)
summary(aov(x~cond1*cond2+Error(subject/(cond1*cond2))))
```

5.7 Régression linéaire

Comme nous l'avons vu dans le cas des distributions conjointes (cf. section 4.1.2), la démarche pour effectuer de la régression linéaire est la suivante :

```
a<-rnorm(100)
b<-2*a+rnorm(100)
plot(b~a)
r<-lm(b~a)
anova(r)
abline(r)
```

```
a<-rnorm(100)
b<-2*a+rnorm(100)
c<-5*a+rnorm(100)
pairs(cbind(a,b,c))
summary(lm(c~a*b))
```