Une introduction au langage **R**

Faouzi LYAZRHI
UP Biostatistique
Ecole Nationale Vétérinaire, 23, chemin des Capelles, BP 87614, F-31076 Toulouse cédex
email : f.lyazrhi@envt.fr

2005

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. Installer R

En cliquant (par exemple) sur le lien du site miroir du cict, sélectionner dans la page d'accueil (affichée comme ci-dessous) votre système d'exploitation :

Precompiled Binary Distributions

Base system and contributed packages. **Windows and Mac** users most likely want these versions of R.

- Linux
- MacOS X (10.2.x and above) This version of R for the Mac is actively maintained.
- MacOS (System 8.6 to 9.1 and MacOS X up to 10.1.x) Last supported version of R is 1.7.1, there will be no more updates.
- Windows (95 and later)

Cliquer sur base :

Note: CRAN does not have Windows systems and cannot check these binaries for viruses. Use the normal precautions with downloaded executables.

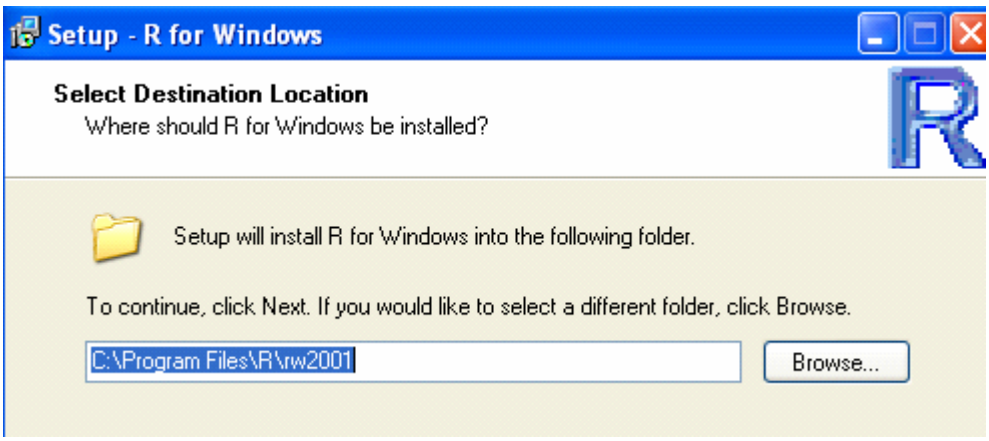
Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch)
contrib	Binaries of contributed packages (managed by Uwe Ligges)

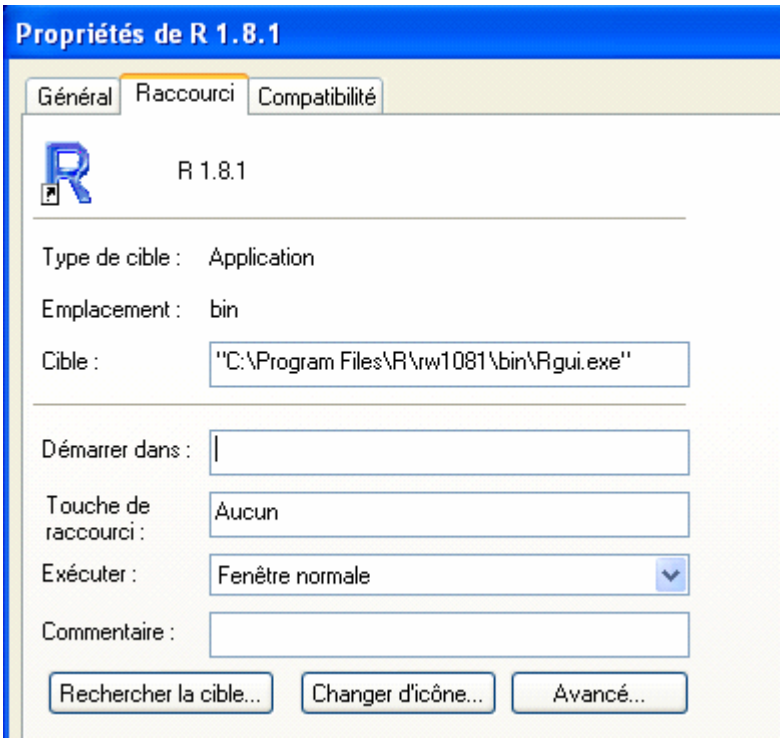
Puis télécharger le fichier rw2001.exe :

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.

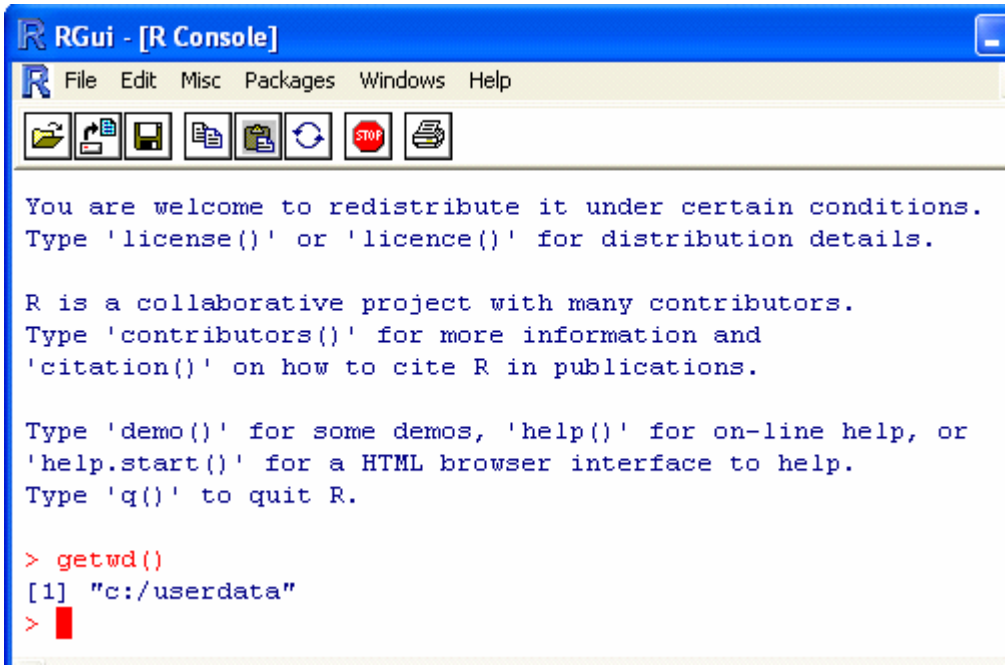
Le fichier rw2001.exe est un exécutable, une fois sauvegarder sur votre disque dur, vous double-cliquez dessus. Vous pouvez garder les options par défaut, nous verrons plus loin comment changer certaines options comme le répertoire de travail.



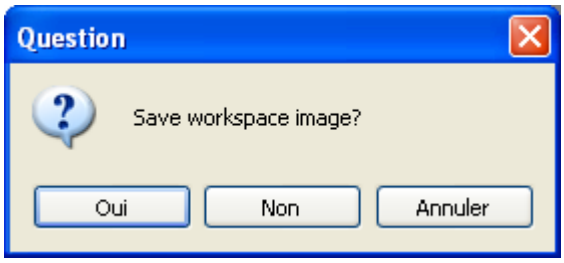
Un raccourci est créé sur le bureau Rgui.exe (R Graphique user interface).
Vous pouvez changer le répertoire de travail en modifiant démarrer dans : pour cela créer un répertoire sous la racine par exemple userdata, et taper le chemin complet dans la case démarrer dans : c:\userdata.



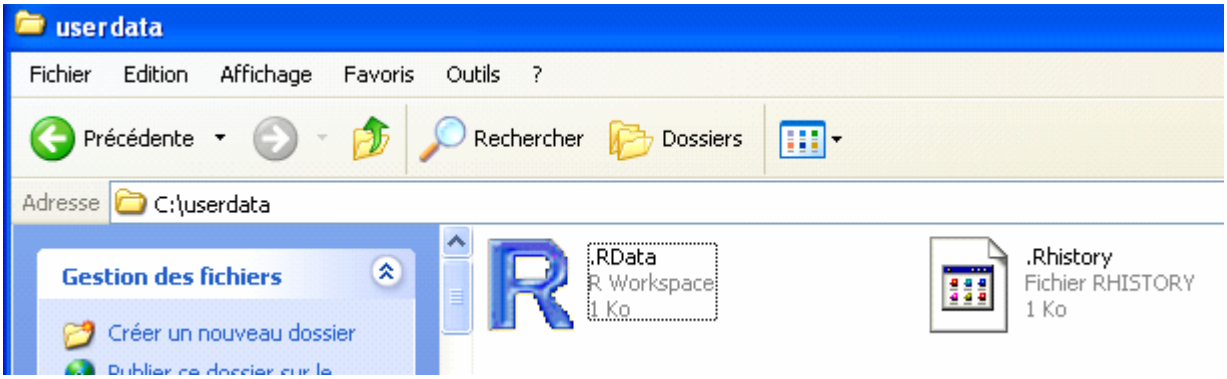
Pour vérifier votre dossier de travail, il suffit de double cliquer sur le raccourci rgui.exe, puis à l'invite de R signalant que la console est en attente d'une instruction (désigné par le symbol >) taper getwd() :



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.
Nous verrons plus loin comment utiliser ces deux fichiers.

2. Premières instructions avec R

Rappelons que R est un langage interactif, donc toujours en attente d'une instruction signalée par l'invite « < ». R dispose d'une aide en ligne très complète et qui vous sera 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 !!

2.1. 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 ente une majuscule une minuscule.

- **Calculer**

```
> 4+8
> 12
> (6+5*2)/2
> 8
> 2^2  (puissance)
> 4
> sqrt (9) (racine carrée)
>3
```

Remarque : A l'aide des flèches de déplacement du clavier vous pouvez rappeler les dernières instructions pour les modifier ou/et les réexécuter.

- **créer un objet**

Un objet peut-être créé à l'aide de l'opérateur « assigner » qui s'écrit avec le symbole < suivi du signe –
Cet objet est stocké dans la mémoire vive et peut être modifié en lui assignant une autre valeur.

```
> n <- 100
```

- **afficher les objets en mémoire**

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

```
> n
> 100
> n <- 100/2+5
> n
> 55
```

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
>nom <- "dupont"
ls()
[1] "n"  "m"  "s" "Dupont"
> ls.str()
m :  num 8   (num, variable numérique)
n :  num 12
s :  num 20
nom :  chr "dupont"  (chr,variable caractère)
```

- **Help (l'aide en ligne)**

R dispose d'une aide en ligne très exhaustive et qui peut vous être très utile. 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.
> help.start()
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

2.2 Importer, exporter des données

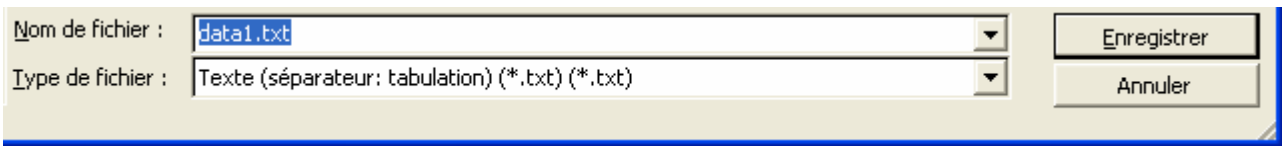
Le langage R peut lire des données provenant de sources externes sous forme de fichiers, comme il peut en créer et sauvegarder dans des formats transportables.
La fonction qui permet la lecture de ces fichiers est read.table, R dispose d'autres variantes de cette fonction (scan(), read.fwf,…).
Ces fonctions 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'.

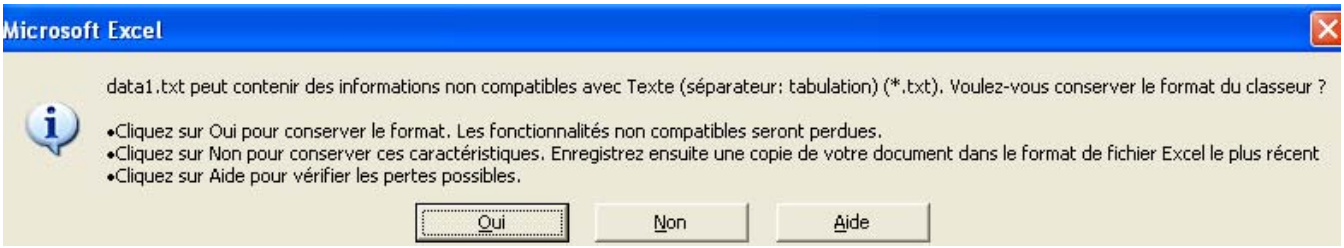
- **Importer des données**

Dans la plupart des cas un fichier de données est créé à partir d'un tableur ou un traitement de texte.
Le fichier data1.xls a été créé avec Excel et contient les données ci-après :

	A	B	C	D	E
1	poids	taille	fumeur	sexe	sport
2	85	184	oui	homme	1
3	65	175	oui	homme	1
4	74	180	oui	homme	2
5	79	175	oui	homme	2
6	71	165	non	homme	1
7	80	185	non	homme	1
8	75	180	non	homme	2
9	69	155	non	homme	2
10	74	168	oui	femme	1
11	64	171	oui	femme	1
12	63	169	oui	femme	2
13	70	165	oui	femme	2
14	55	167	non	femme	1
15	65	164	non	femme	1
16	67	155	non	femme	2
17	70	175	non	femme	2

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





A l'aide de la commande read.table() vous pouvez lire le fichier data1.txt dans R :
> read.table("data1.txt")
Cette instruction va lire le contenu du fichier data.txt :

```
> read.table("data1.txt")
      V1     V2     V3    V4    V5
1  poids taille fumeur  sexe sport
2     85    184    oui homme     1
3     65    175    oui homme     1
4     74    180    oui homme     2
5     79    175    oui homme     2
6     71    165    non homme     1
7     80    185    non homme     1
8     75    180    non homme     2
9     69    155    non homme     2
10    74    168    oui femme     1
11    64    171    oui femme     1
12    63    169    oui femme     2
13    70    165    oui femme     2
14    55    167    non femme     1
15    65    164    non femme     1
16    67    155    non femme     2
17    70    175    non femme     2
>
```

R nomme par défaut les variables V1, V2, V3,..etc. Par conséquent la première ligne contient le nom des variables.

```
> read.table("data1.txt",h=T)
   poids taille fumeur  sexe sport
1     85    184    oui homme     1
2     65    175    oui homme     1
3     74    180    oui homme     2
4     79    175    oui homme     2
5     71    165    non homme     1
6     80    185    non homme     1
7     75    180    non homme     2
8     69    155    non homme     2
9     74    168    oui femme     1
10    64    171    oui femme     1
11    63    169    oui femme     2
12    70    165    oui femme     2
13    55    167    non femme     1
14    65    164    non femme     1
15    67    155    non femme     2
16    70    175    non femme     2
>
```

L'argument « h=T », header=TRUE indique que dans le fichier data1.txt la première ligne correspond au nom des variables.
Si le fichier contient des données manquantes représentées par un caractère (tout caractère sauf un blanc). Dans ce cas il suffit d'utiliser l'argument na.strings= " NA", ou NA est le caractère représentant la donnée manquante.

> read.table("nom du fichier", na.strings="NA")

- **Conserver le contenu du fichier**

On peut conserver le contenu du fichier et le stocker dans un objet qu'on peut manipuler dans R :

```
> data2 <-read.table("data1.txt",h=TRUE)
> attach(data2)
> data2
```

la commande attach() permet de manipuler les variables séparément en les appelant par leur nom :

```
> poids
 [1] 85 65 74 79 71 80 75 69 74 64 63 70 55 65 67 70
>
```

Remarque 1 : si la première ligne ne contient pas le nom de la variable, dans ce cas chaque variable peut être appelée séparément à l'aide de l'instruction data2$V1, data2$V2,…etc
Remarque 2 : pour afficher les nome des variables contenues dans un fichier il suffit de taper names(data2) :

```
> names(data2)
[1] "poids"  "taille" "fumeur" "sexe"   "sport"  "etat"
>
```

R dispose d'un tableur, l'appel du tableur se fait à l'aide de l'instruction edit()
> edit(data2)

Data Editor

	poids	taille	fumeur	sexe	sport	v
1	85	184	oui	homme	1	
2	65	175	oui	homme	1	
3	74	180	oui	homme	2	
4	79	175	oui	homme	2	
5	71	165	non	homme	1	
6	80	185	non	homme	1	
7	75	180	non	homme	2	
8	69	155	non	homme	2	
9	74	168	oui	femme	1	
10	64	171	oui	femme	1	
11	63	169	oui	femme	2	
12	70	165	oui	femme	2	
13	55	167	non	femme	1	
14	65	164	non	femme	1	
15	67	155	non	femme	2	
16	70	175	non	femme	2	
17						

Le contenu du fichier peut être modifié à l'aide de l'éditeur. Ne pas oublier de confirmer les modifications en sauvegardant le fichier dans un autre objet :
> data2modif <- edit(data2)
ou bien à l'aide de la commande fix() :
> fix(data2)

Remarque :
Il est possible d'importer un fichier dans R depuis un répertoire distant par exemple depuis un site. le fichier data1.txt peut-être importer depuis le site http://www.biostat.envt.fr/~lyazrhi/data1.txt
il suffit de taper :

```
> data2 <-read.table("http://www.biostat.envt.fr/~lyazrhi/data1.txt",h=TRUE)
> attach(data2)
```

- **Exporter des données**

On peut exporter un objet créé dans R dans un autre logiciel par exemple Excel ou Word. Pour cela on utilise la fonction write.table() qui sauvegarde le contenu de l'objet dans un fichier texte :

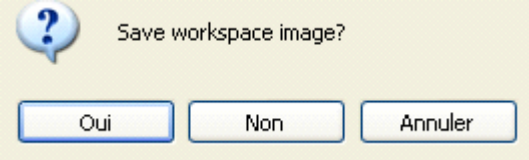
```
> write.table(poids,file="data2expor",quote=FALSE,row.names=FALSE,col.names=FALSE)
>
```

file=" " correspond au nom du fichier où sera sauvegardé le contenu de l'objet.
quote, si FALSE les variables et leur contenu ne seront pas considérés comme caractère et donc ne seront pas écrits entre " ".
row.names si FALSE indique qu'il ne faut pas créer dans le fichier une colonne contenant les noms des lignes.
col.names si FALSE même chose pour les noms des colonnes.

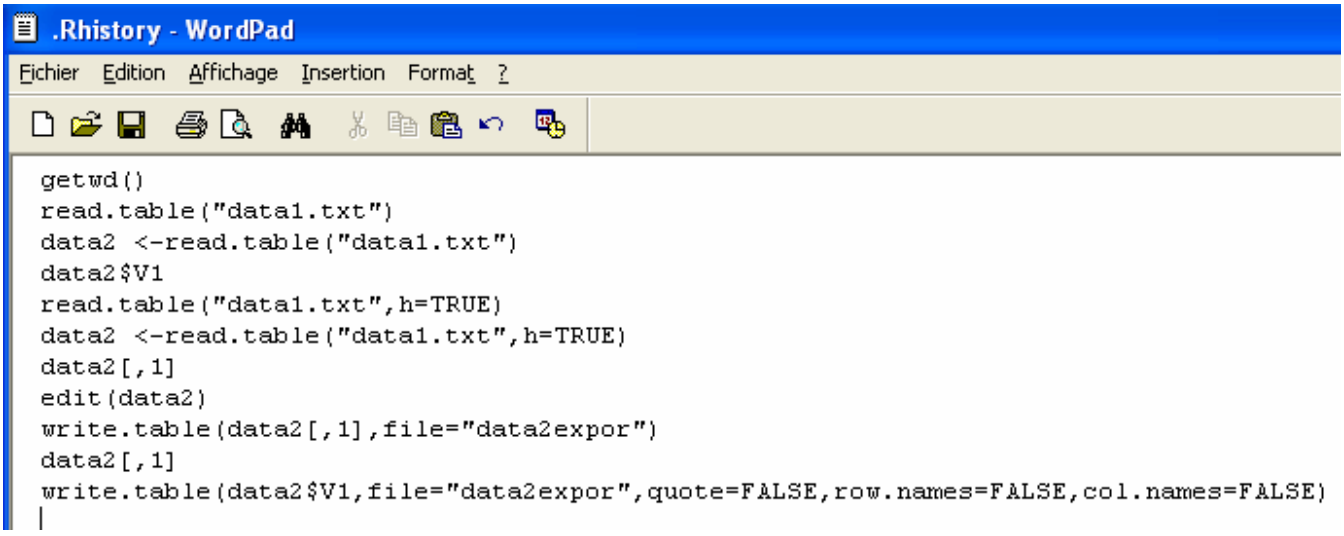
Pour consulter le fichier créé (data2expor dans l'exemple), vous ouvrez Excel puis fichier-ouvrir et éditer le fichier créé qui se trouve dans votre répertoire de travail.

- **Utiliser l'historique**

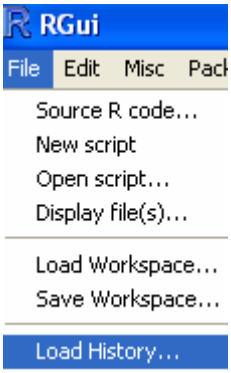
L'utilisation de l'historique n'est possible que si vous avez répondu oui en quittant R :



L'historique de toutes les commandes utilisées est sauvegardé dans le fichier .Rhistory. Vous pouvez l'éditer avec un éditeur de texte de votre choix :



Cet historique peut-être utilisé dans une autre session, pour cela vous pouvez l'ouvrir à l'aide de la commande *load history* du menu File :



Ensuite avec les flèches ``en haut'' et ``en bas'' du clavier vous rééxecuter toutes les commandes en les modifiant éventuellement.

- **Conserver les résultats d'une analyse**

Par ailleurs on peut conserver les commandes et/ou les résultats affichés dans la fenêtre principale de R ou des graphiques dans un document (Word par exemple). Pour cela il suffit de sélectionner avec la souris les résultats que vous souhaitez inclure dans un rapport ou conserver dans un document et de faire un copier/ coller. Sinon, Les résultats peuvent être renvoyés automatiquement dans un fichier texte à l'aide de la commande sink() :

> sink('sortie.txt')
> a=1:10 (génère une suite de nombres de 1 à 10)
> a
>b=a/4
>b
> sink() (ferme le fichier résultat et redirige les sorties à l'écran)
Vous pouvez ouvrir le fichier sortie.txt avec un Word par exemple

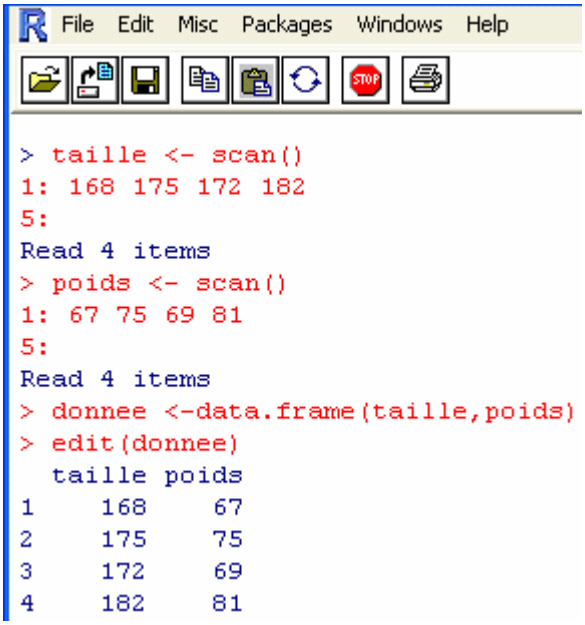
2.3 Créer, générer, saisir des données

R dispose d'une large bibliothèque de fonctions mathématiques et de densité de probabilités qui permettent de créer, transformer des données ou de les générer et de les stocker dans des fichiers.
R permet aussi la saisie directe d'un tableau de données.

- **Saisir des données**

On peut saisir les données et les stocker dans un fichier en les tapant au clavier. Pour cela on peut les saisir colonne (vecteur) par colonne en utilisant la commande scan() ou c(), ou bien les saisir sous forme de table (matrice) à l'aide de la commande data.frame() ou matrix().
Par exemple supposons que l'on dispose de deux variables taille (cm) et poids (kg) mesurés sur 4 individus :
taille : 168 ; 175 ; 172 ; 182
poids : 67 ; 75 ; 69 ; 81

1ère méthode :
Avec la commande scan(), on saisie au clavier les valeurs de chacune des deux variables, puis la commande data.frame() permet des les stocker dans une même tableau qui peut être transportable :



Remarque : Avec la commande write.table (voire plus haut) on peut exporter le fichier dans d'autres logiciels.

2ème méthode :

On peut utiliser la fonction data.frame() qui fait appel à la commande c() :

> data.frame(taille=c(168 ,175,172,182), poids=c(67,75,69,81))

3ème méthode :

On peut utiliser la commande matrix() qui fait appel aussi à l'instruction c().

> matrix(data=c(168,67,175,75,172,69,182,81),nr=4,nc=2,byrow=TRUE)

```
    [,1] [,2]
[1,]  168   67
[2,]  175   75
[3,]  172   69
[4,]  182   81
```

Les arguments nr et nc déclarent le nombre de lignes et de colonnes et l'argument byrow si TRUE indique que le remplissage de la matrice se fait par ligne.

- **Générer des données**

L'utilisation de variables qualitatives en statistique est très fréquente en particulier on a souvent besoin de coder certains facteurs comme le facteur individu, traitement, élevage,…etc.

- l'instruction ' : ' génère une suite d'entiers

> x <- 1 :20

x contient les entiers 1 ;2 ;3 ;4 ;……20

- la fonction seq() permet de générer une suite de nombres avec un incrément donné

> x <- seq(1,4,0.5)

x contient la suite 1 ; 1.5 ; 2 ;2.5 ;3 ;3.5 ;4

- la fonction rep() génère une suite d'un même nombre

> x <- rep(1 ;5)

x contient la suite 1 ;1 ; 1;1 ;1

- La commande gl(k,n) permet de générer n réplications d'un même nombre pour chacun des k niveaux d'un facteur. Cette fonction admet deux arguments (très utiles pour l'analyse de la variance !!) : lenght et labels.

Voici 3 exemples :

> x <- gl(3,2)

x contient : 11 22 33

> x <- gl(3,2,12)

x contient : 11 22 33 11 22 33
> x<- gl(3,2,c("A", "B","C"))
x contient : AABBCC

- **Générer des observations** (nombres aléatoires)

Le recours à la simulation de données est très fréquent en statistique, aussi R dispose-t-il d'une bibliothèque de fonctions de densité de probabilités qui permettent de simuler des échantillons d'observations provenant d'une loi de probabilités donnée.
Le tableau ci-dessous donne la liste et les commandes correspondantes des lois de probabilités usuelles :

Loi de Probabilités	Fonction
Normale	rnorm(n, mean='',sd='')
Student	rt (n, df)
Khi-deux	rchisq(n, df)
Fisher	rf(n, df1, df2)
Uniforme	runif(n, min='', max='')
Poisson	rpois(n, lambda)
Binomiale	rbinom(n, size, prob)
Hypergéométrique	rhyper(nn, m, n, k)
Statistique de Wilcoxon	rwilcox(nn, m, n), rsignrank(nn, n)

Toutes les fonctions peuvent être adaptées en changeant la première lettre de la fonction : rfonction() pour générer des observations, dfonction() pour calculer la densité de probabilités , pfonction() pour calculer la densité de probabilités cumulées et qfonction() pour calculer le quantile correspondant à une probabilité donnée.
Exemples :
> pnorm(1.96,mean=0,sd=1)
[1] 0.9750021
> qnorm(0.975,mean=0,sd=1)
[1] 1.959964
> x <- rnorm(6, mean=0,sd=1) « génère 6 observations provenant d'une loi normale centrée réduite »
> x
[1] 0.9720369 0.1537153 0.8520248 -0.5951156 2.0430104 -1.4644110

3. Quelques fonctions statistiques élémentaires

Dans ce paragraphe nous allons décrire les fonctions usuelles que l'on rencontre en statistique. Pour chaque fonction nous nous contenterons d'évoquer son utilisation avec les arguments par défaut. Ne pas hésiter à faire appel à l'aide en ligne pour avoir plus d'informations sur chaque fonction.
La fonction summary() appliqué à un fichier ou à une variable renvoie les statistiques élémentaires, min, max, moyenne et les trois quartiles.

```
> data2 <- read.table("data1.txt",h=T)
> summary(data2)
     poids           taille       fumeur     sexe        sport
 Min.   :55.00   Min.   :155.0   non:8   femme:8   Min.   :1.0
 1st Qu.:65.00   1st Qu.:165.0   oui:8   homme:8   1st Qu.:1.0
 Median :70.00   Median :170.0                     Median :1.5
 Mean   :70.38   Mean   :170.8                     Mean   :1.5
 3rd Qu.:74.25   3rd Qu.:176.3                     3rd Qu.:2.0
 Max.   :85.00   Max.   :185.0                     Max.   :2.0
>
```

On peut aussi demander les statistiques élémentaires pour le poids seulement :

```
> summary(poids)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  55.00   65.00   70.00   70.38   74.25   85.00
>
```

Les statistiques élémentaires sur la taille chez les hommes :

```
> subset(data2,sexe=="homme")
  poids taille fumeur  sexe sport   etat
1    85    184    oui homme     1 malade
2    65    175    oui homme     1 malade
3    74    180    oui homme     2  gueri
4    79    175    oui homme     2 malade
5    71    165    non homme     1  gueri
6    80    185    non homme     1  gueri
7    75    180    non homme     2 malade
8    69    155    non homme     2 malade
> summary(poids)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  55.00   65.00   70.00   70.38   74.25   85.00
>
```

Les statistiques élémentaires sur la taille suivant le sexe :

```
> tapply(poids,sexe,summary)
$femme
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  55.00   63.75   66.00   66.00   70.00   74.00

$homme
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  65.00   70.50   74.50   74.75   79.25   85.00

>
```

Le tableau suivant donne un descriptif des différentes fonctions statistiques usuelles :

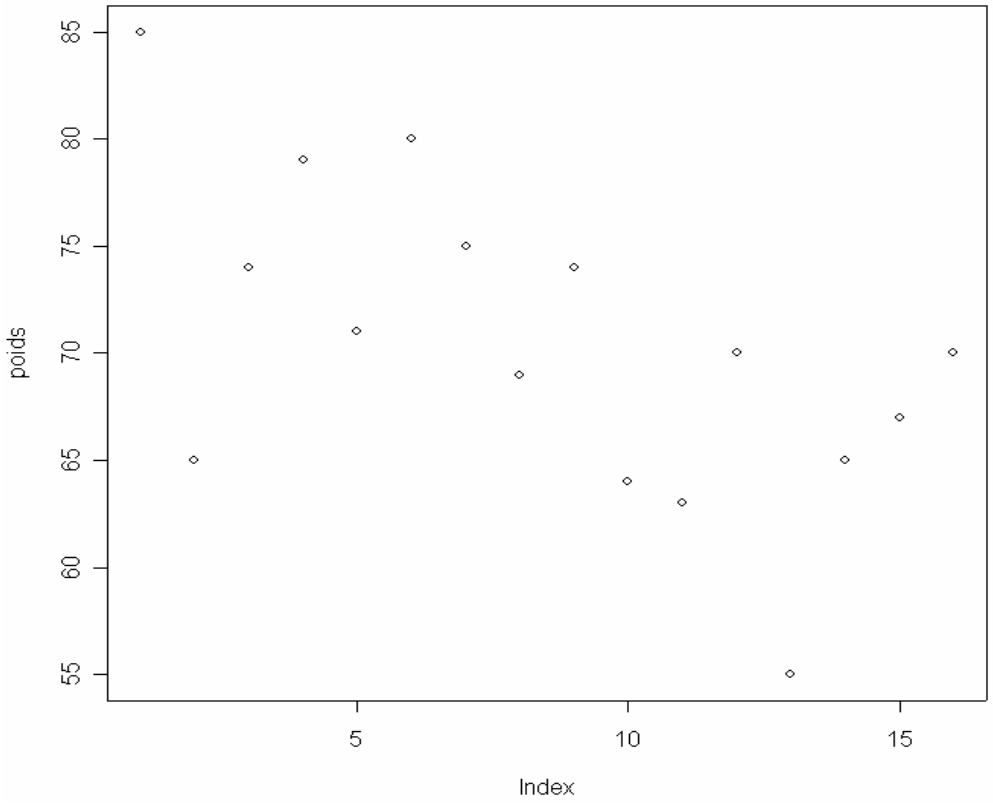
fonction	description
sum(x)	la somme des éléments de x
prod(x)	les produits des éléments de x
max(x)	la plus grand des éléments de x
min(x)	le plus petit des éléments de x
range(x)	l'étendue des éléments de x
length(x)	le nombre d'éléments de x
mean(x)	la moyenne des éléments de x
median(x)	la médiane des éléments de x
var(x)	la variance des éléments de x (calculée avec n-1)
sd(x)	l'écart-type de x
quantile(x)	les 3 quartiles de x
cov(x)	matrice de variances-covariances
cor(x)	matrice des corrélations
cov(x,y)	covariance entre x et y
cor(x,y)	corrélation entre x et y
sort(x)	ordonne les éléments de x et y
table(x,y)	table de contingence

4. Faire des graphiques avec R

Comme pour les statistiques élémentaires, la fonction ' plot 'est une fonction générique pour les graphiques. Cette fonction admet plusieurs arguments (pour s'en convaincre taper ?plot), nous allons décrire certains d'entre eux.

```
> data2 <-read.table("data1.txt",h=TRUE)
> plot(poids)
>
```

A l'exécution de cette commande le graphique s'affiche dans une fenêtre :



et en cliquant avec le bouton droit de la souris sur le graphique vous pouvez le sauvegarder dans un fichier (2 types de format proposés) ou bien le copier et le coller par exemple dans un document Word.

Voici les différentes commandes des graphiques usuels :

Fonction graphique	description
plot(x)	graphe des valeurs de x (en y) contre leurs rangs (en x)
plot(x,y)	graphe de x contre y
coplot(x~y|z)	graphe de x contre y pour chaque valeur de z
stripchart(x)	consiste à mettre les points sur un axe
hist(x, nclass=..)	histogramme de x
qqnorm(x)	quantiles de x en fonction de celles attendues d'une loi normale (droite d'Henry)
boxplot(x)	boite à moustaches de x
piechart(x)	diagramme en camembert de x
boxplot(x~y)	boite à moustaches de x pour chaque niveau du facteur y

Pour avoir la liste de toutes les options des commandes graphiques et leur définition vous pouvez utiliser help(par) ou ?par.
En voici quelques unes avec la fonction plot() :

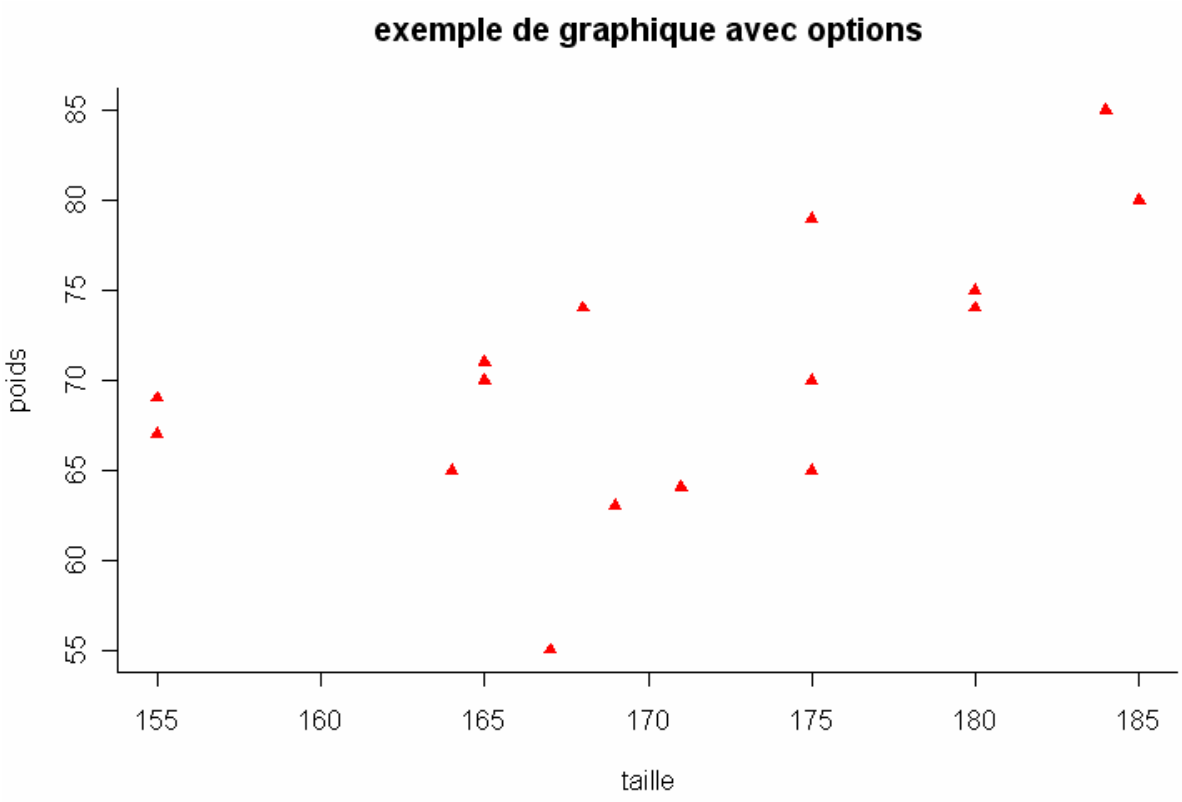
plot(x,y. type= " ", xlab= " titre de l'axe des x ", ylab="titre de l'axe des y", xlim=c(-a,a), ylim=c(-b,b), pch=--, col=" ", bty=" ", tcl=--, main="titre du graphique", las=--, cex=--, lty=--)

type : spécifie le type de graphique, "p" : points, "l" : lignes, "b" : points connectés par des lignes, "o" lignes recouvrant les points, "h" : lignes verticales, "s" : escaliers, les données étant représentées par le haut des lignes, " S" : les données étant représentées par le bas des lignes.
xlim, ylim : définissent les limites de l'axe des x et des y
pch : spécifie le symbole utilisé pour représenter les points. Tous les symboles ont un code, • (20), ◊(5), □ (22), ∇ (6), ⊕ (10), ⊗ (13), ♦ (18), ∗ (8), ▲ (17), ○ (1)…etc.)
col : la couleur du symbole utilisé
bty : spécifie la forme du cadre du graphique ("o","u","l","7","c","]"), la forme du carde ressemblant au caractère choisi. Le caractère "n" supprime le cadre du graphique
las : spécifie la disposition des annotations des axes : 0, parallèles aux axes, 1 horizontales, 2 perpendiculaires aux axes, 3 verticales)
cex : contrôle la taille des symboles.
lty : spécifie le type de ligne tracée (1 : continue, 2 : tiret, 3 : gras, 4 : point et tiret5 : tiret long, 6 : tiret long et court)
tcl : spécifie la longueur du trait de graduation par défaut tcl=-0.5
cex.axis, cex.lab, cex.main, contrôlent respectivement la taille des annotations des axes, la taille des titres d'axes et la taille du titre du graphique

Un exemple de graphique :

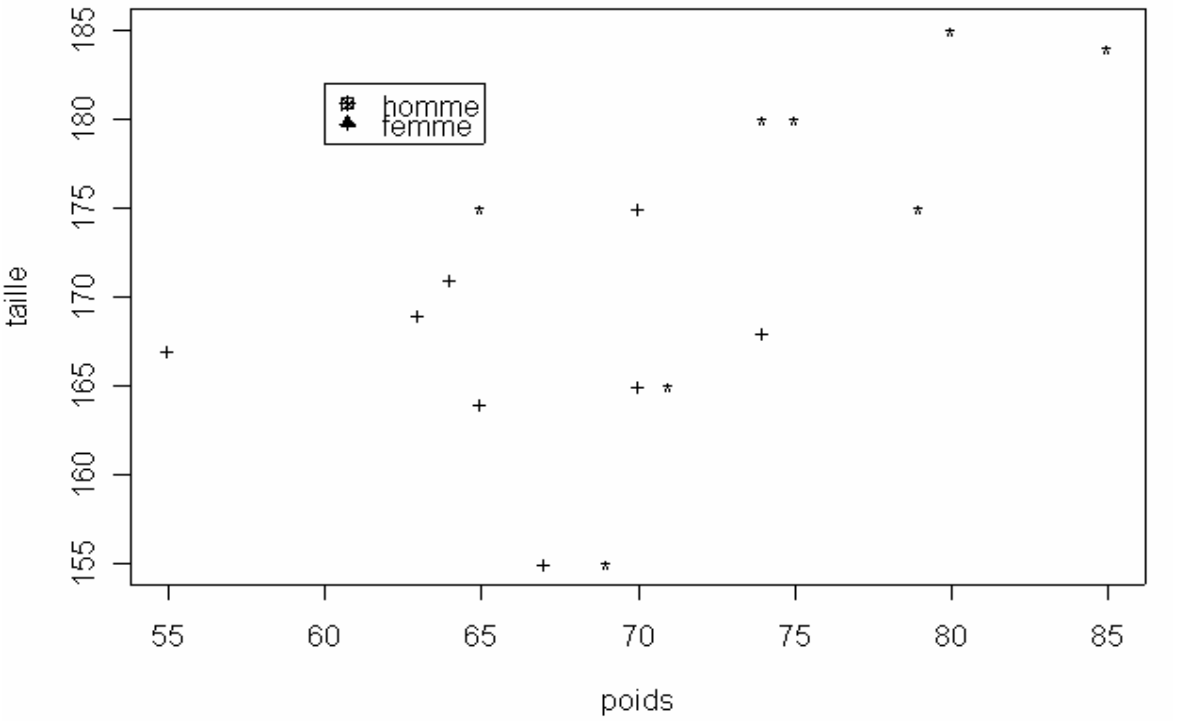
```
> data2 <-read.table("data1.txt",h=TRUE)
> attach(data2)
> plot(taille,poids,xlab="taille",ylab="poids",pch=17,col="red",bty="l",main="exemple de graphique avec options")
>
```

Le résultat obtenu avec cette commande est donné dans la figure ci-dessous.



On peut aussi représenter la taille en fonction du poids et par sexe et rajouter une légende :

```
> symboles[sexe=="homme"] <- "*"  # à part les hommes qui sont "*"
> legend(60,182,c("homme","femme"),pch=c(8,3))
> symboles <- rep("+",length(taille))  # à priori, tous sont représentés par des "+"
> symboles[sexe=="homme"] <- "*"  # à part les hommes qui sont "*"
> plot(taille~poids,pch=symboles)
> legend(60,182,c("homme","femme"),pch=c(8,3))
>
```



Il existe dans R d'autres commandes graphiques qui permettent de rajouter ou de modifier certaines options une fois que le graphique est fait.

points(x,y) :	ajoute des points au graphique
lines(x,y) :	ajoute des lignes
text(x,y,labels,..) :	ajoute un texte au coordonnées (x,y)
abline(a,b) :	trace une ligne de pente b et d'ordonnée à l'origine a
abline(h=y) :	trace une droite parallèle à l'axe des x
abline(v=x) :	trace une droite parallèle à l'axe des y
abline(lm(y~x)) :	trace la droite de régression de y sur x
legend(x,y) :	ajoute une légende aux coordonnées (x,y) avec le symbole donné par legend.

5. Test, Anova et Régression linéaire avec R

Dans ce paragraphe nous allons présenter à travers des exemples les tests les plus utilisés en statistique inférentielle. Cette liste n'est pas exhaustive, vous pouvez utiliser l'aide en ligne pour plus de détails sur les options supplémentaires ou pour connaître la commande correspondant à un test donné.

5.1 Comparaison de deux moyennes

Dans ce paragraphe nous allons décrire comment faire un test de Student pour comparer deux moyennes. On souhaite comparer les tailles moyennes chez les hommes et les femmes, les données utilisées sont celles du fichier data1.txt (http://www.biostat.envt.fr/~lyazrhi/data1.txt).

- ✓ **Cas où les variances sont homogènes (test de Student)**

```
> t.test(taille[sexe=="homme"],taille[sexe=="femme"],var.equal=TRUE)

        Two Sample t-test

data:  taille[sexe == "homme"] and taille[sexe == "femme"]
t = 1.9498, df = 14, p-value = 0.07151
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8123133 17.0623133
sample estimates:
mean of x mean of y
  174.875   166.750

>
```

- ✓ **Cas où les variances ne sont pas homogènes (test d'Aspin-welch)**

```
> t.test(taille[sexe=="homme"],taille[sexe=="femme"])

        Welch Two Sample t-test

data:  taille[sexe == "homme"] and taille[sexe == "femme"]
t = 1.9498, df = 11.171, p-value = 0.07674
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.029431 17.279431
sample estimates:
mean of x mean of y
  174.875   166.750
```

5.2. Comparaison de deux variances

On souhaite comparer les dispersions des tailles chez les hommes et les femmes

- ✓ **Test de Bartlett**

```
> bartlett.test(taille,sexe)

        Bartlett test for homogeneity of variances

data:  taille and sexe
Bartlett's K-squared = 1.9082, df = 1, p-value = 0.1672

>
```

5.3. Comparaison de deux pourcentages

On souhaite comparer le pourcentage de malades en fonction du sexe. Pour cela on peut d'abord créer une table de contingence à l'aide de la commande table().

```
> table(sexe,etat)
        etat
sexe     gueri malade
  femme 5      3
  homme 3      5
```

- ✓ **Test du Khi-deux**

Le test du Khi-deux pour comparer les pourcentages de malades se fait de la façon suivante :

```
> chisq.test(table(sexe,etat))

        Pearson's Chi-squared test with Yates' continuity correction

data:  table(sexe, etat)
X-squared = 0.25, df = 1, p-value = 0.6171

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(table(sexe, etat))
```

- ✓ **Test de Fisher exact**

```
> fisher.test(table(sexe,etat))

        Fisher's Exact Test for Count Data

data:  table(sexe, etat)
p-value = 0.6193
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  0.2554057 32.6048425
sample estimates:
odds ratio
  2.598673


> >
```

5.4. Test de conformité

- ✓ **Test de Kolmogorov-Smirnov**

On souhaite tester la normalité de la variable taille, pour cela on utilise la commande ks.test(), cette commande permet de tester si l'échantillon provient bien d'un loi normale de moyenne m estimée par la moyenne de l'échantillon mean(var) et d'écart-type sigma estimé par l'écart-type de l'échantillon sd(var).

```
> ks.test(taille,"pnorm", mean(taille),sd(taille))

        One-sample Kolmogorov-Smirnov test

data:  taille
D = 0.1152, p-value = 0.9837
alternative hypothesis: two.sided
```

Remarque : cette commande peut-être adaptée à toute autre loi de probabilités, utiliser l'aide en ligne pour en savoir plus.

5.5. Anova à 1 facteur

Suposons que l'on veuille étudier l'effet du facteur "sport" à 4 niveaux sur la variable "poids". Pour cela il faut déclarer la variable "sport" comme un facteur à 4 niveaux à l'aide de la commande as.factor().
L'anova s'effectue à l'aide des commandes aov() et anova() :

```
> facsport <-as.factor(sport)
> facsport
 [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
Levels: 1 2 3 4
> anova(aov(poids~facsport))
Analysis of Variance Table

Response: poids
           Df Sum Sq Mean Sq F value Pr(>F)
facsport    3  43.25   14.42  0.2194  0.881
Residuals  12 788.50   65.71
>
```

5.6. Comparaisons multiples

Pour comparer plus de deux groupes, R offre la possibilité de faire des comparaisons multiples. Ces comparaisons multiples viennent à la suite d'une analyse de la variance a fin qu'elles puissent s'effectuer par rapport à la bonne résiduelle.

- **Correction de bonferroni**

```
> pairwise.t.test(poids,facsport,p.adjust.method=p.adjust("bonferroni"))

        Pairwise comparisons using t tests with pooled SD

data:  poids and facsport

  1 2 3
2 1 - -
3 1 1 -
4 1 1 1

P value adjustment method: bonferroni
>
```

Remarque : R propose d'autres corrections que celle de Bonferroni, il suffit de taper ?p.djust pour afficher la liste exhaustive de toutes les corrections proposées.

Il arrive qu'on veuille comparer deux niveaux du facteur contrôlé (contrast), pour cela on utilise la commande t.test() :

```
> t.test(poids[facsport==1],poids[facsport==4])

        Welch Two Sample t-test

data:  poids[facsport == 1] and poids[facsport == 4]
t = 0.3385, df = 3.718, p-value = 0.7532
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -13.04332  16.54332
sample estimates:
mean of x mean of y
    72.00     70.25
```

- **Test de Tukey HSD (Honest Significant Differences)**

R offre la possibilité d'utiliser d'autres tests de comparaisons multiples, par exemple le test de Tukey HSD. On commence par une analyse de la variance puis le test de Tukey :

```
> r <-aov(poids~facsport)
> TukeyHSD(r,"facsport")
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = poids ~ facsport)

$facsport
     diff       lwr      upr
2-1 -0.50 -17.51731 16.51731
3-1 -4.25 -21.26731 12.76731
4-1 -1.75 -18.76731 15.26731
3-2 -3.75 -20.76731 13.26731
4-2 -1.25 -18.26731 15.76731
4-3  2.50 -14.51731 19.51731
```

5.7. ANOVA à 2 facteurs

Dans ce paragraphe on se propose d'étudier l'effet des facteurs "sexe" et "fumeur" avec une éventuelle interaction sur la variable "poids". Comme pour l'ANOVA à 1 facteur, il convient de déclarer les variables "sexe" et "fumeur" comme facteurs.

```
> facsport <-as.factor(sport)
> facsexe <-as.factor(sexe)
> facfumeur <-as.factor(fumeur)
> r <-aov(poids~facsexe*facfumeur)
> anova(r)
Analysis of Variance Table

Response: poids
                 Df Sum Sq Mean Sq F value  Pr(>F)
facsexe           1 306.25  306.25  7.4544 0.01826 *
facfumeur         1  30.25   30.25  0.7363 0.40766
facsexe:facfumeur 1   2.25    2.25  0.0548 0.81891
Residuals        12 493.00   41.08
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

5.8. Régression linéaire

La régression linéaire dans R se fait à l'aide de la commande lm (linear model). Nous allons l'illuster à travers un exemple :
Supposons que l'on veuille faire une régression linéaire de la taille sur le poids (fichier data1.txt) :

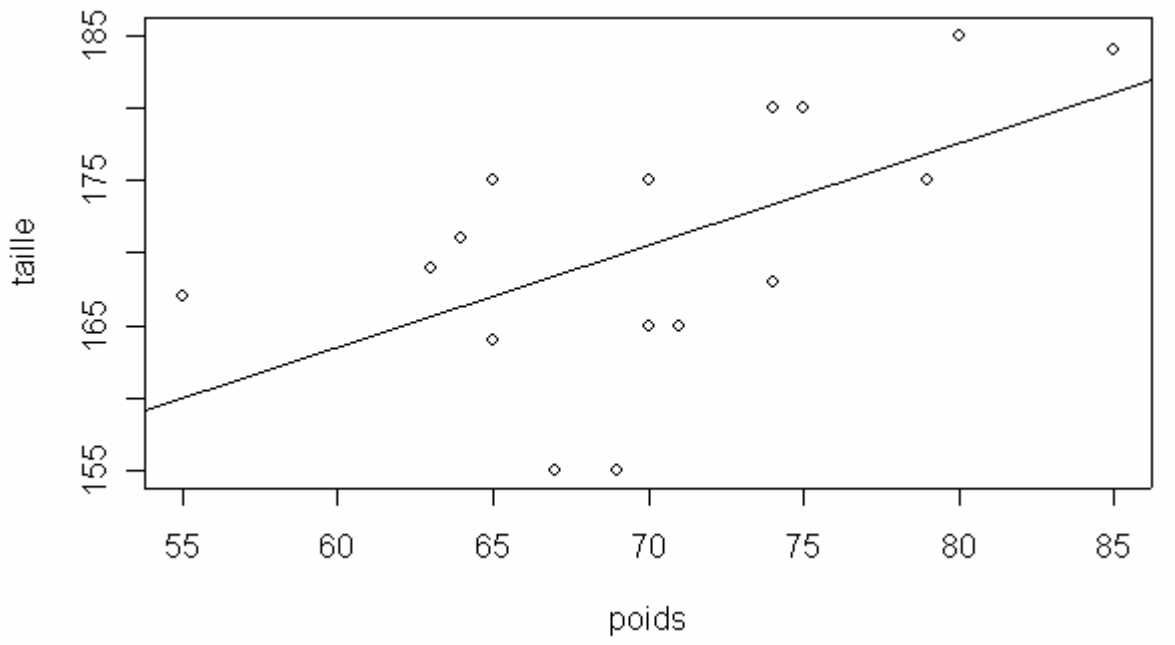
```
> data2 <- read.table("data1.txt",h=TRUE)
> attach(data2)
> names(data2)
[1] "poids"  "taille" "fumeur" "sexe"   "sport"  "etat"
> regression <-lm(taille~poids)
> summary(regression)
```

Remarque : Par défaut la régression est faite avec ordonnée à l'origine.
Les résultats affichés par sont comme suit :

```
Call:
lm(formula = taille ~ poids)

Residuals:
    Min      1Q  Median      3Q     Max
-14.840  -5.419   3.124   6.094   7.988

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 121.0508    18.7796   6.446 1.53e-05 ***
poids         0.7071     0.2655   2.664   0.0185 *
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 7.656 on 14 degrees of freedom
Multiple R-Squared: 0.3363,     Adjusted R-squared: 0.2889
F-statistic: 7.095 on 1 and 14 DF,  p-value: 0.01853
```

Pour tracer la droite de régression il suffit de taper la commande suivante :

```
> plot(taille~poids)
> abline(lm(taille~poids))
>
```



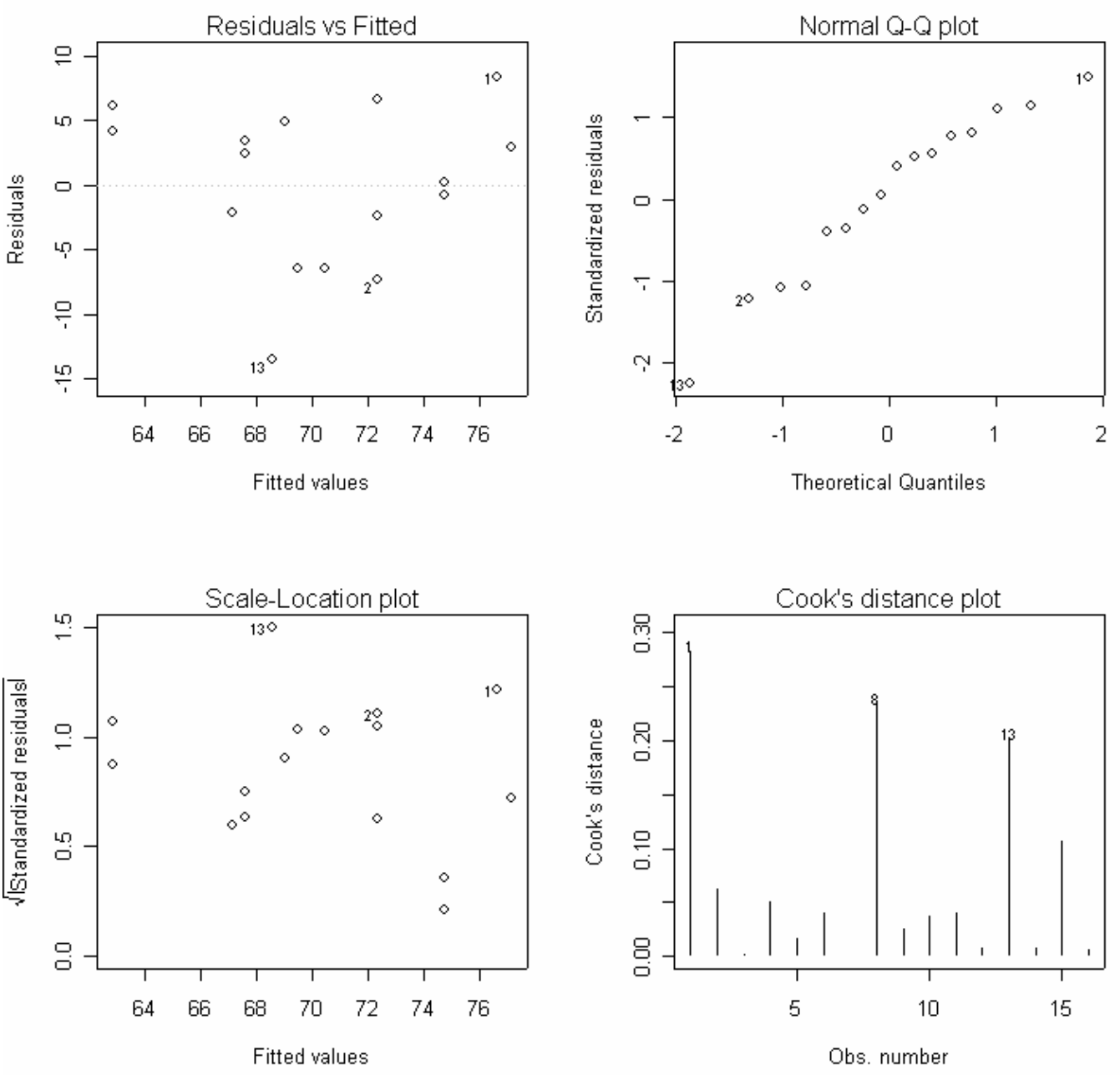
R offre la possibilité de faire les différents graphiques usuels pour valider les conditions nécessaires à une régression linéaire. Ces graphiques sont au nombre de quatre, et pour pouvoir les visualiser simultanément dans la même fenêtre graphique on utilise la commande mfrow=() :

```
> par(mfrow=c(2,2))
> plot(regression)
>
```

mfrow=c(2,2) , signifie qu'on divise la feuille graphique en 4 graphiques, le résultat est donné ci-dessous.

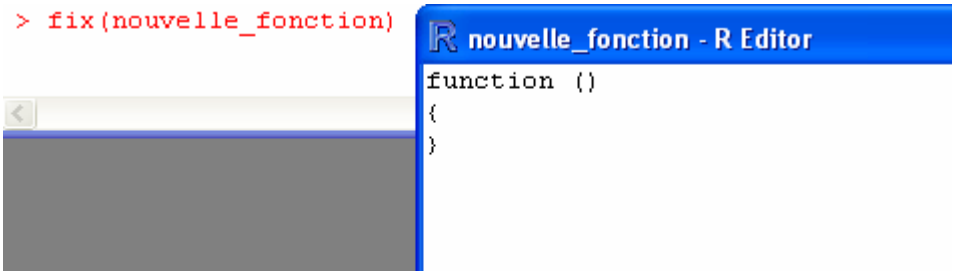
Remarque : Pour revenir à l'affichage par défaut de la fenêtre graphique, il suffit de tape la commande

```
> par(opar)
```



6 Ecrire une fonction dans R

Il est possible d'écrire une fonction (macro) personnalisée constituée d'une suite de commandes à l'aide d'un éditeur de texte ou tout simplement directement dans l'éditeur de fonction de R. Pour cela il suffit de taper la commande fix(nom de la fonction) qui ouvre automatiquement l'éditeur de fonction de R :



La définition d'une nouvelle fonction se fait sous la forme :
nom_de la fonction <- function(arg1,arg2,.....)
{
bloc d'instructiuons
}

Remarque : Si la nouvelle fonction n'a pas besoin d'arguments pour s'exécuter, remplacer fuction(arg1,arg2,…) par function()

En voici un exemple :

```
fregresssion <-function()
{
par(mfrow=c(2,2))
x <-rnorm(12,6)
y <- rnorm(21,9)
plot(x,y)
abline(lm(y~x))
hist(x)
hist(y)
}
```

Pour exécuter la fonction fregression, il suffit de taper fregression()