

# **FORMATION AU LOGICIEL R**

**(durée : 2 jours)**

version du 04 Novembre 2011



**André Bouchier**

© 2006-2011, André Bouchier (4 Novembre 2011)



<http://creativecommons.org/licenses/by-nc-sa/2.0/fr/>

Le document « **Formation au logiciel R** » by [A.Bouchier](#) est mis à disposition selon les termes de la licence *Creative Commons Paternité-Pas d'Utilisation Commerciale-Partage des Conditions Initiales à l'Identique 2.0 France*.

# 1-Qu'est ce que R

- R : logiciel multiplateforme
- R : système statistique et graphique
- R : un logiciel et un langage
- R : logiciel libre
- R : un logiciel gratuit
- R : un logiciel au développement très actif

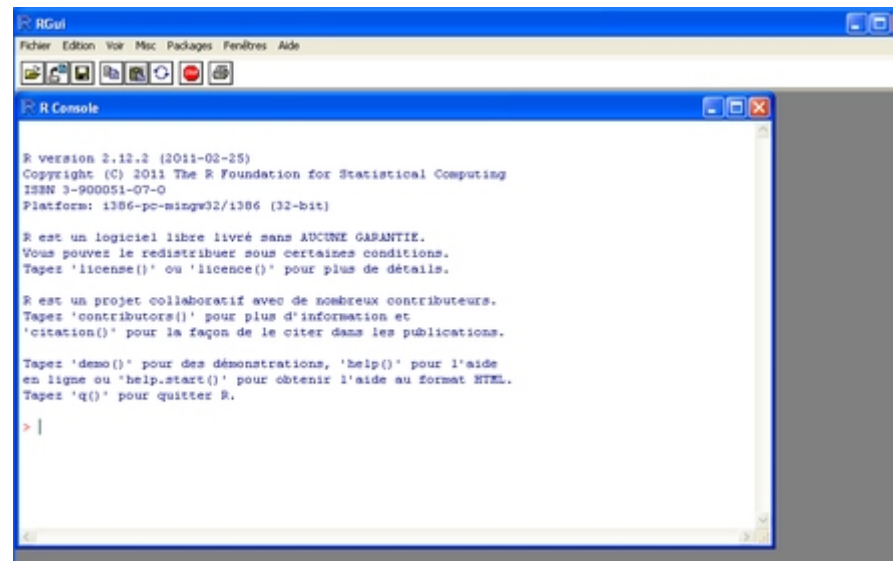
## 2-Installer le logiciel

- Obtenir le logiciel (version Windows)

<http://cran.r-project.org/bin/windows/base/R-2.14.0-win.exe>

- Installer le logiciel

Lancez le programme **R-2.14.0-win.exe**, puis suivez les instructions affichées à l'écran.

The image shows a screenshot of the R software interface. At the top is the 'RGui' window with a menu bar (Fichier, Edition, Voir, Misc, Packages, Fenêtres, Aide) and a toolbar. Below it is the 'R Console' window, which displays the R startup message. The text in the console includes the version (2.12.2), copyright information (© 2011 The R Foundation for Statistical Computing), and platform details (i386-pc-mingw32/i386 (32-bit)). It also provides instructions on how to use R, such as typing 'license()' for the license, 'contributors()' for contributors, 'demo()' for demonstrations, 'help()' for help, 'help.start()' for online help, and 'q()' to quit R. The prompt '> |' is visible at the bottom of the console window.

```
R version 2.12.2 (2011-02-25)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

> |
```

### 3- Prise en main

- Taper **Ctrl L** pour nettoyer la fenêtre 'Rconsole'
  - Faire des opérations :  $5+9$      $10^2$      $2^{0.5}$     `sqrt(2)`
  - Utiliser la flèche (clavier)  $\uparrow$  pour faire défiler les commandes déjà tapées
  - La souris permet de sélectionner (de copier-coller) les lignes dans Rconsole
  - Utiliser les parenthèses

$$4+9*2-1 = 21$$

$$(4+9)*2-1 = 25$$

$$(4+9)*(2-1) = 13$$

## 4-Stocker les résultats dans des variables

- On construit une flèche avec < et -

a<-10

b<-3

c<-a+b

ou

a+b->c

- Afficher le contenu de la variable c

c

[1] 13

- l'ensemble peut s'écrire :

a<-10 ; b<-3 ; c<-a+b ; c

## 5-Quelques opérations

- **Arithmétique**

+ - \* / ^ (puissance)

- **Logique**

> < <= >= == (égal) != (différent) & (et) | (ou) ! (non) xor (ou exclusif)

- **exemple**

$8^{(1/3)}$  = racine cubique de 8



**Attention aux parenthèses !**

$8^{1/3}$  = 2.666667

$8^{(1/3)}$  = 2



*Pour citer R dans une publication : citation()*

## 6-Le répertoire de travail

- Par défaut, R lit et écrit dans le répertoire de travail
- Connaître le répertoire de travail de R :

```
getwd()
```

```
[1] "C:/Documents and Settings/bouchier/Mes documents"
```



pour retrouver vos données soyez attentif au répertoire de travail

- Changer le répertoire de travail de R :

Utiliser le menu : fichier-> 'Changer le répertoire courant'



## 7-Changer de répertoire de travail : fonction setwd()

- Le chemin du répertoire de travail peut être fastidieux à écrire  
`setwd("C:/Documents and Settings/bouchier/Mes documents/Enquetes")`

- Deuxième solution :

`setwd("~")` # le répertoire perso de l'utilisateur est : ~  
`setwd("~/Enquetes")` # erreur si ce répertoire n'existe pas

- Remarque : pour connaître le contenu d'un répertoire : **dir()**

**dir("~/rgis")**

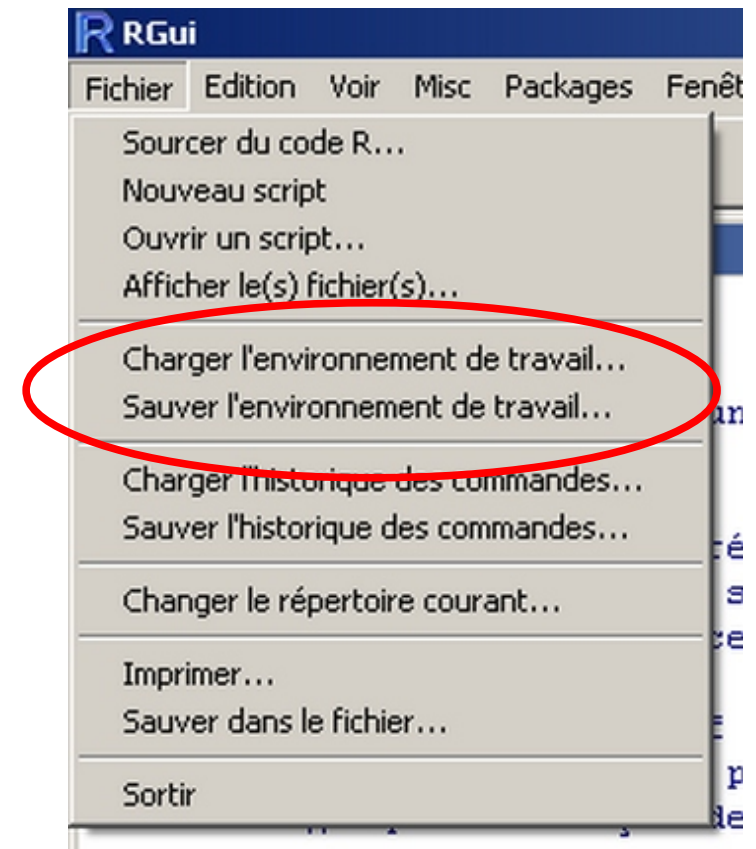
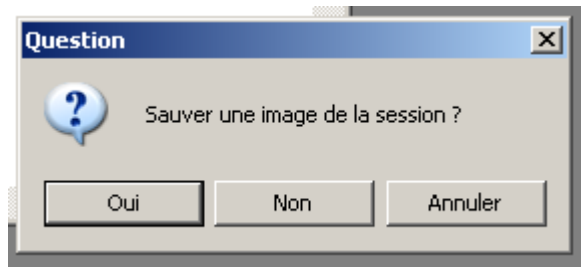
[1]	"Carto.pdf"	"departement"
[3]	"FRA.dbf"	"FRA.prj"
[5]	"FRA.qpj"	"FRA.shp"
[7]	"FRA.shx"	"france"
[9]	"LR.png"	"maps.pdf"

**dir("~/rgis", pattern=".shp")**

[1] "FRA.shp"

## 8-L'environnement de travail

- Les données créées au cours d'une session peuvent être sauvegardées. Les fichiers de données R portent l'extension *.Rdata*
- La commande « sauver l'environnement de travail » copie toutes les données en mémoire dans un fichier à l'extension *.Rdata*
- À la fermeture, R vous propose de sauvegarder l'environnement de travail.



## 9-Lecture de données (données d'exemple)

- R est fourni avec des fichiers de données d'exemple

liste des fichiers disponibles : `data()`

- Charger en mémoire le tableau de données "iris"

`data(iris)`

- Que contient ce tableau ?

`iris` ou `head(iris)`

- Une présentation graphique

`pairs(iris)`

- En savoir plus sur ce tableau de données

`?iris`

## 10-Lecture de données (format binaire R)

- On peut stocker des données au format R (extension .Rdata)
- Lecture du fichier de données R « voit2005.Rdata »

```
load(file.choose())
```

- Le fichier de données a-t-il été chargé en mémoire ?

```
ls()
```

```
[1] "voit2005"
```

- Sauver un tableau de données

Un fichier de données .Rdata peut contenir plusieurs data.frames

```
save(iris, voit2005, file= "test.Rdata")
```

```
dir(pattern=".Rdata")
```

```
[1] "test.Rdata"
```

# 11-Objets en mémoire

- La fonction `ls()` permet de lister les objets en mémoire

```
ls()
```

- Plus de détails avec `ls.str()`

```
ls.str()
```

```
print(ls.str(), max.level = 0)
```

- Effacer des objets en mémoire

```
rm(a,b)
```

- Effacer tous les objets en mémoire

```
rm(list=ls())
```

## 12-Types de données : les classes

- **Vector** : une variable dans le sens général

`is.vector(x)` ; `as.vector(x)`

- **Factor** : variable qualitative (facteur)

`is.factor()` ; `as.factor()`

- **Array** : une matrice (données du même type)

`is.matrix()` ; `as.matrix()`

- **Data.frame** : un jeu de données composé de vecteurs de même dimension.

`is.data.frame()` ; `as.data.frame()`

## 13-Connaitre le types de données

- Une classe d'objet peut être composé de données de différents types  
numérique, caractère, entier, réel, logique
- Utiliser la fonction `typeof()` ou la fonction `str()`

```
a<-"inra"
```

```
typeof(a)
```

```
[1] "character"
```

```
a<-4
```

```
typeof(a)
```

```
[1] "double"
```

```
a<-as.integer(a)
```

```
typeof(a)
```

```
[1] "integer"
```

## 14-Utiliser un data.frame (1)

- Connaître le nom des variables du data.frame

```
names(voit2005)
```

- Les dimensions du data.frame [ lignes , Colonnes ]

```
dim(voit2005)
```

```
dim(voit2005)[1] # nombre de lignes
```

```
dim(voit2005)[2] # nombre de colonnes
```

- Le nombre de colonnes

```
length(voit2005)
```



## 15-Utiliser un data.frame (2)

**Il existe plusieurs façons d'accéder aux variables du data.frame**

- En précisant le nom du data.frame pour chaque variable

```
plot(voit2005$Longueur, voit2005$Largeur)
```

- Par leur numéro ( voir *names(voit2005)* )

```
plot(voit2005[,3], voit2005[,4])
```

- En attachant le data.frame (*uniquement pour la lecture*)

```
attach(voit2005)  
  plot(Longueur, Largeur)  
detach()
```

## 16-Extraire des données d'un data.frame ( indexation )

- Les 5 premières lignes avec les variables 2 à 5  
`voit2005[1:5, 2:5]`
- Les 5 premières lignes mais avec les variables 1, 3 et 6  
`voit2005[1:5, c(1,3,6)]`
- Toutes les lignes (sauf la 3<sup>ème</sup>), toutes les variables (sauf la 1<sup>ère</sup>)  
`voit2005[-3, -1]`
- Les 5 premières lignes, toutes les variables sauf les n° 1, 3, et 5  
`voit2005[1:5, c(-1,-3,-5)]`
- On conserve les individus pour lesquels la puissance > 10  
`voit2005[voit2005$Puissance>10, ]`

## 17-Extraire des données d'un data.frame ( fonction subset() )

- Ne conserver que les variables "Puissance" et "Vitesse" et que les véhicules dont la vitesse maxi est supérieure à 200 km/h

```
subset(voit2005, Vitesse > 200, select = c(Puissance, Vitesse))
```

				Puissance	Vitesse
Alfa-Romeo	155	2.0		10	205
Alfa-Romeo	164	2.5	T	7	202
BMW	730i			16	222
Citroen	XM	2.0i		11	201
Citroen	XM	V6		16	222
Ford	Scorpio	2900i		15	201
Peugeot	605	Sv24		16	235

**Exercice :** en utilisant la fonction subset(), sélectionnez les véhicules dont la consommation est inférieure à 6l/100 et la puissance fiscale égale = 4 CV

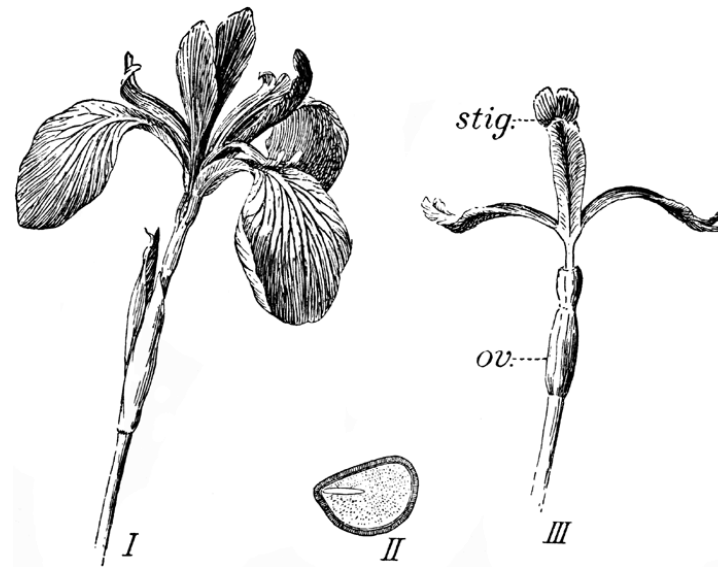
## 18-Identifier les lignes du tableau de données

- Toutes les lignes d'un data.frame ont un identificateur unique  
`row.names(voit2005)`
- On peut accéder à un individu en particulier  
`voit2005["Renault 21 Prima TD", ]`
- Ou à une collection d'individus  
`voit2005[c("Renault 21 Prima TD","BMW 518i"), 1:3]`



## 19-Exercice

- Représentez graphiquement l'ensemble des relations x-y des données quantitatives du data.frame `iris`
- Extraire les données de la variété `virginica`. Quelles sont les moyennes des variables pour cette variété ( fonction `mean()` )



## 20-Types de données dans un vecteur

```
> is.numeric(iris$Petal.Length)
[1] TRUE
```

# test sur toutes les variables : sapply() retourne une liste

```
> sapply(iris, is.numeric)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          TRUE          TRUE          TRUE          TRUE          FALSE
```

# numéro des variables numériques

```
> which(sapply(iris, is.numeric))
Sepal.Length Sepal.Width Petal.Length Petal.Width
           1           2           3           4
```

# noms des variables numériques

```
> names(which(sapply(iris, is.numeric)))
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
```

# Récupération des numéros de variables

```
> as.vector(which(sapply(iris, is.numeric)))
[1] 1 2 3 4
```

# Représentation graphique

```
> numero<-which(sapply(iris, is.numeric))
> pairs(iris[, numero])
```

## 21-Générer une séquence

- Répéter une valeur

```
rep("non", 20)
```

- Séquence simple

```
z<-2:12
```

- Attention,

avec `x<-10` comparez `1:x-1` et `1:(x-1)`

- On peut utiliser la fonction `seq()` pour modifier l'incrément

```
seq(1, 9, 0.5)
```

- La fonction `seq()` peut calculer les éléments de la série

```
seq(length=8, from=1, to=5)
```

```
[1] 1.000000 1.571429 2.142857 2.714286 3.285714 3.857143 4.428571 5.000000
```

## 22-Les séquences aléatoires

- Loi normale

```
x<-rnorm(1000,mean=0, sd=1)
```

- Loi uniforme

```
x<-runif(100,min=2,max=4)
```

- On peut vérifier la précision de ces fonctions avec

```
mean(x) ;var(x)
```

- **exercice** : essayez avec 10 , 100 individus, puis avec 10000 et 100000



## 23-Demander de l'aide

- En savoir plus sur une fonction

?mean

?median

?IQR

- Comment importe-t-on des fichiers textes ?

?read.table

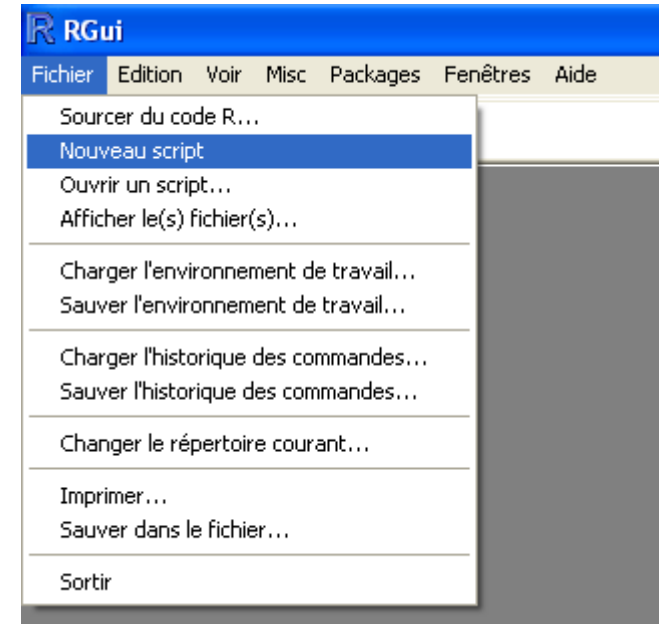
## 24-Écrire des scripts avec un éditeur de texte

### Première solution : éditeur de script R

- Ouvrez l'éditeur de texte fourni par R
- Saisissez vos commandes

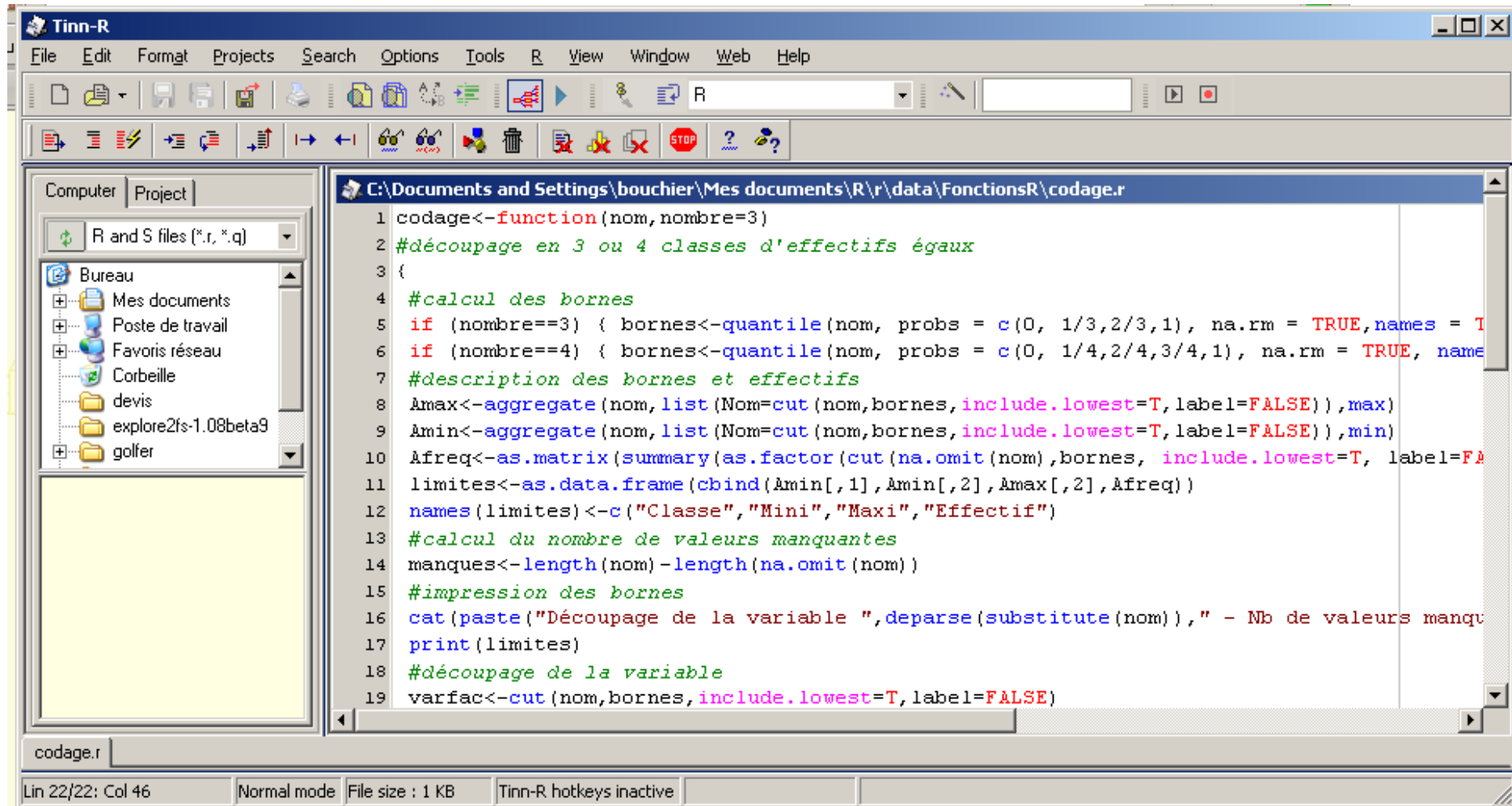
```
# 1000 valeurs suivant une loi normale {0,1}  
x<-rnorm(1000,mean=0, sd=1)  
#Calcul de la moyenne  
mean(x)  
#Calcul de la variance  
var(x)
```

- Transférez-les dans la fenêtre R
  1. Ctrl-A tout sélectionner
  2. Ctrl-R coller vers R-console



## Deuxième solution : Tinn-R (uniquement pour Windows)

- disponible à cette adresse <http://www.sciviews.org/Tinn-R/>
- avec coloration syntaxique et copie-coller direct vers R

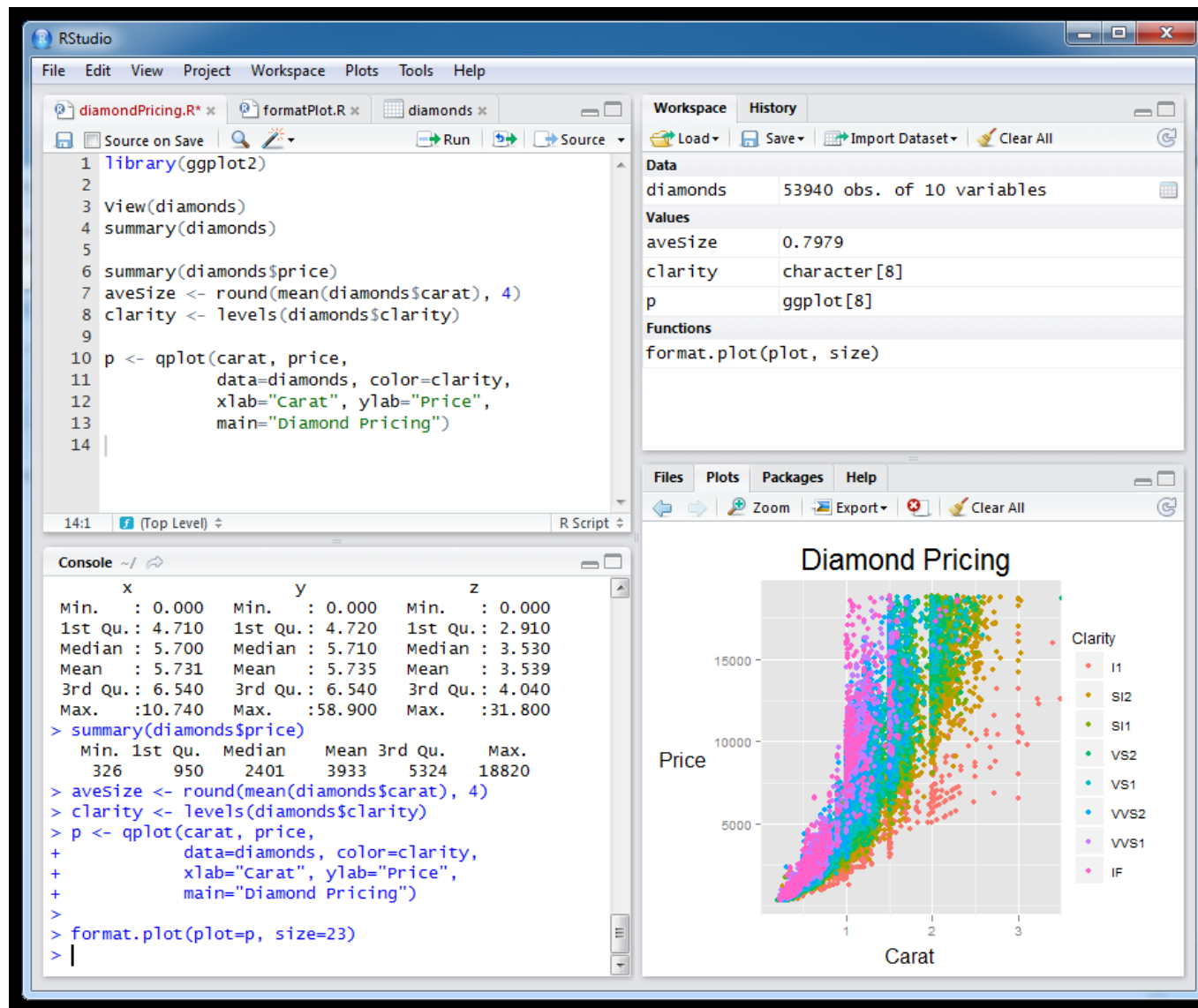


The screenshot shows the Tinn-R application window. On the left is a file explorer showing the 'Computer' and 'Project' views. The 'Project' view is active, showing a list of files and folders including 'Bureau', 'Mes documents', 'Poste de travail', 'Favoris réseau', 'Corbeille', 'devis', 'explore2fs-1.08beta9', and 'golfer'. The main area is a code editor displaying an R script named 'codage.r'. The script defines a function 'codage' that takes 'nom' and 'nombre' as arguments. It uses various R functions like 'quantile', 'aggregate', 'as.matrix', 'summary', 'as.factor', 'cut', 'length', and 'paste' to process data. The code is color-coded: comments are green, function names and variables are blue, and logical values and strings are red. The status bar at the bottom indicates 'Lin 22/22: Col 46', 'Normal mode', 'File size : 1 KB', and 'Tinn-R hotkeys inactive'.

```
1 codage<-function(nom,nombre=3)
2 #découpage en 3 ou 4 classes d'effectifs égaux
3 {
4   #calcul des bornes
5   if (nombre==3) { bornes<-quantile(nom, probs = c(0, 1/3,2/3,1), na.rm = TRUE,names = T
6   if (nombre==4) { bornes<-quantile(nom, probs = c(0, 1/4,2/4,3/4,1), na.rm = TRUE, name
7   #description des bornes et effectifs
8   Amax<-aggregate(nom,list(Nom=cut(nom,bornes,include.lowest=T,label=FALSE)),max)
9   Amin<-aggregate(nom,list(Nom=cut(nom,bornes,include.lowest=T,label=FALSE)),min)
10  Afreq<-as.matrix(summary(as.factor(cut(na.omit(nom),bornes, include.lowest=T, label=FA
11  limites<-as.data.frame(cbind(Amin[,1],Amin[,2],Amax[,2],Afreq))
12  names(limites)<-c("Classe","Mini","Maxi","Effectif")
13  #calcul du nombre de valeurs manquantes
14  manques<-length(nom)-length(na.omit(nom))
15  #impression des bornes
16  cat(paste("Découpage de la variable ",deparse(substitute(nom))," - Nb de valeurs manqu
17  print(limites)
18  #découpage de la variable
19  varfac<-cut(nom,bornes,include.lowest=T,label=FALSE)
```

## Troisième solution : Rstudio

- multiplateforme, disponible à cette adresse : <http://rstudio.org/>



## 25-Importer les données d'un fichier texte

- Lire un fichier de données texte avec données manquantes codées M, le séparateur décimal est une virgule (fichier bledur.txt)

```
bledur <- read.table( "~/bledur.txt", header=T, na.string="M", dec=",", sep=" ")
```

- Si vous ne connaissez pas le format du fichier texte des données :

```
file.show(file.choose()) # affichez le contenu d'un fichier texte
```

- Plus simplement, de façon interactive :

```
bledur <- read.table(file.choose(), header=T, na.string="M", dec=",", sep=" ")
```



## 26-Exercices

En utilisant l'éditeur de script de R :

- importez le fichier texte `voit2005.txt`  
combien de lignes et de variables possède-t-il ?
- Quelles sont les moyennes des variables Longueur, Largeur et Surface du data.frame `voit2005` ?
- Combien de véhicules ont une cylindrée plus petite que  $1000 \text{ cm}^3$  ?
- Quel est l'écart-type de la variable « `Longueur` » ?
- Quelle est sa médiane, sa moyenne, son IQR ?

## 27-Variables du fichier exemple « bledur »

Ces données sont extraites d'une enquête agronomique

"Numero"	identifiant de la parcelle
"RDT"	rendement en grains à la récolte
"PLM"	nombre de plantes levées par m <sup>2</sup>
"ZON"	zone géographique
"ARG"	taux d'argile dans le sol
"LIM"	taux de limon dans le sol
"SAB"	taux de sable dans le sol
"VRT"	codes des 6 variétés cultivées
"PGM"	poids de 1000 grains à la récolte
"MST"	matière sèche totale à la récolte (aérien + racinaire)
"AZP"	taux d'azote dans la plante
"VRTC"	variétés cultivées après regroupement en 3 classes

## 28-Le format de données R

- Le data.frame "bledur" peut être enregistré au format R en utilisant la fonction save()

```
save(bledur, file = "bledur.Rdata")
```

- Ce fichier est sauvegardé dans le répertoire de travail par défaut

- Pour changer le répertoire de travail

```
setwd(dirname(file.choose())) # le répertoire choisi ne peut pas être vide
```

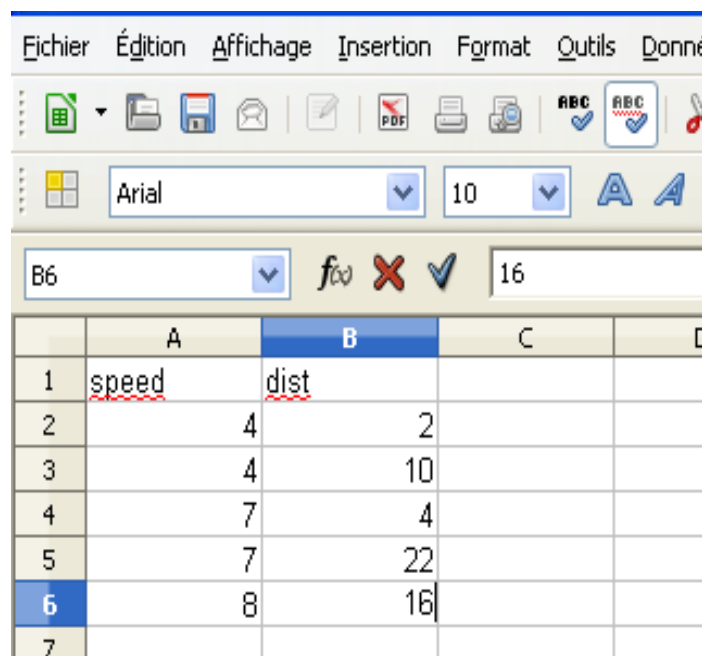
```
save(bledur, file = "bledur.Rdata")
```

- Un fichier .Rdata peut être chargé à l'aide de la fonction load()



## 29-Passage d'Excel/Open-Office à R (format csv)

- Saisissez ces données sous Excel ou OpenOffice. C'est un extrait du tableau de données cars ( ?cars - pour en savoir plus )
- Enregistrer ce tableau au format csv sous le nom de : **cars20.csv**  
Attention à bien choisir le séparateur de champs (ici un ;)
- Pour le lire, utilisez la fonction "**read.table()**"



	A	B	C	D
1	speed	dist		
2	4	2		
3	4	10		
4	7	4		
5	7	22		
6	8	16		
7				

## 30-Modifier les données d'un tableau

- Lecture du tableau

```
cars20<-read.table("~/cars20.csv", header=T, sep=";")
```

- créer une nouvelle variable vitesse en km/h

```
vitkmh<-cars20$speed*1.61
```

- l'ajouter au tableau de données

```
cars20<-data.frame(cars20, vitkmh)
```

- modifier une variable (distance en mètres)

```
cars20$dist<-cars20$dist*30.48
```

- renommer une variable

```
names(cars20)[2]<-"distmetres"
```

	speed	distmetres	vitkmh
1	4	60.96	6.44
2	4	304.80	6.44
3	7	121.92	11.27
4	7	670.56	11.27
5	8	487.68	12.88

## 31-Gérer des données

Quand c'est possible, utilisez des outils de gestion de données (*voir rodbc*)

Cependant, R permet :

- de créer un nouveau tableau de données

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

- de créer un nouveau tableau à partir d'un tableau existant

```
xnew <- edit(bledur)
```

- d'éditer un objet (le modifier)

```
cars20 <- edit(cars20)
```

R Editeur de données					
	speed	distmetres	vitkmh	var4	var5
1	4	60.96	6.44		
2	4	304.8	6.44		
3	7	121.92	11.27		
4	7	670.56	11.27		
5	8	487.68	12.88		
6					
7					

## 32-Supprimer les données manquantes

- Supprimer les données manquantes de tout le tableau

```
na.omit(voit2005)
```

- compter le nombre d'individus ayant des données manquantes

```
dim(voit2005)[1] - dim(na.omit(voit2005))[1]  
[1] 10
```

- **exercice** : lire le fichier *produit.txt*. Attention, il contient des données manquantes.

1. Quelle est la moyenne de la variable PRODUIT ?
2. Combien de lignes contiennent des données manquantes ?

## 33-Joindre des tableaux de données

- nous disposons de 2 fichiers (format csv) issus de l'INSEE :
  - 1.Diplome2008.csv : contient pour quelques départements le taux de non diplômés parmi la population de plus de 15 ans.
  - 2.Pauvrete.csv : contient pour quelques départements le taux de pauvreté au seuil de 60%
- les départements disponibles ne sont pas identiques pour les 2 fichiers
- pour joindre ces fichiers de données :
  - 1.lire les tableaux de données avec la fonction `read.table()`
  - 2.joindre ces données avec la fonction `merge()`

```
pauvrete<-read.table(file.choose(), sep=";", header=T, dec=",")  
diplome<-read.table(file.choose(), sep=";", header=T, dec=",")  
merge(pauvrete, diplome)
```

## 34-Joindre des tableaux de données (suite)

- Le résultat de `merge(pauvrete, diplome)` correspond aux 10 départements présents à la fois dans les 2 data.frame. Ils ont été automatiquement joints grâce à la colonne commune **LIBGEO**

	LIBGEO	DEP	TAUX60	CODGEO	P08_DIPL0_NSCOL15P
1	Ardèche	7	14.0	7	18.8
2	Ariège	9	16.9	9	19.1
3	Aude	11	19.3	11	21.9
4	Aveyron	12	15.0	12	16.2
5	Gard	30	18.0	30	20.2
6	Gers	32	14.9	32	17.9
7	Hérault	34	17.5	34	18.4
8	Lozère	48	15.7	48	18.2
9	Puy-de-Dôme	63	12.3	63	14.6
10	Pyrénées-Orientales	66	19.1	66	20.2

- En vous aidant de l'aide fournie par R, créez un tableau :
  - contenant tous les départements présents dans les 2 data.frame
  - ne contenant que les départements contenus dans le data.frame **"diplome"**

## 35-Décrire des variables quantitatives

```
summary(bledur[, c(2,3,5,6,7,9,10,11)])
```

```
maliste<-c(2,3,5,6,7,9,10,11) # liste des données quantitatives
```

```
summary(bledur[, maliste])
```

- Un paramètre statistique pour chaque niveau d'une variable qualitative :

```
aggregate(bledur[,maliste], list("variete"=bledur$VRTC), mean)
```

```
aggregate(bledur[,maliste], list("variete"=bledur$VRTC, "zone"=bledur$ZON), mean)
```

	variete	zone	RDT	PLM	ARG	LIM	SAB	PGM	MST	AZP
1	1	1	11.40	103.40	38.52	41.50	19.98	38.00	35.82	3.36
2	2	1	8.45	128.40	27.69	52.46	20.85	39.52	28.64	3.45
3	3	1	13.66	87.00	27.05	53.30	19.65	42.65	35.36	3.16
4	1	2	9.93	101.00	21.22	32.96	45.82	39.87	31.63	3.18
5	2	2	12.76	139.50	24.25	33.20	42.55	32.00	37.97	3.66
6	3	2	13.07	117.50	20.95	39.60	39.45	43.80	31.17	2.91
7	1	3	11.29	137.25	17.10	18.54	64.36	40.58	33.99	3.27
8	2	3	8.61	160.78	20.44	29.43	50.12	30.77	28.42	2.87
9	3	3	15.10	117.00	31.10	47.70	21.20	30.70	41.30	3.84



On peut calculer : mean, sd, max, min, median, sum, ...

## 36-Les histogrammes (1)

- Les données quantitatives

```
hist(bledur$RDT)
```

- On peut choisir le nombre de barres

```
hist(bledur$RDT, nclass=5)
```

- Pour avoir les bornes et les effectifs des classes

```
hist(bledur$RDT, nclass=5, plot=F)
```

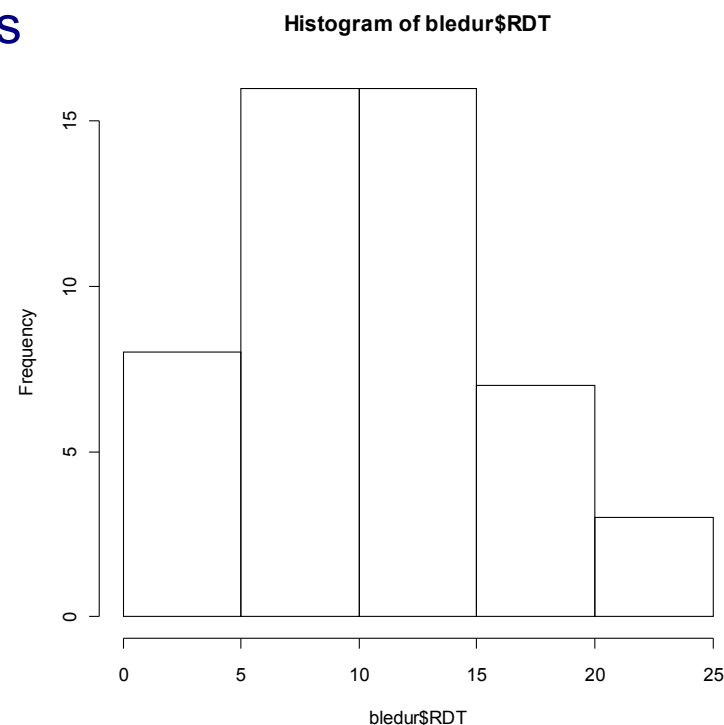
- Choisir ce dont on a besoin :

```
a<- hist(bledur$RDT, nclass=5, plot=F)
```

```
names(a)
```

```
bornes<-a$breaks
```

```
[1] 0 5 10 15 20 25
```





## 37-Les histogrammes (2)

- Attention, quelle différence entre ?

*attach(bledur)*

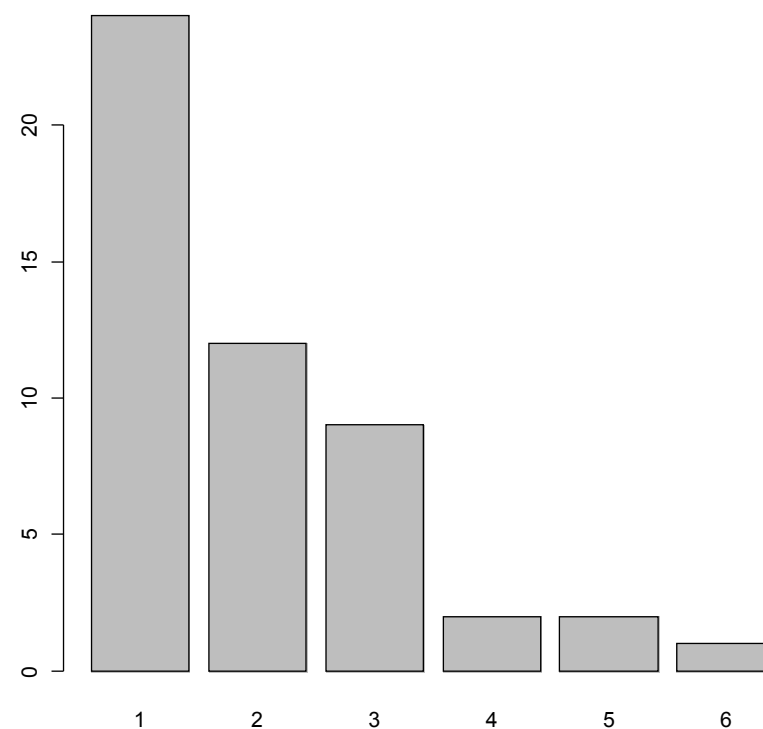
`hist(VRT)`

`barplot(VRT)`

`barplot(summary(VRT))`

`barplot(summary(as.factor(VRT)))`

*detach(bledur)*



## 38-Boîtes à pattes

- Boite simple  
`boxplot(bledur$RDT)`
- Une boite par niveau de facteur  
`boxplot(split(bledur$RDT,bledur$ZON))`  
ou  
`boxplot(bledur$RDT~bledur$ZON)` # résultat identique
- Pour rendre le graphique plus lisible, ajouter un titre  
`title("Rendement", sub="Données INRA", ylab= "Effectifs",  
xlab= "zones géographiques")`
- Représenter les individus  
`rug(bledur$RDT, side=2)`
- Exporter le graphique sous différents formats. Menus File/Save as/...

## 39-Les camemberts

- Les données doivent être préparées

```
pie(bledur$VRT) # mauvais  
pie(summary(as.factor(bledur$VRT))) # correct
```

- Attention à l'ordre d'apparition des niveaux de facteur

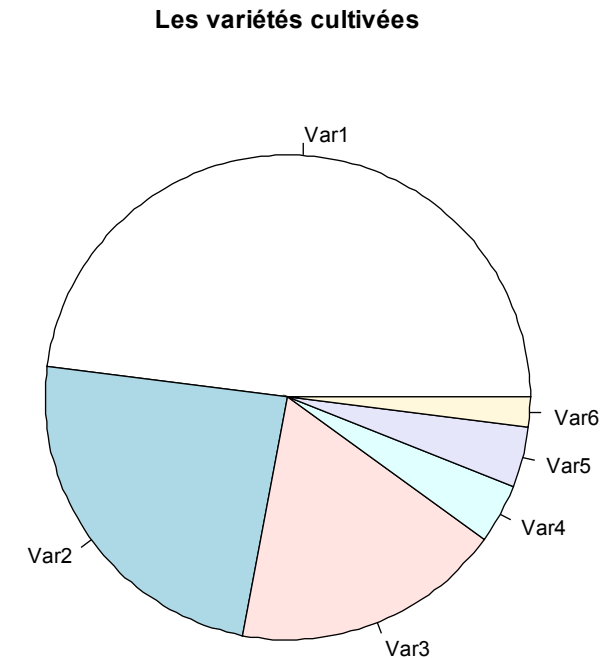
```
summary(as.factor(bledur$VRT))
```

- On ajoute un nom "explicite" pour chaque secteur

```
nom<-c("Var1","Var2","Var3","Var4","Var5","Var6")  
pie(summary(as.factor(bledur$VRT)), label=nom)
```

- On ajoute un titre principal

```
titre<-"Les variétés cultivées"  
pie(summary(as.factor(bledur$VRT)), label=nom, main=titre)
```



**i L'aide de R nous dit :** *Pie charts are a very bad way of displaying information. The eye is good at judging linear measures and bad at judging relative areas.*

## 40-Un graphique en x-y

- Simple graphique

```
plot(bledur$PLM,bledur$RDT)
```

- Graphique "commenté"

```
titre<-"Rendement*Plantes par m2"
```

```
x<-"Nombre de plants par m2" ; y<-"Rendement"
```

```
plot(bledur$PLM,bledur$RDT, main=titre, xlab=x,ylab=y)
```

- Graphique illustré

```
plot(bledur$PLM,bledur$RDT)
```

```
text(bledur$PLM,bledur$RDT,labels=bledur$ZON)
```

- **exercice** : les points et les numéros se superposent. Comment rendre les numéros plus lisibles ? (`?plot` et `?text`)

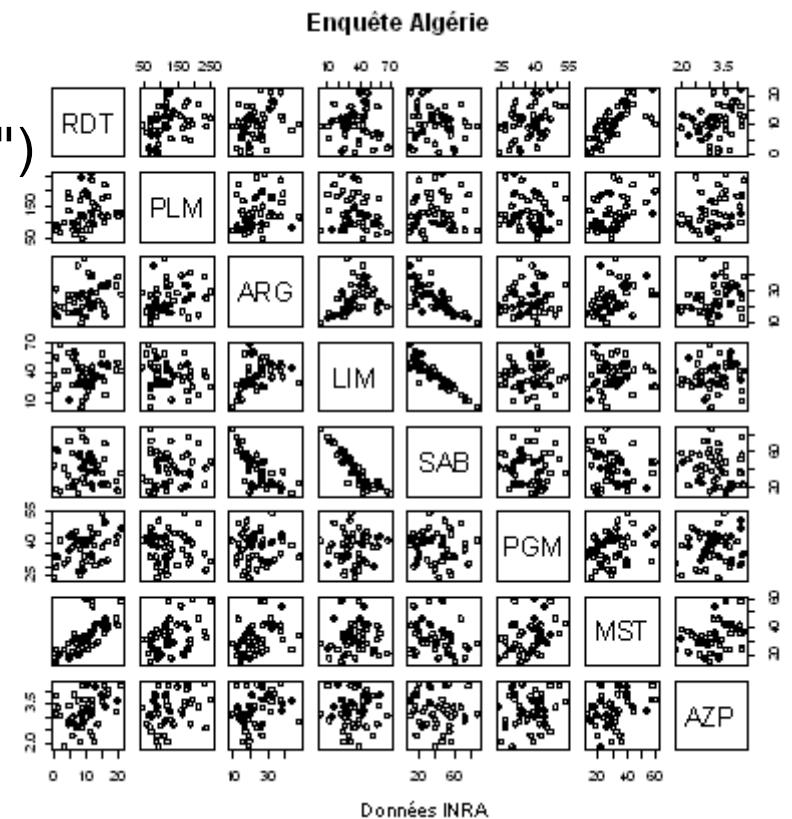
# 41-Graphique de toutes les variables quantitatives

- Vérifier la linéarité des relations entre variables avant de lancer une ACP

```
maliste<-c(2,3,5,6,7,9,10,11) # les variables quantitatives  
pairs(bledur[,maliste])  
title("Enquête Algérie ", sub="Données INRA")
```

- Comment afficher le titre sans cacher l'échelle ?

```
pairs(bledur[,maliste], main="Enquête Algérie")  
title(sub="Données INRA")
```



## 42-Le test du Khi2

- Créer le tableau croisé  
`table(bledur$ZON, bledur$VRTC)`
- Préciser le nom des variables  
`table(bledur$ZON, bledur$VRTC, dnn=c("Zones géographiques","Les variétés"))`
- Tester l'indépendance  
`tb<-table(bledur$ZON,bledur$VRTC)`  
`summary(tb)`
- On peut simuler un tirage aléatoire  
`chisq.test(tb, simulate.p.value = TRUE, B = 10000)$p.value`

Number of cases in table: 50

Number of factors: 2

Test for independence of all factors:

Chisq = 8.207, df = 4, p-value = 0.0843

Chi-squared approximation may be incorrect

## 43-Les corrélations

- Choisir les données quantitatives

```
maliste<-c(2,3,5,6,7,9,10,11)
```

- Un tableau des corrélations

```
cor(bledur[,maliste])
```

- Précisions sur les coefficients de corrélation

```
cor.test(bledur$LIM,bledur$SAB, method = "pearson")
```

```
cor.test(bledur$LIM,bledur$SAB, method = "kendall")
```

```
Pearson's product-moment correlation
```

```
data: bledur$LIM and bledur$SAB
```

```
t = -15.5096, df = 48, p-value < 2.2e-16
```

```
alternative hypothesis: true correlation is not equal to 0
```

```
95 percent confidence interval:
```

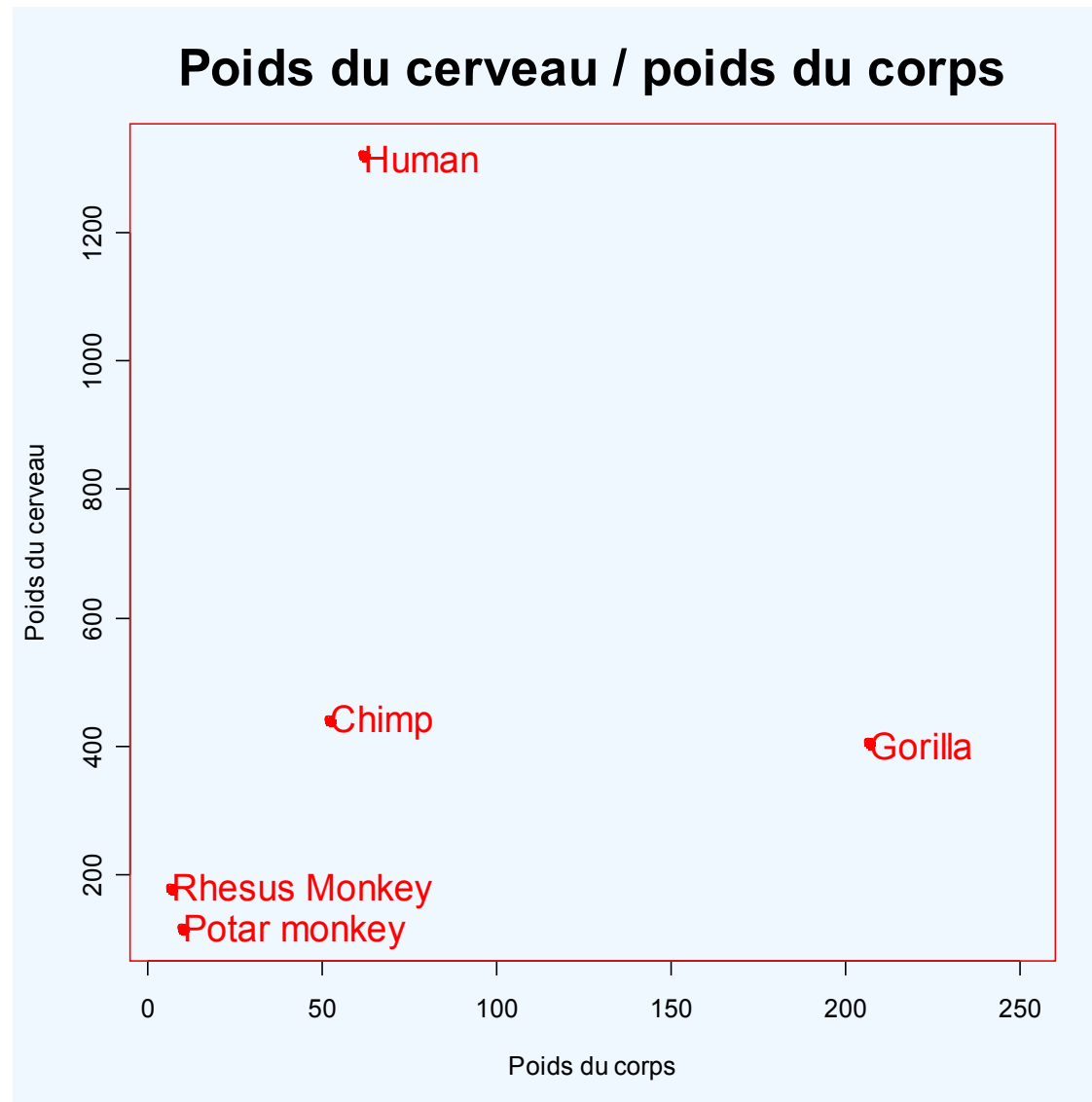
```
-0.9499646 -0.8509641
```

```
sample estimates:
```

```
cor
```

```
-0.9130445
```

## 44-Graphique : exemple





## 45-Graphiques : gestion des paramètres

- Sauver les paramètres avant de les modifier : "touche panique"

- Stocker tous les paramètres par défaut

```
old.par <- par(no.readonly = TRUE)
```

- Dessiner un graphique en modifiant les paramètres

```
par(bg="aliceblue",col="red")  
plot(.....)
```

- Initialiser les paramètres aux valeurs par défaut

```
par(old.par)
```



De nombreux exemples de graphiques sont visibles à cette adresse :  
<http://addictedtor.free.fr/graphiques/>

## 46-Graphiques : exercice

- Les données

```
Bodywt<-c( 10, 207,    62, 6.8, 52.2)
```

```
Brainwt<-c(115, 406, 1320, 179, 440)
```

```
Noms<-c("Potar monkey", "Gorilla", "Human", "Rhesus Monkey", "Chimp")
```

- Couleur de fond du graphique

```
par(bg="aliceblue", col="red")
```

- Le titre et les labels

```
titre<-"Poids du cerveau / poids du corps"
```

```
labelX<-"Poids du corps"
```

```
labelY<-"Poids du cerveau"
```

- Le graphique (pch=type de point)

```
plot(Bodywt, Brainwt, xlim=c(5,250), main=titre, xlab=labelX, ylab=labelY, pch=16)
```

```
text(Bodywt, Brainwt, labels=Noms, adj=0)
```

- Les couleurs disponibles : colors()

● **exercice** : Comment rendre ce graphique le plus laid possible ? Modifiez les couleurs, les tailles de caractères, ...

## 47-Découper la fenêtre graphique

- La fenêtre graphique en 4 parties

`par(mfcol=c(2,2))`

1	3
2	4

La fenêtre graphique en 6 parties

`par(mfcol=c(3,2))`

1	4
2	5
3	6

- La fenêtre graphique en 6 parties

`par(mfcol=c(2,3))`

1	3	5
2	4	6

## 48-Découper la fenêtre graphique : un exemple

- Description de la variable rendement

```
par(mfcol=c(2,2))
```

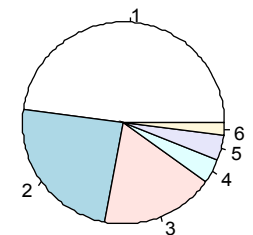
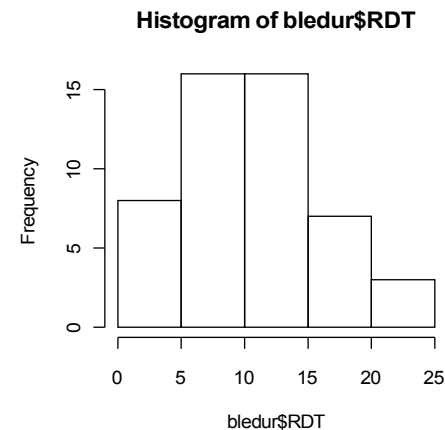
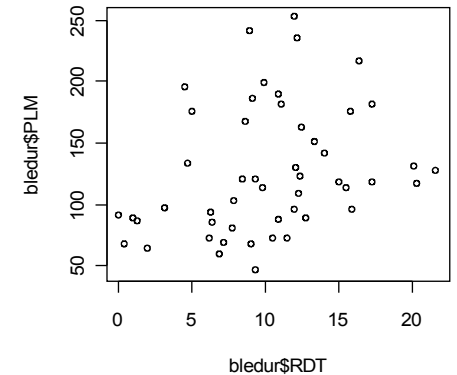
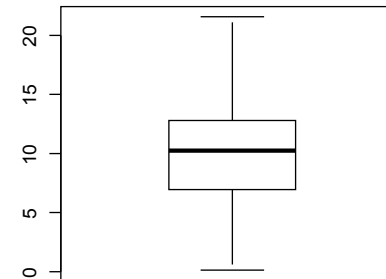
```
boxplot(bledur$RDT)
```


```
hist(bledur$RDT)
```

```
plot(bledur$RDT,bledur$PLM)
```

```
pie(summary(as.factor(bledur$VRT)))
```

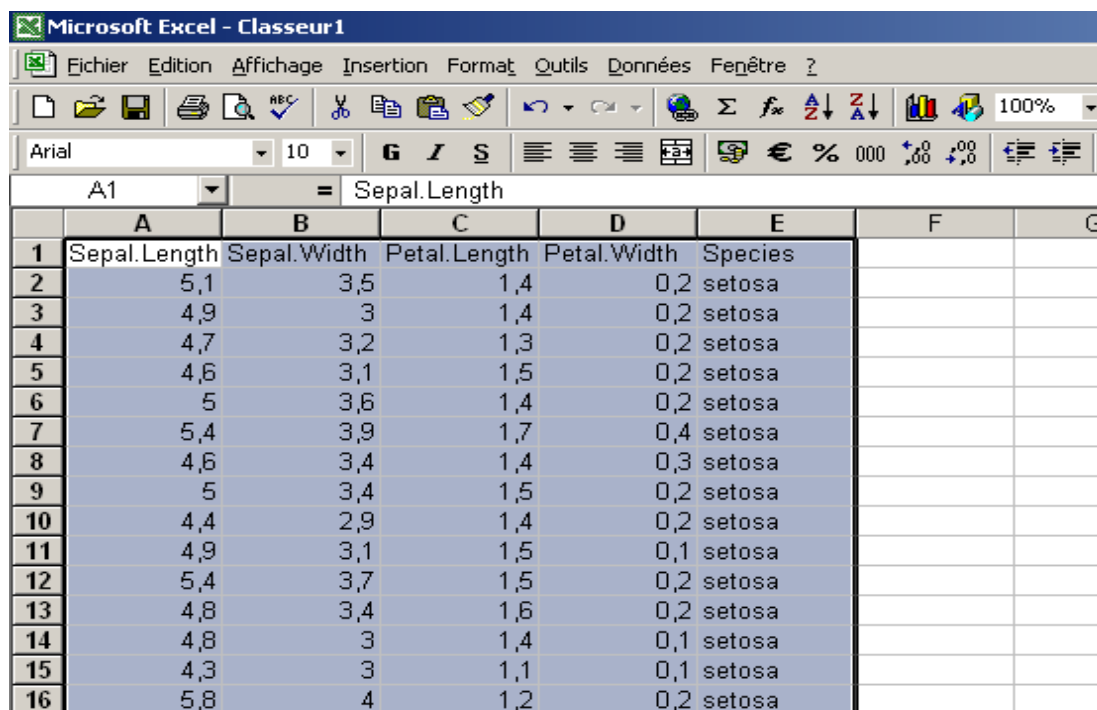
```
par(mfcol=c(1,1))
```



 **Remarque** : Les fonctions `split.screen()` ou `layout()` peuvent aussi être utilisées pour découper la fenêtre graphique

## 49-Copier-coller des données vers Open-Office (ou excel)

- Utilisez le fichier de données d'exemple "iris"
- Copier le data.frame "iris" dans le presse papier (clipboard)  
`write.table(iris,"clipboard", sep="\t", dec=".", row.names=F, col.names=T)`
- Ouvrir Excel ou OOO et coller le presse papier dans une feuille de calcul



	A	B	C	D	E	F	G
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species		
2	5,1	3,5	1,4	0,2	setosa		
3	4,9	3	1,4	0,2	setosa		
4	4,7	3,2	1,3	0,2	setosa		
5	4,6	3,1	1,5	0,2	setosa		
6	5	3,6	1,4	0,2	setosa		
7	5,4	3,9	1,7	0,4	setosa		
8	4,6	3,4	1,4	0,3	setosa		
9	5	3,4	1,5	0,2	setosa		
10	4,4	2,9	1,4	0,2	setosa		
11	4,9	3,1	1,5	0,1	setosa		
12	5,4	3,7	1,5	0,2	setosa		
13	4,8	3,4	1,6	0,2	setosa		
14	4,8	3	1,4	0,1	setosa		
15	4,3	3	1,1	0,1	setosa		
16	5,8	4	1,2	0,2	setosa		

- **Attention**, si il y a un identifiant de ligne (row.names=T)

- S'il y a un identifiant de lignes, on obtient un décalage des noms de colonnes dans le tableur :

```
write.table(iris,"clipboard", sep="\t", dec=",", row.names=T, col.names=T)
```

"Sepal.Length"	"Sepal.Width"	"Petal.Length"	"Petal.Width"	"Species"	
"1"	5,1	3,5	1,4	0,2	"setosa"
"2"	4,9	3	1,4	0,2	"setosa"
"3"	4,7	3,2	1,3	0,2	"setosa"
"4"	4,6	3,1	1,5	0,2	"setosa"
"5"	5	3,6	1,4	0,2	"setosa"

- Une solution : créer une nouvelle colonne contenant l'identifiant

```
numeros<-row.names(iris)
```

```
don<- data.frame(numeros, iris)
```

```
write.table(don,"clipboard", sep="\t", dec=",", row.names=F, col.names=T)
```

"numeros"	"Sepal.Length"	"Sepal.Width"	"Petal.Length"	"Petal.Width"	"Species"
"1"	5,1	3,5	1,4	0,2	"setosa"
"2"	4,9	3	1,4	0,2	"setosa"
"3"	4,7	3,2	1,3	0,2	"setosa"
"4"	4,6	3,1	1,5	0,2	"setosa"
"5"	5	3,6	1,4	0,2	"setosa"

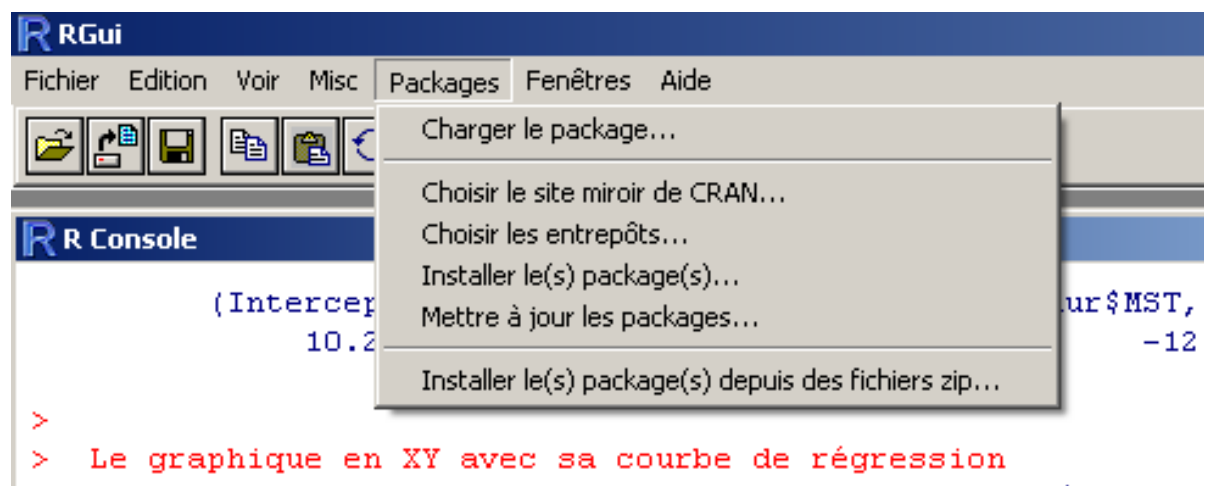
## 50-Utiliser des bibliothèques de fonctions

- Si la bibliothèque est déjà installée sur votre machine :

1) On peut utiliser le menu de R

Packages -> Charger le package, puis cliquez sur la bibliothèque désirée

2) On peut aussi taper l'instruction suivante dans la console  
library(nom de la bibliothèque)



## 51-les bibliothèques disponibles

- Pour consulter les bibliothèques installées sur votre machine  
cliquez sur **aide** -> **aide HTML**, puis, dans la page html : **Packages**  
( sous Gnu/Linux tapez : **help.start()** dans la console R )
- Pour consulter les bibliothèques disponibles sur internet  
rendez-vous sur le site : <http://lib.stat.cmu.edu/R/CRAN/>  
dans le menu de gauche, cliquez sur **Packages**

 Attention, certaines bibliothèques ne fonctionnent que sous GNU/Linux, d'autres ne sont utilisables que sous MS-Windows



## 52-Installer une bibliothèques de fonctions

- Vous avez une connexion internet,

c'est le cas le plus simple. On peut utiliser le menu de R  
package -> installer le(s) package(s) puis suivez les instructions

- Vous n'avez pas de connexion internet

vous installerez une bibliothèque à partir d'un fichier local (sur CD-Rom, extension .zip). **Attention aux dépendances !**

package -> installer le(s) package(s) depuis des fichiers zip



on peut effectuer les mêmes opérations en utilisant la fonction :  
install.packages()

## 53-Bibliothèque de fonction : lire un fichier MS-Excel

- On utilisera la bibliothèque de fonctions « xlsReadWrite »

```
library(xlsReadWrite)
```

```
read.xls(file.choose(), colNames= TRUE, sheet= 1, from= 1, rowNames=T)
```

- en choisissant le tableau Excel « Science » on obtient :

	A73	A74	A75	A76	A77	A78
<i>LifeSciences</i>	4489	4303	4402	4350	4266	4361
<i>PhysicalSciences</i>	4101	3800	3749	3572	3410	3234
<i>SocialSciences</i>	3354	3286	3344	3278	3137	3008
<i>BehavioralSciences</i>	2444	2587	2749	2878	2960	3049
<i>Engineering</i>	3338	3144	2959	2791	2641	2432
<i>Mathematics</i>	1222	1196	1149	1003	959	959

## 54-Bibliothèque de fonction : lire/écrire un fichier dBase

- On utilisera la fonction `read.dbf()` de la bibliothèque `foreign`

```
library(foreign)
```

```
read.dbf(file.choose()) # importez le fichier her.dbf situé  
                        dans le répertoire "data/Enquetes"
```

- Pour exporter au format dBase, on utilisera la fonction `write.dbf()`

```
write.dbf(iris, file="~/iris.dbf")
```

Sepal_Leng,N,19,15	Sepal_Widt,N,19,15	Petal_Leng,N,19,15	Petal_Widt,N,19,15	Species,C,10
5,10000000000000000	3,5000000000000000	1,4000000000000000	0,2000000000000000	setosa
4,9000000000000000	3,0000000000000000	1,4000000000000000	0,2000000000000000	setosa
4,7000000000000000	3,2000000000000000	1,3000000000000000	0,2000000000000000	setosa
4,6000000000000000	3,1000000000000000	1,5000000000000000	0,2000000000000000	setosa

## 55-Création d'une liste (list)

- À la différence d'un tableau de données, il n'y a pas de contrainte sur le type d'éléments contenus dans une liste.

### Par exemple :

```
data(iris)
iris.date<-date()
iris.cor<-cor(iris[,-5])
iris.legend<-"Un exemple de liste R"
iris.resume<-summary(iris)
```

```
iris.liste<-list(date=iris.date,cor=iris.cor,legend=iris.legend, resume=iris.resume)
```

```
names(iris.liste)
[1] "date"  "cor"   "legend" "resume"
```

## 56-Utilisation d'une liste (list)

- Extraction par index

```
> iris.liste[[1]]  
[1] "Fri Sep 21 08:36:56 2007"
```

- Extraction par nom

```
> iris.liste$date  
[1] "Fri Sep 21 08:36:56 2007"
```

- Extraction d'un sous-élément

```
> iris.liste$resume[1,]
```

```
Sepal.Length  Sepal.Width  Petal.Length  Petal.Width  Species  
"Min. :4.300 " "Min. :2.000 " "Min. :1.000 " "Min. :0.100 " "setosa :50 "
```

```
> iris.liste[[4]][1,]
```

```
Sepal.Length  Sepal.Width  Petal.Length  Petal.Width  Species  
"Min. :4.300 " "Min. :2.000 " "Min. :1.000 " "Min. :0.100 " "setosa :50 "
```

## 57-Régression linéaire

- Rappel : lecture des données `bledur.txt`

```
bledur<-read.table(file.choose(), header=T, na.string="M", dec=".", sep=" ")
```

- Le modèle

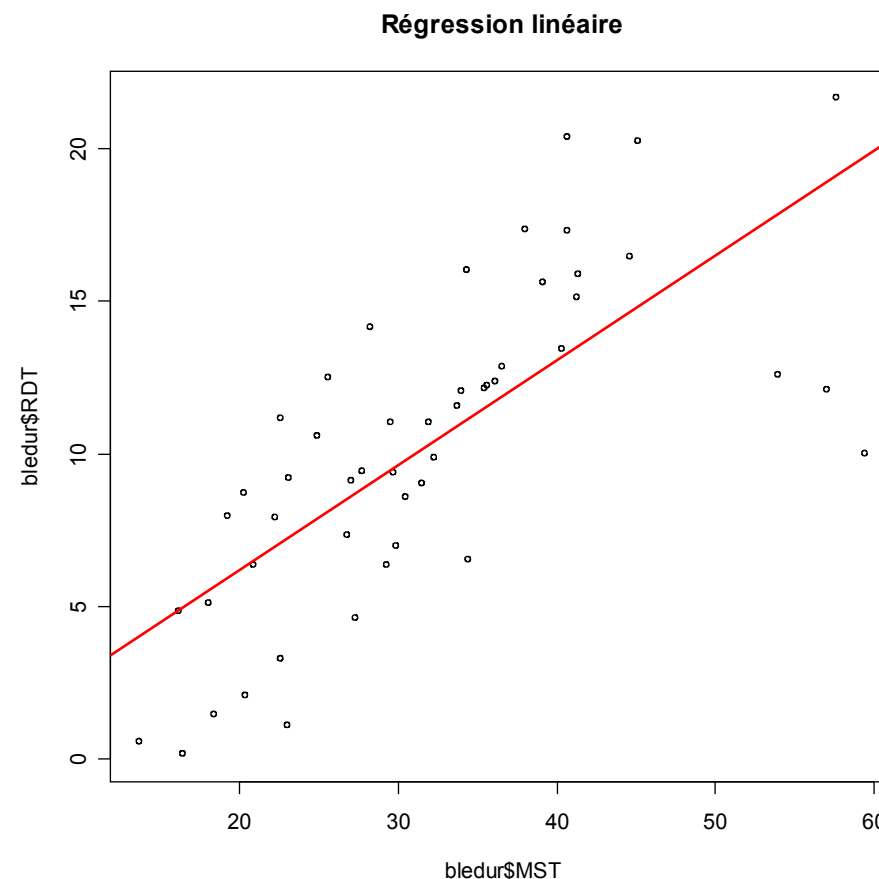
```
modele<-lm(RDT~MST,data=bledur)
```

- Les résultats

```
summary(modele)
```

- Une représentation graphique

```
plot(bledur$MST,bledur$RDT)  
abline(modele, lwd=2, col="red")  
title("Régression linéaire")
```



## 58-Régression linéaire : les résultats

- Que contient le modèle ?

`names(modele)`

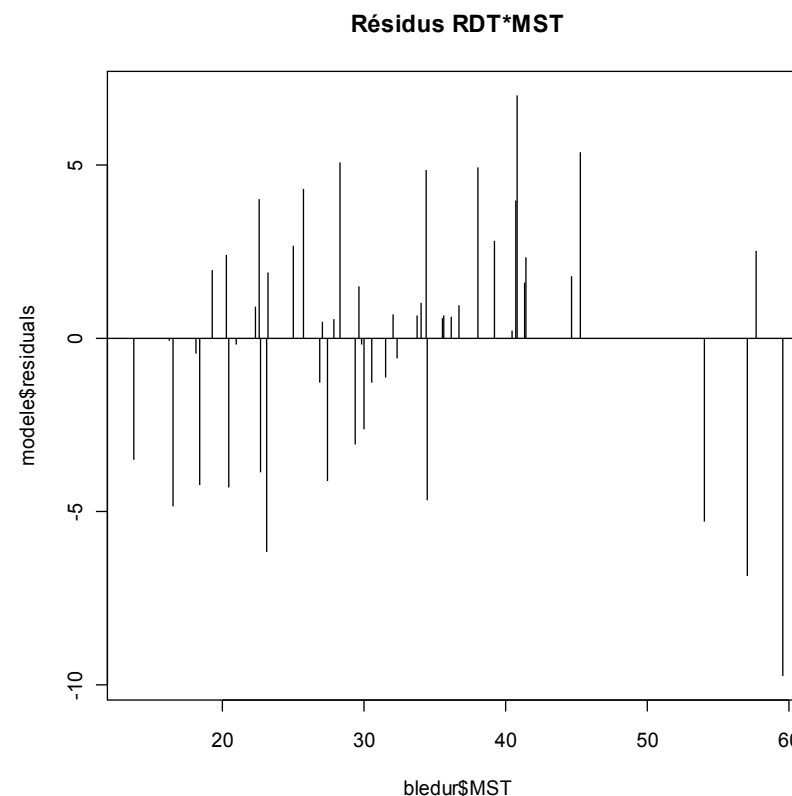
- Plus précisément

`coef(modele)`

`predict(modele)`

- Une représentation graphique des résidus

```
plot(bledur$MST,modele$residuals,main="Résidus RDT*MST", type="h")  
abline(0,0)
```



## 59-Et si la régression n'est pas linéaire ? le lissage

- Ajustement par lissage

# Fits a cubic smoothing spline

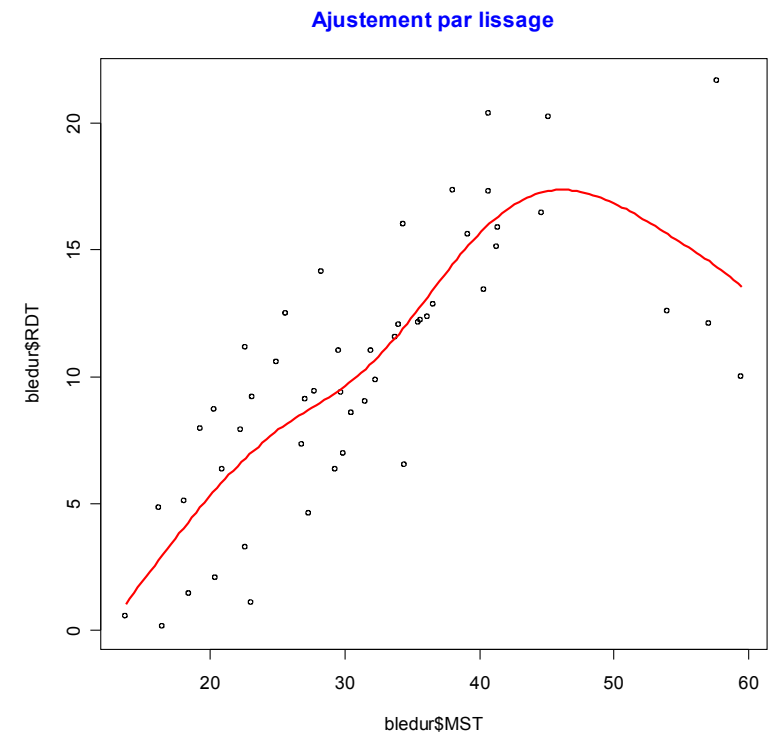
```
sp <- smooth.spline(bledur$MST,bledur$RDT, spar = 0.9)
```

# Projection des points sur un graphe XY

```
plot(bledur$MST,bledur$RDT,col.main="blue", main="Ajustement par lissage")
```

# tracé de l'ajustement :

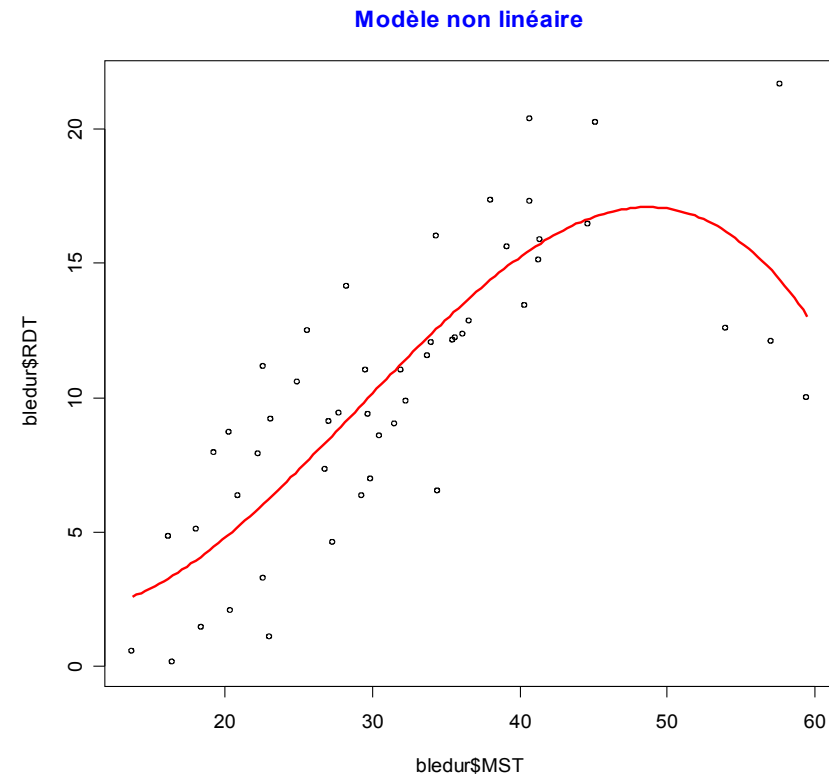
```
lines(spline(sp), col="red", lwd=2)
```





## 60-Une régression non linéaire

- Utilisation de la fonction `poly()`  
`mod<-lm(bledur$RDT ~ poly(bledur$MST, 3))`
- Le graphique en XY avec sa courbe de régression  
`plot(bledur$MST,bledur$RDT, col.main="blue", main="Modèle non linéaire")`  
`lines(spline(bledur$MST, mod$fitted.values), col="red", lwd=2)`
- Les paramètres du modèle  
`mod`



## 61-Analyse de la variance : les données

- Lecture des données : tableau « anova.txt »

```
expe<-read.table(file.choose(), header=T, sep=";", dec=".")
```

#attention les facteurs doivent être préparés

```
expe$bloc<-as.factor(expe$bloc) ; expe$trait<-as.factor(expe$trait)
```

- On travaille par défaut sur le data.frame "expe"

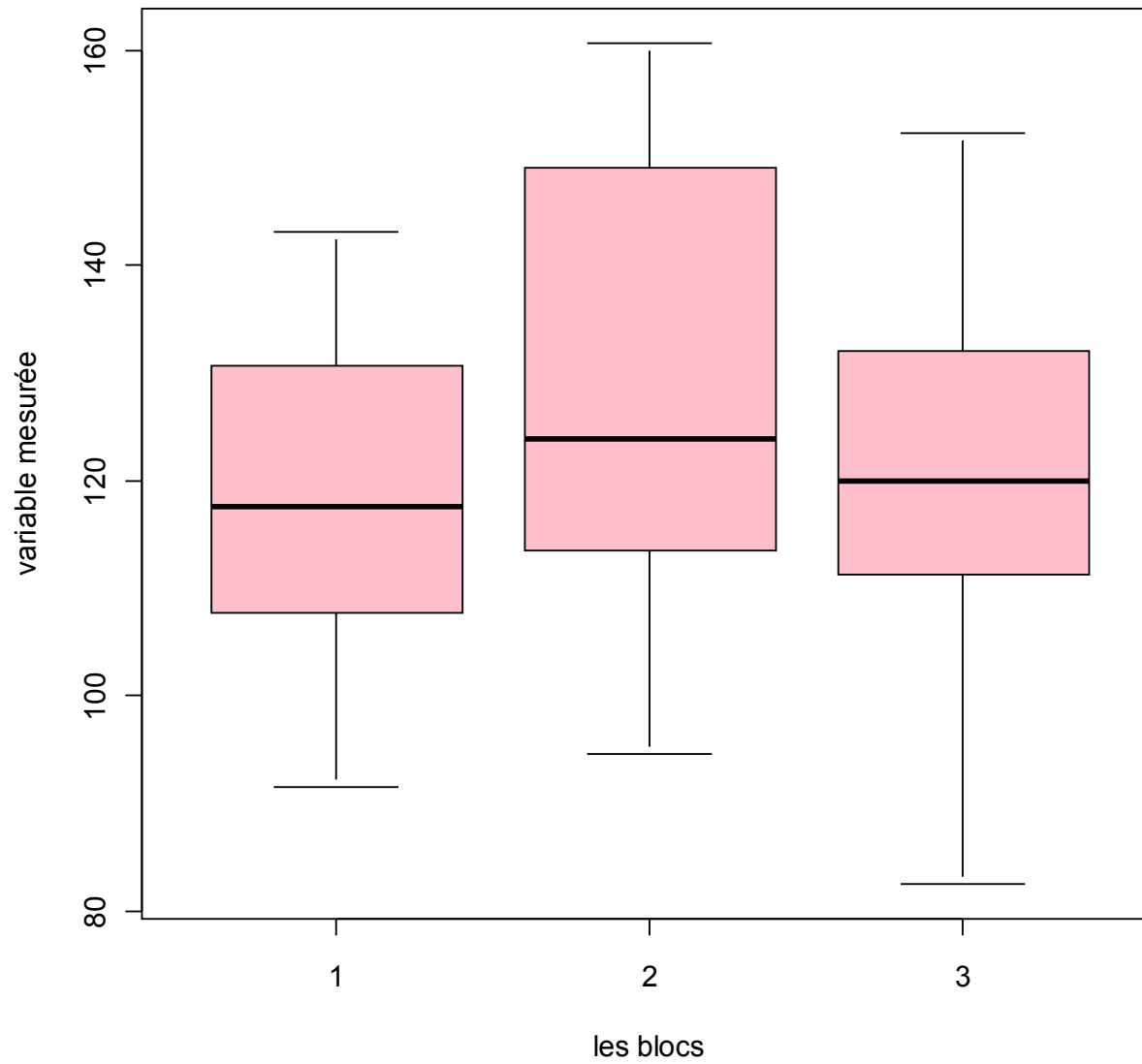
```
attach(expe)
```

- Observation graphique des données

```
boxplot(mesure ~ bloc, horizontal = F, main="Analyse de la variance",  
        ylab = 'variable mesurée', xlab = 'les blocs', col = "pink" )
```

```
detach(expe)
```

## Analyse de la variance



## 62-Analyse de variance à 2 facteurs : l'ANOVA

- Test d'égalité des variances

```
bartlett.test(expe$measure,expe$bloc)
```

```
bartlett.test(expe$measure,expe$trait)
```

- Analyse de variance

```
expe.aov <- aov(measure ~ bloc + trait, data=expe)
```

```
summary(expe.aov)
```

```
summary(expe.aov)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
bloc	2	492.6	246.3	0.9595	0.40181
trait	9	6460.2	717.8	2.7964	0.03026 *
Residuals	18	4620.4	256.7		

## 63-Analyse de variance à 2 facteurs : les résidus

- Vérification de la normalité des résidus

# test de shapiro

```
shapiro.test(expe.aov$residuals)
```

# histogramme des résidus

```
hist(expe.aov$residuals, freq=F, xlab="", main="Les résidus")
```

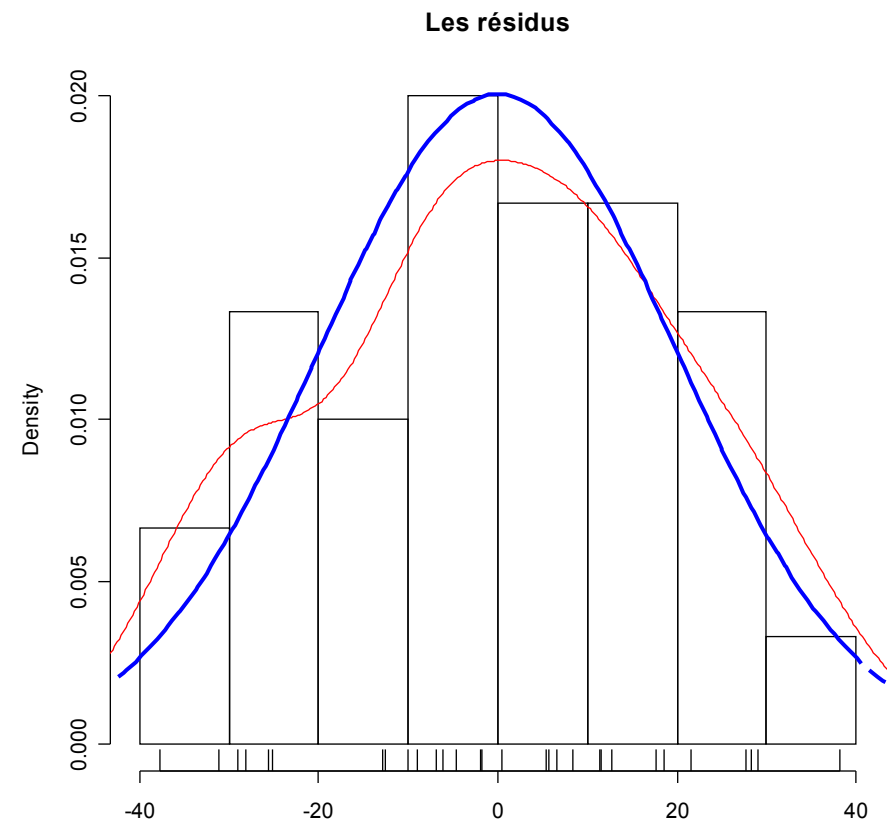
```
lines(density( expe.aov$residuals), col="red")
```

```
rug(jitter(expe.aov$residuals, 5))
```

## 64-Les résidus, représentation graphique enrichie

- ajouter la densité de la courbe normale

```
f <- function(t) # fonction densité d'une loi normale
{
  dnorm(t, mean=mean(expe.aov$residuals), sd= sd(expe.aov$residuals) )
}
curve(f, add=T, col="blue", lwd=3, lty=2)
```



## 65-Analyse de variances : test SNK

- Cette fonction est disponible dans le bibliothèque « agricolae »

```
library(agricolae) # chargement de la bibliothèque
modele<-aov(mesure ~ trait, data=expe)
SNK.test(modele,"trait", main="Effet du traitement")
```

Means with the same letter are not significantly different

Groups, Treatments and means

a	6	152.0267
ab	5	138.13
ab	9	126.3633
ab	1	125.3433
ab	3	122.68
ab	2	120.92
ab	7	115.85
ab	4	109.1467
b	0	106.7167
b	8	99.09667

## 66-Analyse de variance : les interactions

- R propose des graphiques d'interaction

```
par(mfrow=c(2,1))
```

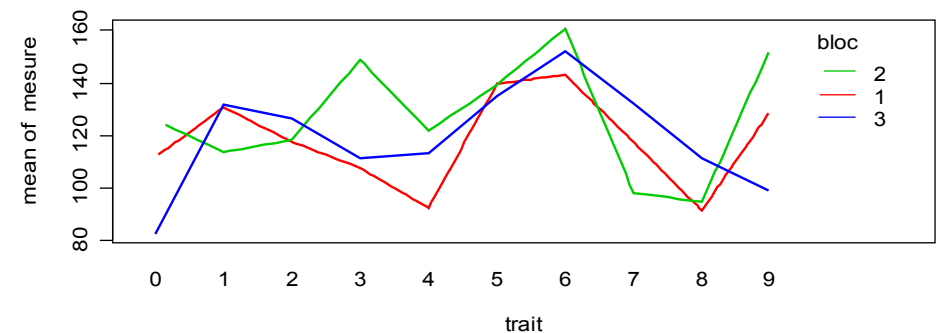
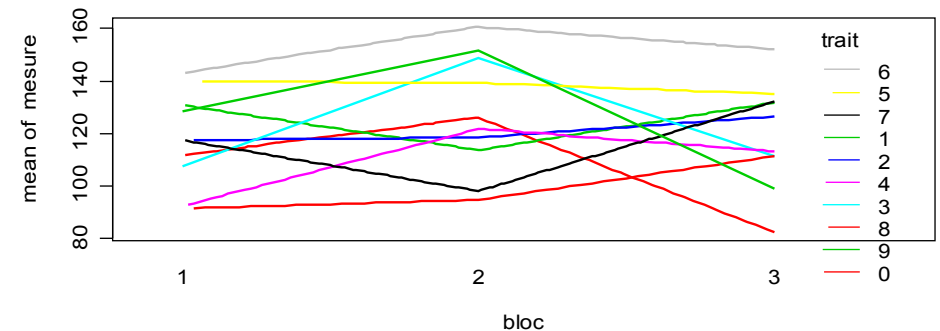
```
attach(expe)
```

```
interaction.plot(bloc, trait, mesure, col = 2:20, lwd=2)
```

```
interaction.plot(trait, bloc, mesure, col = 2:20, lwd=2)
```

```
detach(expe)
```

```
par(mfrow=c(1,1))
```





## 67-Puissance d'une expérience

Si vous ne connaissez pas les variances inter et intra :

# delta est la différence de résultat qu'on désire mettre en évidence

`x<-sd(expe.aov$residual)` # 12.62238 = écart-type du résidu

`power.t.test(power=NULL, n=3 , sd = x, delta = 50)`

Two-sample t test power calculation

```
      n = 3
  delta = 50
     sd = 12.62
sig.level = 0.05
  power = 0.94448
alternative = two.sided
```

NOTE: n is number in \*each\* group



**Remarque** : le Package 'samplesize' fournit des fonctions de calcul de taille d'échantillons.

## 68-Utilisation des dates

- Récupération des dates/heures du système

```
Sys.time() ; Sys.Date()
```

- Mise en forme des dates/heures (voir strptime() pour les formats)

```
format(Sys.time(), "%a %b %d %X %Y")
```

```
[1] "ven. nov. 04 17:13:56 2011"
```

- Création d'une séquence temporelle

```
serie<-seq(as.Date("2000/1/1"), as.Date("2000/1/31"), by="days")
```

```
format(serie, "%d-%A-%Y")
```

- Création d'un vecteur au format date

```
dates <- c("02/27/92", "02/27/92", "01/14/92", "02/28/92", "02/01/92")
```

```
as.Date(dates, "%m/%d/%y")
```

```
[1] "1992-02-27" "1992-02-27" "1992-01-14" "1992-02-28" "1992-02-01"
```

- Calcul de durées

```
x<-as.Date("2001/1/31") - as.Date("2000/1/1")
```

```
as.numeric(x)
```

```
[1] 396
```

## 69-Recodage de variables

- Utilisation de la fonction `recode()` de la bibliothèque "car"

# utilisation du data.frame `bledur` ; variable quantitative

```
recode(bledur$RDT,  
"  
  10:5 = 1 ;  
  5:10 = 2 ;  
  10:15 = 3 ;  
  15:hi = 4 ;  
  else = NA  
" )
```

# utilisation du data.frame `bledur` ; variable qualitative

```
recode(iris$Species,  
"  
  c('setosa', 'versicolor') = 'V1' ;  
  else = 'V2'  
" )
```

## 70-Rediriger les sorties vers des fichiers

- Rediriger du texte de R vers un fichier .txt

on utilisera la fonction `sink()`

- Rediriger les graphes R vers des fichiers images

on utilisera les fonctions `png()` ou `jpeg()` (*dev.off()* pour clore le fichier image)  
on peut aussi utiliser la fonction `savePlot()`

- Pour une mise en page directement dans un traitement de texte (OpenOffice)

on utilisera la bibliothèque `odfWeave` et la fonction `odfWeave()`  
pour en savoir plus : <http://alea.fr.eu.org/pages/intro-R>

**exemple :** le fichier 'nlin.png' est créé dans le répertoire de travail

```
png(file="nlin.png", bg="transparent")
mod<-lm(bledur$RDT ~ poly(bledur$MST, 3))
plot(bledur$MST,bledur$RDT, col.main="blue", main="Non linéaire")
lines(spline(bledur$MST, mod$fitted.values), col="red", lwd=2)
dev.off()
```

# 71-Analyse en composantes principales : les variables

- Le tableau des données : **Conso.txt**

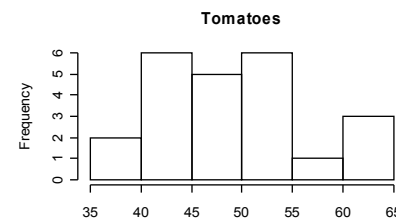
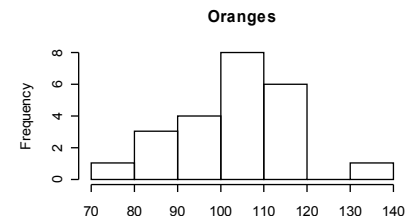
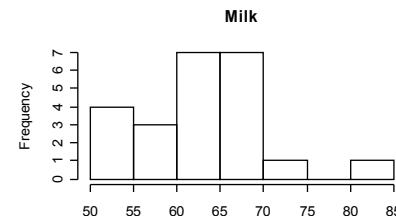
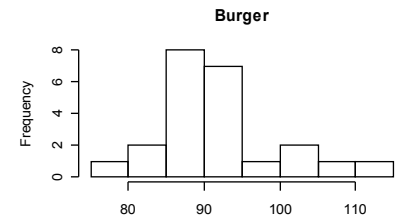
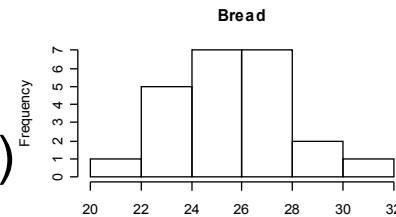
```
donconso<-read.table(file.choose(),header=T,dec="," ,sep=" ")
```

- Les histogrammes de toutes les variables

```
par(mfrow=c(3,2))
for(i in 2:6)
{
  hist(donconso[,i],main=names(donconso)[i], xlab="")
}
par(mfrow=c(1,1))
```

- Les relations entre les variables quantitatives

```
pairs(donconso[,-1])
```



## 72-L'A.C.P. : fonction princomp()

- Il existe plusieurs bibliothèques permettant des analyses multivariées
- On charge la bibliothèque contenant les fonctions  
`library(stats)`
- La variable "zone" devient identifiant des individus  
`row.names(donconso)<-donconso$zone`
- On crée une matrice ne contenant que les données numériques à analyser  
`donpca<-as.matrix(donconso[,-1])`
- On lance l'analyse en composantes principale  
`z<- princomp(donpca)`  
`summary(z)`

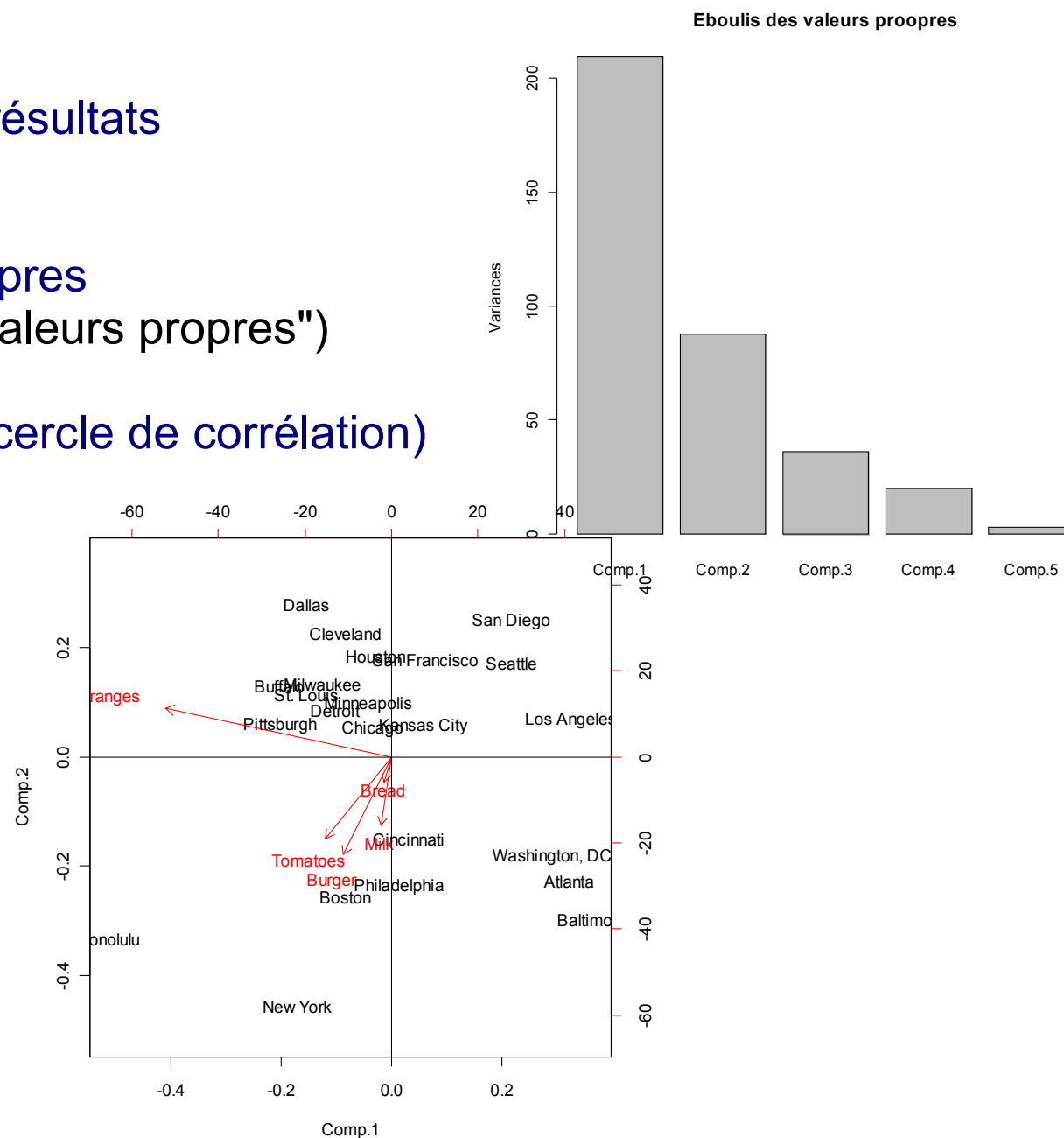
Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Standard deviation	14.4733202	9.3667073	6.0020981	4.4615844	1.702211420
Proportion of Variance	0.5883514	0.2464191	0.1011828	0.0559086	0.008138182
Cumulative Proportion	0.5883514	0.8347704	0.9359532	0.9918618	1.000000000

## 73-L'A.C.P. : les résultats

- Impression d'un résumé des résultats  
`summary(z)`
- Histogramme des valeurs propres  
`plot(z, main="Eboulis des valeurs propres")`
- Le plan principal (individus + cercle de corrélation)  
`biplot(z)`  
`abline(h=0,v=0)`
- Les composantes principales  
`z$scores`

● **Exercice** : faire une ACP sur le fichier de données *Science.txt*



## 74-Créer une fonction (1)

Cette fonction doit dessiner un graphique XY et tracer une droite de régression.

- Le programme qu'on veut "simplifier"

```
# lecture des données : fichier bledur.txt
```

```
bledur <- read.table(file.choose(), header=T, na.string="M", dec=".", sep=" ")
```

```
# régression linéaire
```

```
z<-aov(bledur$SAB ~ bledur$LIM)
```

```
# graphe XY
```

```
plot(bledur$LIM , bledur$SAB , ylab="SAB",xlab="LIM")
```

```
# ajout de la droite de régression
```

```
abline(z)
```



## 75-Créer une fonction (2)

- La fonction :

```
grapheXY<-function(X,Y,TAB)
{
  attach(TAB)
  z<-aov(Y ~ X)
  titre<-paste("Relation entre ",substitute(X), " et ",substitute(Y))
  plot(X, Y, ylab= substitute (Y), xlab= substitute (X), main=titre)
  abline(z)
  detach(TAB)
}
```



**Remarque :** *il peut-être utile d'insérer des commentaires...*

## 76-Utilisation de la fonction

- Sauvegarder la fonction

sauvegarder la fonction dans un fichier : grapheXY.R (attention à l'extension)

- Charger la fonction

pour l'utiliser, lire le fichier contenant vos fonctions avec le menu

« **File/Source R code** »

ou insérer cette instruction dans votre script

source("*emplacement sur le disque*/grapheXY.R")

- Utiliser la fonction

grapheXY(LIM, ARG, bledur)