

Formation Permanente

CENTRE INRA de MONTPELLIER

FORMATION AU LOGICIEL R

(durée : 2 jours)

version du 1 Juin 2006



André Bouchier

Copyright © André Bouchier.

© 2006, André Bouchier (20 Janvier 2006)

Permission est accordée de copier et distribuer ce document, en partie ou en totalité, dans n'importe quelle langue, sur n'importe quel support, à condition que la notice © ci-dessus soit incluse dans toutes les copies. Permission est accordée de traduire ce document, en partie ou en totalité, dans n'importe quelle langue, à condition que la notice © ci-dessus soit incluse.

1-Qu'est ce que R

- R : logiciel multiplateforme
- R : système statistique et graphique
- R : un logiciel et un langage
- R : logiciel libre
- R : langage orienté objet

2-Prise en main du logiciel

- Installation

Sur le CD, dans le répertoire Rwindows, lancez le programme SetupR.exe.
Suivez les instructions affichées à l'écran.

- Prise en main

Taper **Ctrl L** pour nettoyer la fenêtre 'Rconsole'

Faire des opérations : $10+3*5+\log(10)+2$

$(10+3)*(5+\log(10))+2$

Utiliser la flèche (clavier) \uparrow pour faire défiler les commandes déjà tapées

La souris permet de sélectionner, de copier-coller les lignes de Rconsole

3-Exemple d'utilisation :

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

- Charger le tableau de données "iris"

```
data(iris)
```

- Que contient ce tableau ?

```
summary(iris)
```

- Une présentation graphique

```
pairs(iris)
```

4-Stocker les résultats

- 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

```
a<-0; ifelse (a>=1, b<-"oui", b<-"non"); b
```

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

6-L'environnement de travail

- Connaître le répertoire de travail de R

```
getwd()
```

- Changer le répertoire de travail de R

```
setwd("E:/HD20GO/R/r/Rwindows/data/FormatR")
```

- Le chemin peut-être fastidieux à écrire

```
setwd(dirname(file.choose()))
```

```
nom<-file.choose() #permet de choisir un fichier de façon interactive
```

```
dirname(nom) #extrait le chemin d'un fichier
```


7-Lecture de données R (format binaire)

- Lecture du fichier de données R « voit2005.Rdata »

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

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

```
ls()
```

8-Objets en mémoire

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

```
ls()
```

- Effacer les objets en mémoire

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

- Effacer tous les objets

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

9-Types de données

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

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

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

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

- **Array** : matrice à k dimensions

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

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

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

10-Connaître le types de données

- Un vecteur peut être composé de données de différents types
numérique, caractère, entier, réel
- Utiliser la fonction `typeof()`

```
a<-"inra"  
typeof(a)  
[1] "character"
```

```
a<-4  
typeof(a)  
[1] "double"
```

```
a<-as.integer(a)  
typeof(a)  
[1] "integer"
```

11-Utiliser un data.frame (1)

- Connaître le nom des variables du data.frame

```
names(voit2005)
```

- Les dimensions du data.frame

```
dim(voit2005)
```

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

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

- Le nombre de colonnes

```
length(voit2005)
```

12-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

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

13-Extraire des données d'un data.frame

- 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^{ème}), toutes les variables (sauf la 1^{ère})
`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$Puissance>10,]`
- Combiner des conditions
`voit2005[voit2005$Puissance>10 & voit2005$Cylindree>2000 , 1:3]`

14-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","Ford MONDEO TD","BMW 518i"), 1:3]
```


15-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

16-Générer une séquence

- Séquence simple

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

```
typeof(z)
```

- Attention, comparez

```
X<-10
```

```
1:X-1
```

```
et
```

```
1:(X-1)
```

- On peut utiliser la fonction seq()

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

17-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) ; length(x)
```

- ***exercice*** : essayez avec 10 000 individus, puis avec 100 000

18-Demander de l'aide

- En savoir plus sur une fonction

```
help.search("title")
```

```
?mean
```

- Comment importe-t-on des fichiers textes ?

```
?read.table
```

19-Utilisation d'un éditeur de texte pour écrire des scripts

- Ouvrez un éditeur de texte, "notepad", "Tinn-R" ou autre
- 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)
```

```
#Combien de valeurs ?
```

```
length(x)
```

- Transférez-les dans la fenêtre R par simple copier/coller
- N'oubliez pas de commenter vos "programmes"

20-Exercices

- Quelles sont les moyennes des variables Longueur, Largeur et Surface du data.frame **voit2005**
- Combien de véhicules ont une cylindrée plus petite que 1000 cm^3 ?
- Représentez les graphiques x-y de toutes les variables quantitatives du data.frame **voit2005**

21-Importer des données au format texte

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

```
don <- read.table("E:/HD20GO/R/r/Rwindows/data/FormatTexte/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 :

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

- Exercice : importez le fichier texte **sen.txt**. Combien de lignes et de variables possède-t-il ?

22-Enregistrer un tableau de données au format R

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

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

- Par défaut, ce fichier est sauvegardé dans le répertoire de travail de R
- On peut choisir un autre répertoire :

```
save(sen, file = "c:/temp/sen.Rdata")
```

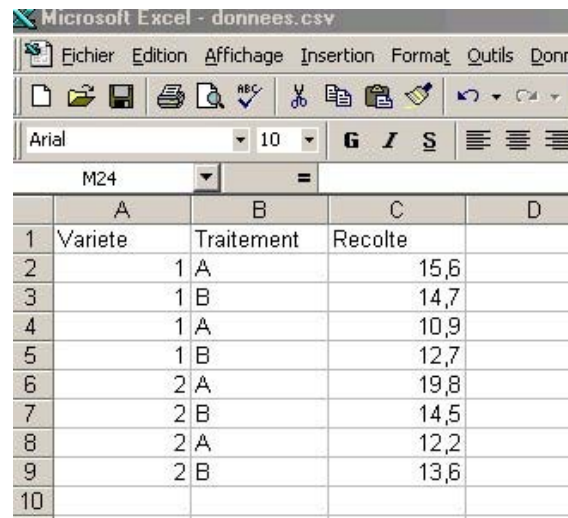
- Ou alors plus simplement :

```
setwd(dirname(file.choose())) # le répertoire choisi ne peut pas être vide  
save(sen, file = "sen.Rdata")
```


23-Passage d'Excel à R (utilisation du format .csv)

- Saisissez ces données sous Excel. La première ligne contient les noms des variables

Enregistrer ce tableau au format csv



	A	B	C	D
1	Variete	Traitement	Recolte	
2		1 A	15,6	
3		1 B	14,7	
4		1 A	10,9	
5		1 B	12,7	
6		2 A	19,8	
7		2 B	14,5	
8		2 A	12,2	
9		2 B	13,6	
10				

- Pour le lire, utilisez la fonction "`read.table`"
- Enregistrez-le au format R - fonction `save()`

24-Éditer des données

Le mieux est d'utiliser des outils de gestion de données.

Cependant, R permet :

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

```
xnew <- edit(don)
```

- de créer un nouveau tableau de données

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

- d'éditer un objet

```
don<-edit(don)
```

25-Modifier les données d'un tableau

- Créer une nouvelle variable

```
v1<-don$LIM/don$ARG*100 ; v1<-round(v1, digits = 2)
```

- L'ajouter dans le tableau de données

```
x<-data.frame(don,v1)
```

- Modifier une variable

```
x$RDT<-x$RDT/10
```

- Renommer une variable

```
names(x)[2]<-"Rendement"
```

- Trier un tableau de données

```
don[sort.list(don[,2]),]  
don[sort.list(don$Numero),]
```

26-Combiner 2 data.frame

Merge two data frames by common columns or row names

- Un premier data.frame décrit les exploitations agricoles

```
exploitation<-read.table("d:/data/exploitation.txt",header=T,sep=" ",dec=".")
```

- Le 2ème contient les assolements pour chaque exploitation

```
cultures<-read.table("d:/data/culture.txt",header=T,sep=" ",dec=".")
```

- La fonction "merge()" combine les 2 tableaux de données

```
merge(exploitation, cultures, by.x = "exploitation", by.y = "exploitation", sort=T)
```

27-Supprimer les données manquantes d'un tableau

- Supprimer les données manquantes de tout le tableau

```
na.omit(don)
```

```
don2<-na.omit(don)
```

- ***exercice** : lire le fichier **produit.txt**. Attention, il contient des données manquantes. Quelle est la moyenne de la variable PRODUIT ?*

28-Décrire des variables quantitatives

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

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

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

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

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

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

29-Les histogrammes (1)

- Les données quantitatives
`hist(don$RDT)`
- On peut choisir le nombre de barres
`hist(don$RDT,nclass=5)`
- Pour avoir les bornes et les effectifs des classes
`hist(don$RDT,nclass=5,plot=F)`
- Choisir ce dont on a besoin :
`a<- hist(don$RDT,nclass=5,plot=F)`
`names(a)`
`bornes<-a$breaks`

30-Les histogrammes (2)

- Attention, quelle différence entre ?

```
hist(don$VRT)
```

```
barplot(summary(don$VRT))
```

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


31-Boites à pattes

- Boite simple

```
boxplot(don$RDT)
```

- Une boite par niveau de facteur

```
boxplot(split(don$RDT,don$ZON))
```

- Pour rendre le graphique plus lisible, ajouter un titre


```
title("Rendement", sub="Données INRA", ylab= "Effectifs",  
      xlab= "zones géographiques")
```

- Représenter les individus

```
rug(don$RDT, side=2)
```

- Exporter le graphique sous différents formats. Menus File/Save as/...

32-Les camemberts

- Les données doivent être préparées
`pie(don$VRT) # mauvais`
`pie(summary(as.factor(don$VRT))) # correct`
- On ajoute un nom "explicite" pour chaque secteur
`nom<-c("Variété1","Variété2","Variété3","Variété4","Variété5","Variété6")`
`pie(summary(as.factor(don$VRT)),label=nom)`
- On ajoute un titre principal
`titre<-"Les variétés cultivées"`
`pie(summary(as.factor(don$VRT)),label=nom,main=titre)`
-  *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.*

33-Un graphique en x-y

- Simple graphique

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

- Graphique "commenté"

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

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

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

- Graphique illustré

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

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

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

34-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)
pairs(don[,maliste])
title("Enquête Algérie ", sub="Données INRA")
```

- Comment afficher le titre sans cacher l'échelle ?

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

35-Le test du Khi2

- **Créer le tableau croisé**

```
table(don$ZON,don$VRTC)
```

- **Afficher les noms de variables du tableau**

```
table(don$ZON,don$VRTC,deparse.level = 2)
```

- **Préciser le nom des variables**

```
table(don$ZON,don$VRTC, dnn=c("Zones géographiques","Les variétés"))
```

- **Tester l'indépendance**

```
tb<-table(don$ZON,don$VRTC)  
summary(tb)
```

- **On peut simuler un tirage aléatoire**

```
chisq.test(tb, simulate.p.value = TRUE, B = 10000)$p.value
```

36-Le test du Khi 2 - Découpage en classes de variable

- Utiliser une variable quantitative découpée en classes

```
tb<- table(cut(don$RDT, quantile(don$RDT)), don$VRTC,  
           dnn=c("Rendement","Variétés"))
```

- Le test du khi2

```
summary(tb)
```

- Ki2 par simulation

```
chisq.test(tb, simulate.p.value = TRUE, B = 10000)$p.value
```

37-Les corrélations

- Choisir les données quantitatives

```
lv<-c(2,3,5,6,7,9,10,11)  
donquant<-don[,lv]
```

- Un tableau des corrélations

```
cor(donquant)
```

- Précisions sur les coefficients de corrélation

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

38-Représentations graphiques, illustration des points

- Les données

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

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

- Les noms à insérer sur le graphique

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

- Le graphique

```
plot(Bodywt, Brainwt, xlim=c(5, 250))
```

- Les chaines de caractères à insérer

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


39-Représentations 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(Bodywt, Brainwt, xlim=c(5, 250),pch=16)  
text(Bodywt,Brainwt,labels=Noms,adj=0)
```

- Initialiser les paramètres aux valeurs par défaut

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

40-Représentations graphiques, illustration des points (2)

- **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, ...*

41-Découper la fenêtre graphique

- La fenêtre graphique en 4 parties

```
layout(matrix(c(1:4),2,2))
```

```
layout.show(4)
```

1	3
2	4

- La fenêtre graphique en 6 parties

```
layout(matrix(c(1:6),3,2))
```

```
layout.show(6)
```

1	4
2	5
3	6

- La fenêtre graphique en 6 parties

```
layout(matrix(c(1:6),2,3))
```

```
layout.show(6)
```

1	3	5
2	4	6

42-Découper la fenêtre graphique : un exemple d'utilisation

- Description de la variable rendement

```
layout(matrix(c(1:4),2,2))
```

```
boxplot(don$RDT)
```

```
hist(don$RDT)
```

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

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

```
layout(1)
```

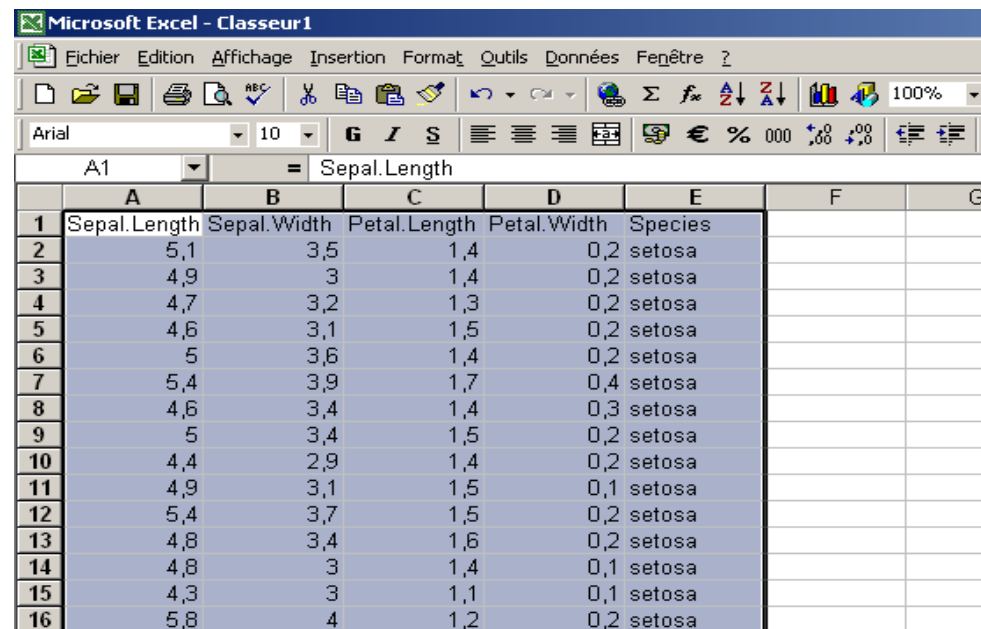
- Note : *La fonction `split.screen()` peut aussi être utilisée pour découper la fenêtre graphique*

43-Copier-coller des données vers MS-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)
```

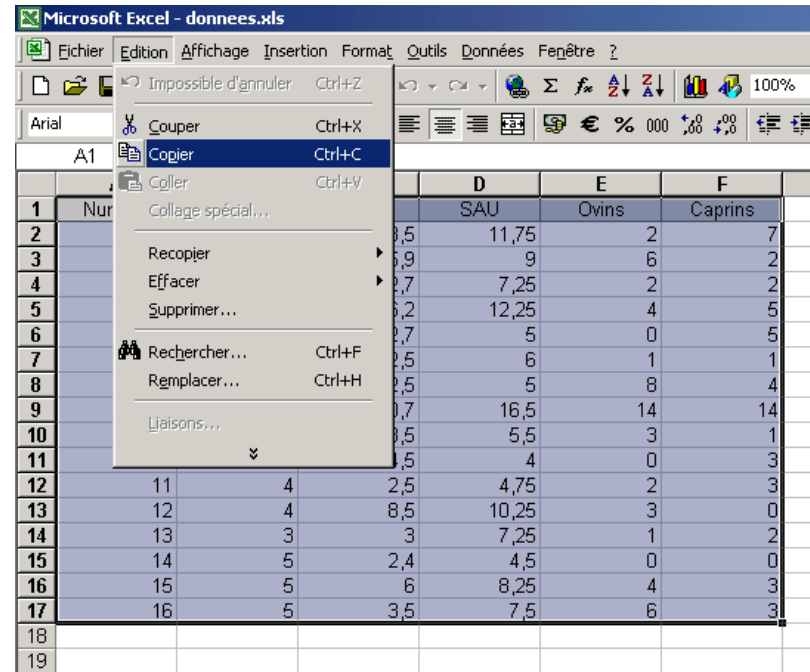
- Ouvrir Excel et copier le presse papier dans une feuille de calcul



	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		

44-Copier-coller des données dans R à partir de MS-Excel :

- Ouvrir le fichier donnees.xls, sélectionner et copier les données



- Copier le presse papier (clipboard) dans le data.frame y

```
y<-read.table("clipboard",sep="\t",dec="," ,header=T)
```

45-Régression linéaire

- **Rappel** : lecture des données **bledur.txt**

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

- **Le modèle**

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

- **Les résultats**

```
summary(modele)
```

- **Une représentation graphique**

```
layout(1)  
plot(don$MST,don$RDT)  
abline(modele)
```

46-Connaître les infos disponibles après une régression linéaire

- Que contient le modèle ?

```
names(modele)
```

- Plus précisément

```
coef(modele)
```

```
predict(modele)
```

- Une représentation graphique des résidus

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

```
abline(0,0)
```


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

- **Rappel** : lecture des données `bledur.txt`

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

- **Ajustement par lissage**

```
# Fits a cubic smoothing spline
```

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

```
# Projection des points sur un graphe XY
```

```
plot(don$MST,don$RDT, col.main=2)
```

```
# Une séquence de points sur l'axe des X
```

```
xx <- seq(min(don$MST),max(don$MST), len=20)
```

```
#Tracé de la courbe d'ajustement
```

```
lines(predict(sp, xx), col = 3)
```

48-Si la régression n'est pas linéaire ? régression non linéaire

- Utilisation de la fonction `poly()`

```
mod<-lm(don$RDT ~ poly(don$MST, 3))
```

- Les paramètres du modèle

```
mod
```

- Fits a cubic smoothing spline

```
sp <- smooth.spline(don$MST,mod$fitted.values, spar = 0.9)
```

- Séquence sur l'axe horizontal

```
xx <- seq(min(don$MST),max(don$MST), len=20)
```

- Le graphique en XY avec sa courbe de régression

```
plot(don$MST,don$RDT, col.main=2)
```

```
lines(predict(sp, xx), col = 3)
```

49-Analyse de la variance : les données

- Lecture des données

```
expe<-read.table("d:/r/Rwindows/data/anova.txt",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 "essai"

```
attach(expe)
```

- Observation graphique des données :

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

```
detach(expe)
```

50-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(mesure ~ bloc + trait, data=expe)
```

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

51 - 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, probability=TRUE, xlab="", main="Les résidus")
```

52-Les résidus, une représentation graphique enrichie

```
residus<- expe.aov$residuals
```

```
# l'histogramme des résidus
```

```
hist(residus, probability=TRUE, xlab="Variable mesurée", main="Les résidus")
```

```
# la courbe de densité des résidus
```

```
points(density(residus, bw=5), type='l', col="blue")
```

```
# le "tapis" des valeurs résiduelles
```

```
rug(jitter(residus, 5))
```

```
# densité de la courbe normale
```

```
f <- function(t) { dnorm(t, mean=mean(residus), sd=sd(residus) ) }
```

```
curve(f, add=T, col="red", lwd=3, lty=2)
```

53-Comparaison multiple de moyennes : test t

- Les moyennes par niveaux de facteur

```
model.tables(aov(mesure~trait,data=expe), type = "means")
```

- Le test de comparaison multiple de moyennes

```
pairwise.t.test(expe$mesure, expe$trait)
```

54-Analyse de variance à 2 facteurs : et les interactions ?

- R propose des graphiques d'interaction

```
attach(expe)
```

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

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

```
detach(expe)
```


55-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")
```

- **Création d'une séquence temporelle**

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

- **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")
```

- **Calcul de durées**

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

56-Rediriger les sorties textes vers un fichier

- Le fichier texte est créé dans le répertoire de travail

```
setwd("c:/temp")
```

```
sink("toto.txt", append=TRUE)
```

- Pour commenter le fichier texte

```
getwd()
```

```
cat("La liste des objets imprimés le : ",date(),"\n")
```

```
ls()
```

- Pour terminer la redirection

```
sink()
```

- **Exercice** : refaire l'analyse de variance (essai.txt). Récupérer les résultats dans un fichier texte : "resultat.txt"

57-Utiliser des bibliothèques de fonctions

- Si la bibliothèque est déjà installée sur votre machine :

1) On peut utiliser le menu de R

package -> charger le package -> cliquez sur la bibliothèque désirée

2) On peut taper l'instruction suivante dans la console

`library(nom de la bibliothèque)`

- Pour consulter le contenu des bibliothèques installées sur votre machine :

cliquez sur le menu R puis,

aide -> aide HTML -> puis, dans la page html : package

58-Installer de nouvelles bibliothèques de fonctions (1)

- Vous avez une connection internet et des droits d'écriture dans les répertoires systèmes :

C'est le cas le plus simple. Dans le console R, tapez :

```
install.packages()
```

- Vous n'avez pas de connection internet mais vous avez des droits d'écriture dans les répertoires systèmes :

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

dans le console R, cliquez sur :

```
install.packages(file.choose(), repos = NULL)
```

59-Installer de nouvelles bibliothèques de fonctions (2)

- Vous avez une connection internet mais pas de droits d'écriture dans les répertoires systèmes :

vous installerez la bibliothèque dans un répertoire personnel :

```
install.packages(lib="c:/temp/package")
```

- Vous n'avez ni connection internet ni droits d'écriture dans les répertoires systèmes :

vous installerez la bibliothèque dans un répertoire personnel à partir d'un fichier local (sur CD-Rom, extension .zip) :

```
install.packages(file.choose(), repos = NULL, lib="c:/temp/package")
```

60-Utiliser une nouvelle bibliothèque de fonctions "locale"

- Vous avez installée bibliothèque R2HTML dans un répertoire perso:

```
library(R2HTML, lib.loc="c:/temp/package")
```

- Pour avoir de l'aide sur une fonction :

```
shell.exec("c:/temp/package/R2HTML/html/00Index.html")
```

```
help(HTML, lib.loc = "c:/temp/package")
```

- Utiliser cette fonction :

```
library(R2HTML, lib.loc="c:/temp/package")
```

```
HTML(iris, file="c:/temp/iris.html", append = FALSE)
```

```
shell.exec("c:/temp/iris.html")
```

61 - Analyse en composantes principales : description des variables

- Le tableau des données

```
donconso<-read.table("D:/R/conso.txt",header=T,dec="," ,sep=" ")
```

- Les histogrammes de toutes les variables

```
layout(matrix(c(1,2,3,4,5,6),3,2))  
for(i in 2:6) {hist(donconso[,i],main=names(donconso)[i])}  
layout(1)
```

- Les relations entre les variables quantitatives

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

62-L'analyse en composantes principales : suite

- On charge la bibliothèque contenant les fonctions
`library(mva)`
- 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)`

63-L'analyse en composantes principales : les résultats

- Impression d'un résumé des résultats

`summary(z)`

- Histogramme des valeurs propres

`plot(z)`

- 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 olympic.txt*

64-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
```

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

```
#régression linéaire
```

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

```
#graphe XY
```

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

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

```
abline(z)
```

65-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...*

66-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, lancer le fichier avec le menu « **File/Source R code** »
ou insérer cette instruction dans votre script
`source("d:/R/Rwindows/data/grapheXY.R")`

- Utiliser la fonction

`grapheXY(LIM,ARG,don)`

Table des matières

1-Qu'est ce que R.....	34-Graphique de toutes les variables quantitatives.....
2-Prise en main du logiciel.....	35-Le test du Khi2.....
3-Exemple d'utilisation :.....	36-Le test du Khi 2 – Découpage en classes de variable.....
4-Stocker les résultats.....	37-Les corrélations.....
5-Quelques opérations.....	38-Représentations graphiques, illustration des points.....
6-L'environnement de travail.....	39-Représentations graphiques, gestion des paramètres.....
7-Lecture de données R (format binaire).....	40-Représentations graphiques, illustration des points (2).....
8-Objets en mémoire.....	41-Découper la fenêtre graphique.....
9-Types de données.....	42-Découper la fenêtre graphique : un exemple d'utilisation.....
10-Connaître le types de données.....	43-Copier-coller des données vers MS-Exel :.....
11-Utiliser un data.frame (1).....	44-Copier-coller des données dans R à partir de MS-Exel :.....
12-Utiliser un data.frame (2).....	45-Régression linéaire.....
13-Extraire des données d'un data.frame.....	46-Connaître les infos disponibles après une régression linéaire.....
14-Identifier les lignes du tableau de données.....	47-Et si la régression n'est pas linéaire ? le lissage.....
15-Exercice.....	48-Si la régression n'est pas linéaire ? régression non linéaire.....
16-Générer une séquence.....	49-Analyse de la variance : les données.....
17-Les séquences aléatoires.....	50-Analyse de variance à 2 facteurs : l'ANOVA.....
18-Demander de l'aide.....	51-Analyse de variance à 2 facteurs : les résidus.....
19-Utilisation d'un éditeur de texte pour écrire des scripts.....	52-Les résidus, une représentation graphique enrichie.....
20-Exercices.....	53-Comparaison multiple de moyennes : test t.....
21-Importer des données au format texte.....	54-Analyse de variance à 2 facteurs : et les interactions ?.....
22-Enregistrer un tableau de données au format R.....	55-Utilisation des dates.....
23-Passage d'Excel à R (utilisation du format .csv).....	56-Rediriger les sorties textes vers un fichier.....
24-Éditer des données	57-Utiliser des bibliothèques de fonctions.....
25-Modifier les données d'un tableau.....	58-Installer de nouvelles bibliothèques de fonctions (1).....
26-Combiner 2 data.frame.....	59-Installer de nouvelles bibliothèques de fonctions (2).....
27-Supprimer les données manquantes d'un tableau.....	60-Utiliser une nouvelle bibliothèque de fonctions "locale".....
28-Décrire des variables quantitatives.....	61-Analyse en composantes principales : description des variables.....
29-Les histogrammes (1).....	62-L'analyse en composantes principales : suite.....
30-Les histogrammes (2).....	63-L'analyse en composantes principales : les résultats.....
31-Boîtes à pattes.....	64-Créer une fonction (1).....
32-Les camemberts.....	65-Créer une fonction (2).....
33-Un graphique en x-y.....	66-Utilisation de la fonction.....

Logiciel et bases de données

André Bouchier

Durée : environ 6 heures

Version du 12 avril 2006

Copyright © André Bouchier.

© 2006, André Bouchier (20 Janvier 2006)

Permission est accordée de copier et distribuer ce document, en partie ou en totalité, dans n'importe quelle langue, sur n'importe quel support, à condition que la notice © ci-dessus soit incluse dans toutes les copies. Permission est accordée de traduire ce document, en partie ou en totalité, dans n'importe quelle langue, à condition que la notice © ci-dessus soit incluse.

1) Lire des données à partir de R ?

- ◆ Les gestionnaires de base de données (mySql, MS-access, 4D,...) ont des fonctions d'exportation de données au format texte.
- ◆ R peut lire ces fichiers textes avec facilité.
- ◆ Le fichier **Sen.txt**, peut être lu grâce à la commande suivante :

```
setwd("g:/r_odbc/datadir")  
x<-read.table("Sen.txt",header=T,dec="," , " , sep=" ; ")
```

names(x)

```
[1] "Numero"      "Village"     "Actifs"      "SAP"         "SAU"         "Anes"  
[7] "CVTrait"     "BovTrait"    "VachTrait"   "BovViande"   "Ovins"       "Caprins"
```


2) Extraire et manipuler des données

➡ On peut extraire du `data.frame` `x` les données nécessaires au traitement statistique envisagé.

➡ Par exemple, on extrait les exploitations pour lesquels la surface utile (SAU) est plus de 10 ha et qui appartiennent au village 2. On ne conservera que les variables *Actifs*, *SAU*, *CVTrait* :

```
x[x[2]==2 & x[5]>10, c(3,5,7)]
```

	Actifs	SAU	CVTrait
8	10.7	16.50	2
47	8.0	11.75	3
48	8.5	11.25	0
55	5.0	11.00	1

3) Utiliser des bases de données

- ◆ Ces fonctionnalités peuvent être suffisantes pour des traitements de fichiers de données simples et *personnels*.
- ◆ Si vous partagez votre base de données avec d'autres utilisateurs, il y a de fortes chances qu'elle soit implantée dans une machine distante (un serveur) et gérée par un administrateur.
- ◆ La base de données étant sur une seule machine, l'administrateur peut gérer les problèmes de confidentialité.
- ◆ Les mises à jours seront disponibles instantanément pour tous les utilisateurs.
- ◆ Les outils d'interrogation d'une base de données distante sont indépendants du logiciel situé sur le serveur et du format des données.

4) Utiliser des bases de données avec R

- ◆ Des outils d'interrogation de base des données ont été intégrés à R.
- ◆ C'est à travers les fonctionnalité **ODBC** des systèmes MS-Windows que nous interrogerons les bases de données.
- ◆ Nous utiliserons pour ça le **langage SQL** (Langage structuré d'interrogation de bases de données).
- ◆ La bibliothèque utilisée pour ces interrogations se nomme **RODBC**.

5) Les formats des données utilisées dans ce cours (1) :

Le format Access :

◆ Microsoft Access 2002 s'illustre par son extrême richesse au sein de la famille Office. Avec tout un éventail de nouvelles fonctionnalités, Access 2002 sera tout autant apprécié des utilisateurs habitués à travailler avec des bases de données, que des novices qui seront séduits par la simplicité d'utilisation de ce produit [...] Les développeurs et les utilisateurs intensifs pourront créer des solutions de bases de données encore plus performantes pour des analyses rapides et aisées.
(source : publicité pour MS-Access)

◆ SGBDR de Microsoft, fourni avec Office dans sa version pro, basé sur le moteur « Jet » et qui stocke toutes les données dans un seul fichier. Des bugs stupéfiants y ont été découverts, en particulier pendant l'été 1998 : Access peut dans certains cas perdre tout simplement les informations qu'on lui confie. (source <http://www.linux-france.org/>)

6) Les formats des données utilisées dans ce cours (2) :

Le format mySql :

- ◆ Deux grandes techniques de bases de données s'affrontent : celle à base de fichiers plats structurés nécessitant un moteur sur chaque poste (dBase, Access,...) et celle à base de serveur de données (mySQL, Paradox,...).
- ◆ Dans ce modèle client/serveur, la base de donnée est installée sur une machine, le serveur. Elle peut-être interrogée depuis n'importe quel poste de travail.
- ◆ Les données sont indépendantes des demandes de traitements de l'information et du logiciel client.
- ◆ **MySQL** est un Système de Gestion de Bases de Données (SGBD) fonctionnant sous Linux et Windows. MySQL est sous Licence GPL (aussi bien sous Linux que Windows).

7) Les données utilisées dans ce cours (1) :

Le fichier « **voitur94** » :

◆ Ce fichier de données présente une image des principaux constructeurs automobiles en 1994. Les modèles commercialisés à cette époque sont décrits à travers les variables suivantes :

"IDENTIFIANT"	le nom du modèle
"PUISS_ADMIN"	la puissance administrative en CV fiscaux
"CYLINDREE"	la cylindrée du modèle
"MOTEUR"	le type de motorisation (essence ou diesel)
"TRANSMISSION"	transmission avant ou arrière
"LONGUEUR"	longueur du véhicule
"LARGEUR"	largeur du véhicule
"SURFACE"	surface du véhicule
"POIDS_TOTAL"	poids total du véhicule
"VIT_MAXI"	vitesse maximale en km/h
"DEP_ARRET"	durée du 400 m départ-arrêt en secondes
"MARQUE"	le constructeur
"MARQUE_QL"	le constructeur (données codée numérique)
"CONSO_MOYE"	la consommation moyenne du véhicule

8) Les données utilisées dans ce cours (2) :

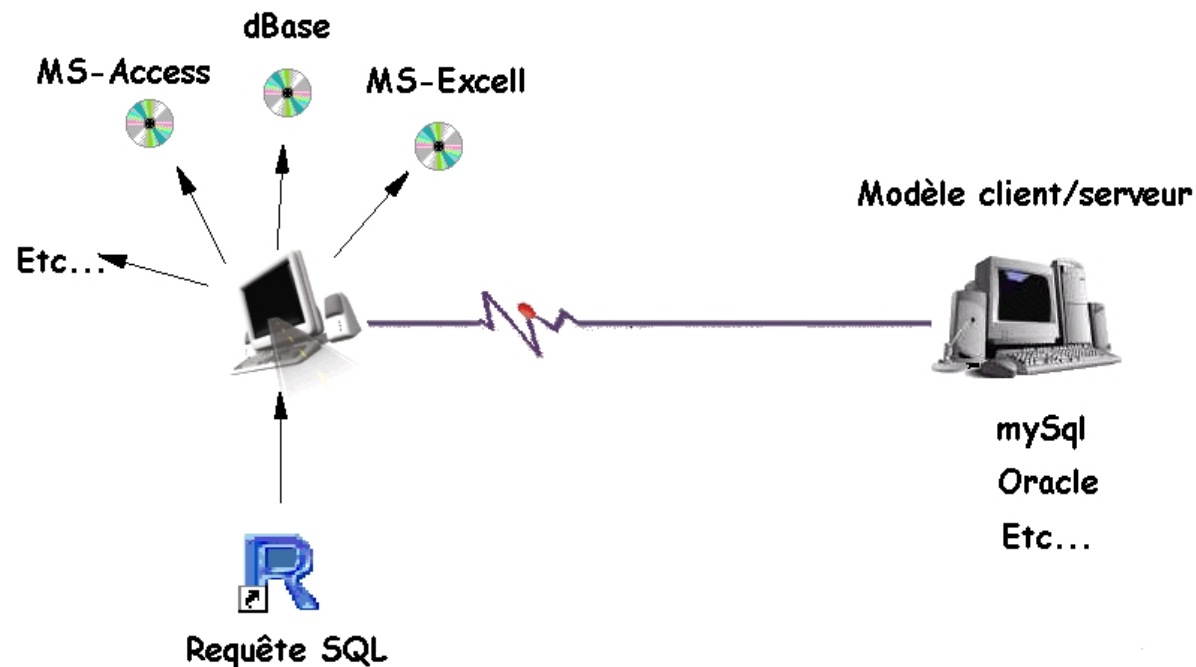
Le fichier « Sénégal » :

◆ Données sur la traction animale au Sénégal en 1983. Le fichier comporte les variables suivantes :

"Numero"	un numéro identifiant l'exploitation agricole
"Village"	le code du village (Quartier)
"Actifs"	le nombre d'actifs dans l'exploitation
"SAP"	la surface possédée
"SAU"	la surface utile
"Anes"	le nombre d'ânes de traction
"CVTrait"	le nombre de chevaux de trait
"BovTrait"	le nombre de bovins de trait
"VachTrait"	le nombre de vache de trait
"BovViande"	le nombre de bovins extensifs
"Ovins"	le nombre d'ovins
"Caprins"	le nombre de caprins

9) L'O.D.B.C. :

- ◆ Open DataBase Connectivity. Couche logicielle devant permettre à une application Windows d'accéder de façon transparente à une base de données SQL. (© Le monde informatique).
- ◆ La technologie ODBC permet d'interfacer de façon standard une application à n'importe quel serveur de bases de données, à condition que celui-ci possède un driver ODBC.



10) Trouver le gestionnaire O.D.B.C. :

- ◆ Le gestionnaire **ODBC** est présent sur les systèmes Windows. Il existe des implémentations sur d'autres plates-formes, notamment UNIX/Linux et mac-OSX
- ◆ Sous Windows 2000 et XP, il faudra aller chercher l'icône **ODBC** dans le panneau de configuration, puis dans les outils d'administration.
- ◆ Sous Windows 95 et 98 le gestionnaire **ODBC** est disponible dans le panneau de configuration.

Note : On appelle **DSN** (Data Source Name) la source de données qui sera accessible par l'intermédiaire de **ODBC**.



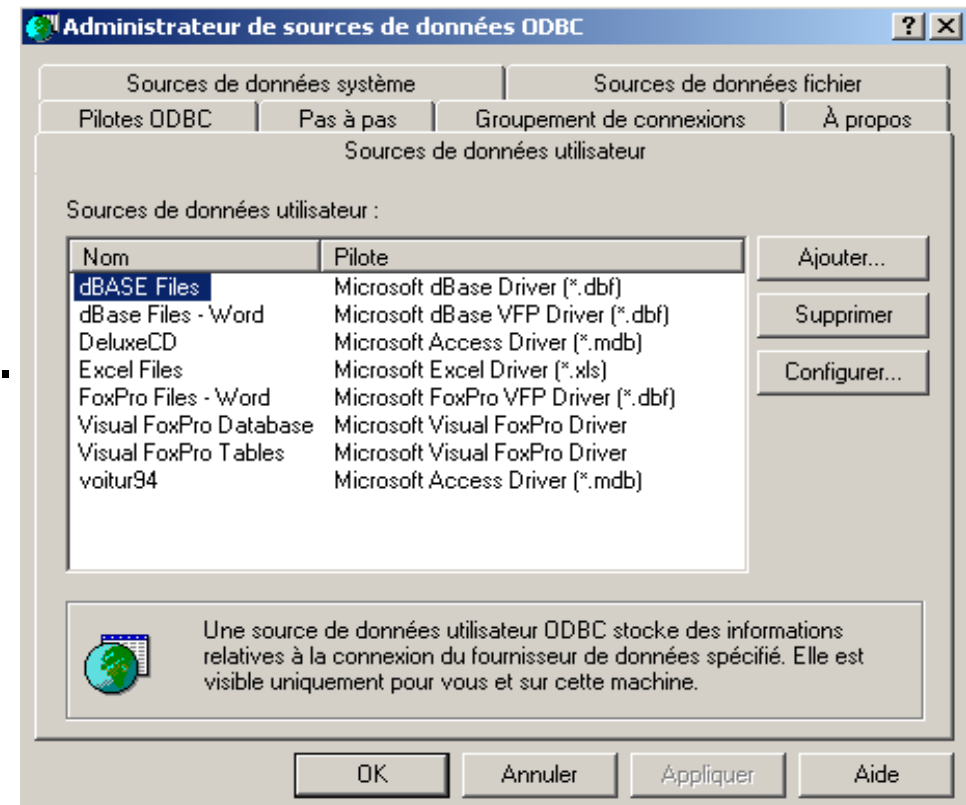
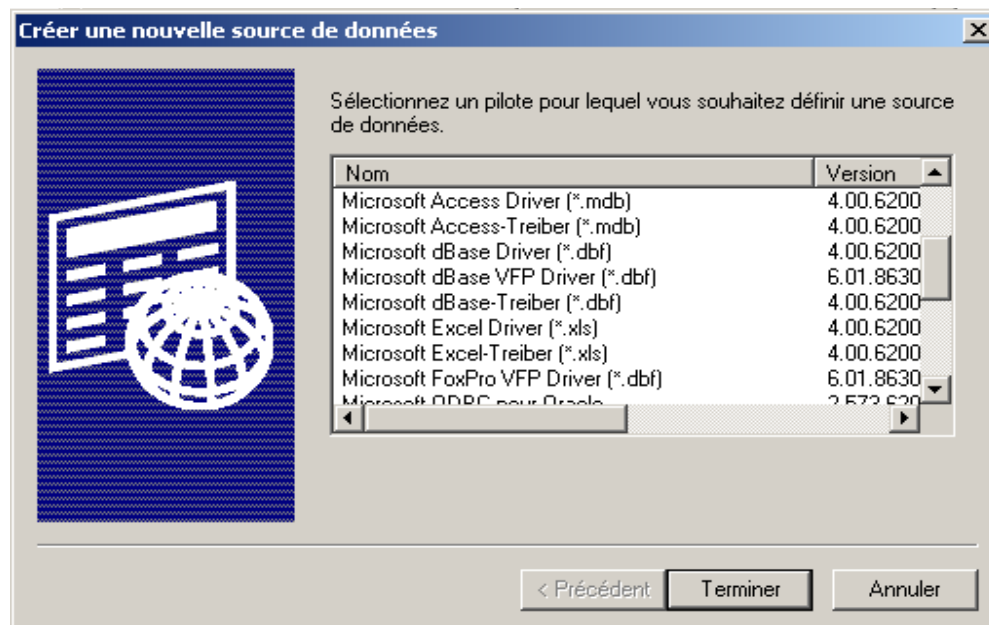
11) Créer une ressource ODBC à partir d'un fichier MS-Access :

➤ L'installation de l'ODBC se fera en trois étapes :

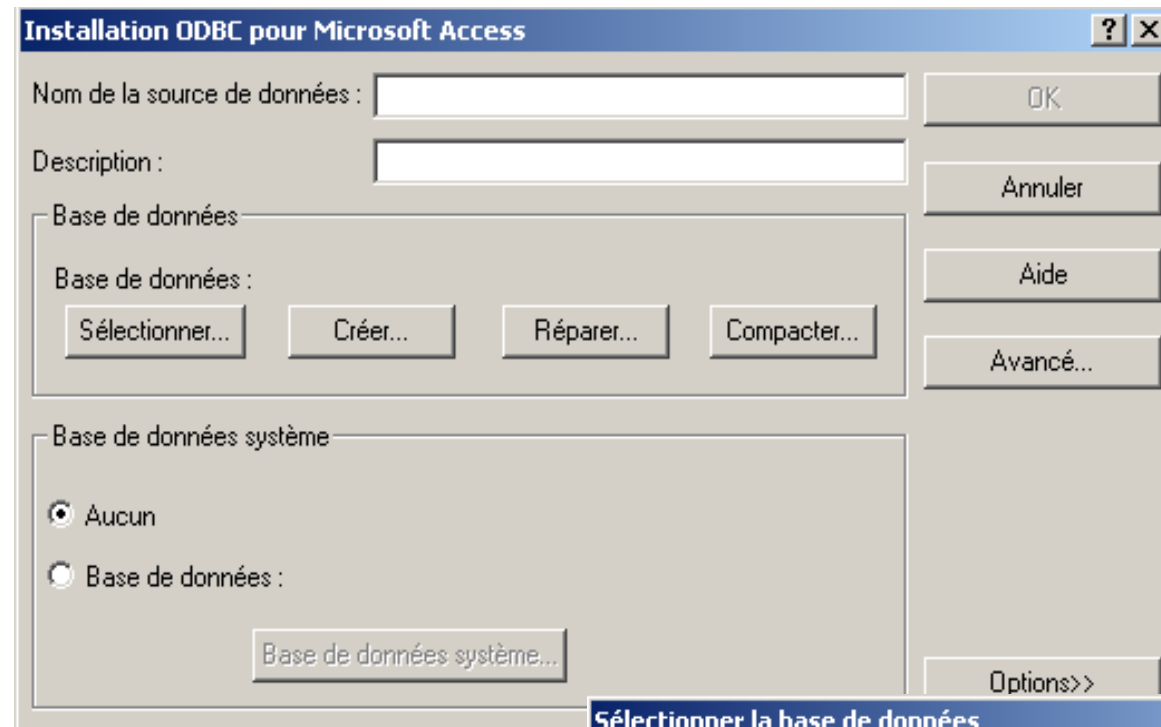
➔ Onglet « source de données utilisateur »

➔ Cliquer sur ajouter

➔ Choisir le pilote de base de données.
Ici, « Microsoft Access Driver »

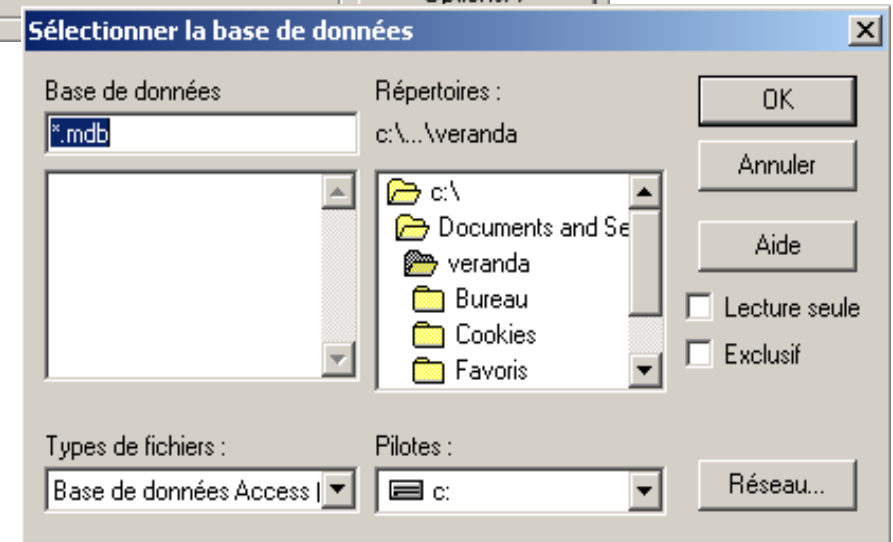


➡ Donner un nom à la ressource ODBC :



➡ Sélectionner le fichier de base de données :

La base de données Access **voitur94.mdb** est maintenant reconnue par l'ODBC sous le nom de **voitur94**.



12) Se connecter à une ressource ODBC :

Lancer R

➡ On charge la bibliothèque de fonctions **RODBC**.

```
library(RODBC)
```

➡ Fermer toutes les connections **ODBC** en cours

```
odbcCloseAll()
```

➡ On connecte R avec la base de données **ODBC**. Celle-ci s'appelle désormais « **bdd** » (ou le nom que vous voulez)

```
bdd <- odbcConnect("voitur94", "", "")  
# "", "" sont pour userId et password
```

13) Obtenir des informations sur le pilote ODBC :

➡ Tout savoir au sujet de la version du pilote ODBC installé sur votre machine et sur la base de données à laquelle vous êtes connecté.

odbcGetInfo (bdd)

DBMS_Name	DBMS_Ver	Driver_ODBC_Ver	Data_Source_Name
"ACCESS"	"03.50.0000"	"03.51"	"voitur94"
Driver_Name	Driver_Ver	ODBC_Ver	Server_Name
"odbcjt32.dll"	"04.00.6200"	"03.52.0000"	"ACCESS"

14) RODBC : 2 familles de fonctions

➡ La bibliothèque ODBC fournit 2 familles de fonctions :

➡ Les fonctions commençant par **ODBC*** fournissent des commandes de bas niveau vers le pilote ODBC.

odbcopen()
odbcclose()

➡ Les fonctions commençant par **SQL*** sont des commandes de haut niveau construites pour lire, sauver, copier et manipuler les données entre les data frame et les bases de données.

sqlCopy()
sqlCopyTable()

Remarque : Many connections can be open at once to any combination of dsn/hosts

15) Liste des fichiers contenus dans la base de données

➡ Liste des fichiers contenus dans la base de données. L'option **errors=TRUE** indique à R qu'il doit s'arrêter et afficher un message en cas d'erreur. Si **errors= FALSE**, R n'affichera que -1 en cas d'erreur :

```
sqlTables(bdd, errors = TRUE)
```

		TABLE_CAT	TABLE_SCHEM	TABLE_NAME	TABLE_TYPE	
REMARKS						
1	D:\\R_ODBC\\datadir\\voitur94		<NA>	MSysAccessObjects	SYSTEM	TABLE <NA>
2	D:\\R_ODBC\\datadir\\voitur94		<NA>	MSysACEs	SYSTEM	TABLE <NA>
3	D:\\R_ODBC\\datadir\\voitur94		<NA>	MSysModules	SYSTEM	TABLE <NA>
4	D:\\R_ODBC\\datadir\\voitur94		<NA>	MSysModules2	SYSTEM	TABLE <NA>
5	D:\\R_ODBC\\datadir\\voitur94		<NA>	MSysObjects	SYSTEM	TABLE <NA>
6	D:\\R_ODBC\\datadir\\voitur94		<NA>	MSysQueries	SYSTEM	TABLE <NA>
7	D:\\R_ODBC\\datadir\\voitur94		<NA>	MSysRelationships	SYSTEM	TABLE <NA>
8	D:\\R_ODBC\\datadir\\voitur94		<NA>	SEN		TABLE <NA>
9	D:\\R_ODBC\\datadir\\voitur94		<NA>	vacances		TABLE <NA>
10	D:\\R_ODBC\\datadir\\voitur94		<NA>	voitur94		TABLE <NA>

Remarque : Les tables **SYSTEM** concernent le fonctionnement interne d'Access, ne pas y toucher.

16) Copier une table d'une base de données dans un data frame

➡ Trois tables sont disponibles : SEN, vacances, voiture94. Attention, à la case des caractères !

➡ On peut copier toute la table voiture94 dans le data.frame **bolides** avec la fonction **sqlFetch()**

```
bolides<-sqlFetch(bdd, "voiture94")
```

```
names(bolides)
```

```
[1] "IDENTIFIANT" "PUISS_ADMI" "CYLINDREE" "MOTEUR" "TRASMISSIO"  
[6] "LONGUEUR" "LARGEUR" "SURFACE" "POIDS_TOTA" "VIT_MAXI"  
[11] "DEP_ARRET" "MARQUE" "MARQUE_QL" "CONSO_MOYE"
```


17) Extraire des données avec la fonction `sqlQuery()` :

- ➡ On veut la liste de tous les véhicules dont la vitesse maxi est supérieure à 200 Km/h

```
sqlQuery(bdd, "SELECT IDENTIFIANT,VIT_MAXI FROM voiture94  
WHERE VIT_MAXI > 200")
```

	IDENTIFIANT	VIT_MAXI
1	Alfa-Roméo 155 2.0	205
2	Alfa-Roméo 164 2.5 T	202
3	BMW 730i	222
4	Citroën XM 2.0i	201
5	Citroën XM V6	222
6	Ford Scorpio 2900i	201
7	Peugeot 605 Sv24	235

18) Exercices :

➡ Exercice 1 :

Créez une liaison ODBC avec le fichier de données senegal.mdb
Copiez la table senegal dans un data.frame.

➡ Exercice 2 :

Représentez graphiquement la relation entre le nombre d'actifs (Actifs) et la surface agricole possédée (SAP).

19) Les commandes SQL : select

➡ La syntaxe générale

SELECT liste de variable **FROM** table de données **WHERE** conditions

➡ Par exemple, sélectionner tout le tableau de données :

```
SELECT * FROM voiture94
```

➡ Pour lancer la requête

```
x<-sqlQuery(bdd, "SELECT * FROM voiture94")
```

20) Les commandes SQL, la clause WHERE :

➡ On recherche la liste des véhicules dont la puissance fiscale est supérieure à 11 CV :

```
sqlQuery(bdd, "SELECT MARQUE, PUISS_ADMI FROM voiture94  
WHERE PUISS_ADMI>11")
```

	MARQUE	PUISS_ADMI
1	BMW	16
2	Citroën	12
3	Citroën	16
4	Ford	15
5	Peugeot	16

21) Les commandes SQL, un caractère générique :

➡ Ne conserver que les marques ayant un o dans leur nom (Ford, Citroën...)

➡ % remplace une chaîne de caractères de longueur quelconque

```
sqlQuery(bdd, "SELECT * FROM voiture94 WHERE MARQUE LIKE '%o%' ")
```

22) Les commandes SQL, une liste pour sélectionner :

➡ Ne conserver que les BMW et les Fiat :

```
sqlQuery(bdd, "SELECT MARQUE,CYLINDREE FROM voiture94  
WHERE MARQUE IN ('BMW', 'Fiat') ")
```

	MARQUE	CYLINDREE
1	BMW	1596
2	BMW	2498
3	BMW	1796
4	BMW	2498
5	BMW	2986
6	Fiat	999
7	Fiat	1372
8	Fiat	1367
9	Fiat	1756
10	Fiat	1929
11	Fiat	1756
12	Fiat	1929
13	Fiat	1995
14	Fiat	2500

23) Les commandes SQL, la clause BETWEEN :

- Ne conserver que les véhicules dont la puissance fiscale est comprise entre 10 et 12 CV

```
sqlQuery(bdd, "SELECT MARQUE,CYLINDREE, PUISS_ADMI FROM voiture94  
WHERE PUISS_ADMI BETWEEN 10 AND 12 " )
```

	MARQUE	CYLINDREE	PUISS_ADMI
1	Alfa-Roméo	1995	10
2	BMW	1796	10
3	Citroën	954	12
4	Citroën	1998	11
5	Citroën	1998	11
6	Fiat	1995	10
7	Ford	1998	10
8	Renault	2165	11

24) Les commandes SQL, Le préfixe NOT :

➡ Les commandes **LIKE**, **IN** et **BETWEEN** peuvent être utilisées avec le préfixe **NOT**. Cela devient :

➡ Toutes les marques n'ayant pas de o dans leur nom :

```
SELECT * FROM voiture94 WHERE MARQUE NOT LIKE '%o%'
```

➡ Toutes les marques autres que Fiat et BMW :

```
SELECT MARQUE,CYLINDREE FROM voiture94 WHERE MARQUE NOT IN  
( 'BMW', 'Fiat' )
```

➡ Tous les modèles dont la puissance fiscale est plus grande que 7 CV **et** plus petite que 4 CV :

```
SELECT IDENTIFIANT, PUISS_ADMI FROM voiture94 WHERE PUISS_ADMI NOT  
BETWEEN 4 AND 7
```


25) Quelques calculs sur le fichier "voitur94" :

➡ Compter :

Combien de modèles dans chaque marque ?

```
sqlQuery(bdd, "SELECT MARQUE,Count(MARQUE) AS NbMarque  
FROM voiture94 GROUP BY MARQUE")
```

	MARQUE	NbMarque
1	Alfa-Roméo	4
2	BMW	5
3	Citroën	13
4	Fiat	9
5	Ford	8
6	Peugeot	10
7	Renault	13

◆ Trier :

Combien de modèles dans chaque marque avec un trie descendant par nom de marques ?

```
sqlQuery(bdd, "SELECT MARQUE,Count(MARQUE) AS NbMarque  
FROM voiture94 GROUP BY MARQUE ORDER BY MARQUE DESC")
```

	MARQUE	NbMarque
1	Renault	13
2	Peugeot	10
3	Ford	8
4	Fiat	9
5	Citroën	13
6	BMW	5
7	Alfa-Roméo	4

On utilisera l'option **ASC** pour un trie ascendant

Attention ! On ne peut trier que selon un champ présent dans la clause SELECT

Combien de modèles dans chaque marque avec un trie par nombre de modèles ?

```
sqlQuery(bdd, "SELECT MARQUE,Count(MARQUE) AS NbMarque  
FROM voiture94 GROUP BY MARQUE ORDER BY Count(MARQUE) DESC")
```

	MARQUE	NbMarque
--	--------	----------

1	Renault	13
2	Citroën	13
3	Peugeot	10
4	Fiat	9
5	Ford	8
6	BMW	5
7	Alfa-Roméo	4

◆ Les maxi et mini :

Quelles sont les consommations maxi et mini de chaque marque ?

```
sqlQuery(bdd, "SELECT MARQUE, MAX(CONSO_MOYE) AS ConsoMax,  
MIN(CONSO_MOYE) AS ConsoMin FROM voitures94 GROUP BY MARQUE ")
```

	MARQUE	ConsoMax	ConsoMin
1	Alfa-Roméo	8.0	6.4
2	BMW	11.1	6.9
3	Citroën	9.5	4.2
4	Fiat	8.9	5.3
5	Ford	10.8	6.1
6	Peugeot	9.5	4.8
7	Renault	9.7	5.5

◆ Les sommes et moyennes :

On utilisera les fonction **AVG()** pour calculer des moyennes et **SUM()** pour des sommes

◆ Les valeurs manquantes :

On accédera aux valeurs manquantes grâce aux fonctions **ISNULL()** et **NOT ISNULL()**

Liste des véhicules dont les champs DEP_ARRET et POIDS_TOTA sont manquants:

```
sqlQuery(bdd, "SELECT IDENTIFIANT, DEP_ARRET, POIDS_TOTA  
FROM voiture94 WHERE ISNULL(DEP_ARRET) AND ISNULL(POIDS_TOTA) ")
```

	IDENTIFIANT	DEP_ARRET	POIDS_TOTA
1	Ford Escort 1400	NA	NA
2	Ford Escort 1800 TD	NA	NA

Liste des véhicules dont le champ DEP_ARRET est manquant et le champs POIDS_TOTA n'est **pas** manquant :

```
sqlQuery(bdd, "SELECT IDENTIFIANT, DEP_ARRET, POIDS_TOTA FROM  
voitur94 WHERE ISNULL(DEP_ARRET) AND NOT ISNULL(POIDS_TOTA) ")
```

	IDENTIFIANT	DEP_ARRET	POIDS_TOTA
1	BMW 525 TD	NA	1465
2	Fiat Panda 1000i.e	NA	715
3	Ford Fiesta 1400	NA	840
4	Ford Fiesta TD	NA	900
5	Ford MONDEO 1800i	NA	1277
6	Ford MONDEO TD	NA	1277
7	Ford Scorpio 2000i	NA	1245
8	Ford Scorpio 2900i	NA	1345

26) Les calculs avec des valeurs manquantes :

- ➡ Le moteur SQL tient compte des données manquantes pour calculer les moyennes :

```
mean (na.omit (bolides$DEP_ARRET) )
```

```
[1] 34.33654
```

```
sqlQuery (bdd, "SELECT AVG (DEP_ARRET) AS MoyDepArr FROM voitur94")
```

```
MoyDepArr
```

```
[1] 34.33654
```

27) Exercice, requêtes sélection sur le fichier "sénégal" :

◆ Exercice 3 :

Sélectionner les exploitations ayant un nombre d'actifs > 2 et une surface possédée < 4

◆ Exercice 4 :

Combien d'exploitations répondent à ces critères ?

◆ Exercice 5 :

Combien d'exploitations par Quartier ?

◆ Exercice 6 :

Quelle est la surface moyenne des exploitations par Quartier ?

28) Création d'une nouvelle table à partir de R :

- On utilise le data.frame `iris` (données d'exemple fournies avec R). On copiera `iris` dans la base de données MS-Access `voitur94`.
- Vous utiliserez la fonction `sqlSave()`
- Pour en savoir plus, tapez : `?sqlSave`

29) Utilisation de requêtes enregistrées :

- ◆ Ouvrez le fichier voiture94.mdb dans MS-Access
- ◆ Créez une requête : vous conserverez les champs IDENTIFIANT et PUISS_ADMI. Vous ne conserverez que les modèles dont le PUISS_ADMI est > 10
- ◆ Sauvegardez cette requête sous le nom de "question1". Vous pouvez fermer Access
- ◆ Représentez graphiquement ces données avec R

30) Supprimer des données d'une table :

- ➡ Attention aux contraintes d'intégrité référentielle de la base de données.
- ➡ Tenter de violer les règles de cohérence de la base de données peut aboutir au mieux à ce que l'opération échoue, au pire à des destructions de données en cascade.
- ➡ Par exemple pour détruire toutes les lignes de la table **Marques** pour lesquelles **NOM** est "**Total**" :

```
sqlQuery(bdd, "DELETE FROM Marques WHERE NOM='Total' ")
```

Attention, l'instruction suivante efface toutes les données :

```
SqlQuery(bdd, "DELETE * FROM Marques")
```

31) Supprimer une table

➡ on peut utiliser la fonction **sqlQuery()** pour envoyer une requête SQL qui effacera la table :

```
sqlQuery(bdd, "DROP Table Marques")
```

➡ ou bien utiliser une fonction plus élaborée **sqlDrop()** qui écrira la requête à notre place :

```
sqlDrop(bdd, "Marques")
```

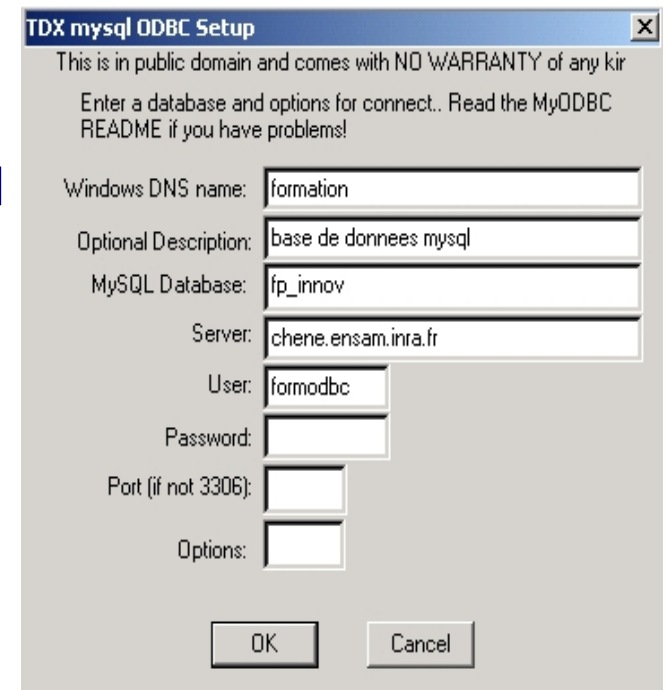
32) Connection distante - MySql ODBC :

➡ La connection ODBC :

Créer une liaison ODBC avec un pilote MySql. Le pilote ODBC pour les bases de données MySql est libre et gratuit, disponible sur internet (voir les liens).

Le premier champ à remplir est « **Windows DSN name** ». C'est le **Data Source Name**, le nom (le surnom) que vous choisissiez.

Prenez un nom explicite afin de vous y retrouver facilement. Ici, le dsn retenu est "**formation**". (explicite ?)



TDX mysql ODBC Setup

This is in public domain and comes with NO WARRANTY of any kind.

Enter a database and options for connect.. Read the MyODBC README if you have problems!

Windows DNS name: formation

Optional Description: base de donnees mysql

MySQL Database: fp_innov

Server: chene.ensam.inra.fr

User: formodbc

Password:

Port (if not 3306):

Options:

OK Cancel

Le deuxième champ est le nom de la base de données sur la machine hôte. Sur le serveur [chene](#), la base de données mySql s'appelle "[fp_innov](#)".

Le troisième champ concerne la machine sur laquelle est installée le serveur. Ici, il s'agit de la machine UNIX « [chene](#) » appartenant au domaine « [ensam.inra.fr](#) ».

Les deux champs suivant concernent votre nom d'utilisateur "[formodbc](#)" et votre mot de passe.

Attention ! Vous avez tous les droits sur cette base de données, alors ne détruisez pas les tables qui ne vous appartiennent pas.

33) MySql ODBC, vérification de la connection ODBC :

➡ On peut maintenant lancer R et utiliser cette base de données :

```
library(RODBC) # appel de la bibliothèque de fonction RODB
```

```
bdd <- odbcConnect("formation", "", "")
```

➡ La fonction `odbcConnect()` permet de se connecter directement en complétant les options nom et mot de passe :

```
bdd <- odbcConnect("formation", "formodbc", "mot de passe")
```

➡ Mais il faut faire attention à ne pas laisser votre mot de passe en clair dans les lignes de programme R

```
odbcGetInfo(bdd) # information sur la base de données
```

```
sqlTables(bdd) # quelles tables dans la base de données ?
```

34) Ajout de données dans la base MySql:

➡ La base "formation" contient la table "protein". Nous allons y ajouter le data.frame "chickwts". Attention, donnez un nom différent pour chaque stagiaire.

```
sqlSave(bdd, iris, verbose = F, tablename="chickwts01",  
rownames = F, append = T)
```

➡ On peut vérifier si les copies se sont bien passées :

```
sqlTables(bdd)
```


35) Interrogation de la base MySql à partir de R :

- ◆ Les requêtes vers MySql se lancent de la même manière qu'avec la base de données MS-Access

```
sqlQuery(bdd, "SELECT feed, Avg(weight) AS MoyPoids,  
FROM chickwts01 GROUP BY feed")
```

- ◆ Supprimer la table `chickwts01` de la base de données mySql :

```
sqlDrop(bdd, "chickwts01")
```

```
sqlTables(bdd)
```

```
odbcCloseAll() #ferme toutes les connections ODBC
```

36) Exercice :

◆ exercice 7 :

Copiez le fichier "**senegal**" de la base de données "**senegal.mdb**" vers la base de données MySql (après avoir créé la ressource ODBC). Vérifier la présence des nouvelles tables.

Attention, donnez lui un nom différent pour chaque stagiaire.

◆ exercice 8 :

Calculez le nombre moyen d'actifs par ha de SAU pour chaque village de la table "**senegal**".

◆ exercice 9 :

Effacez les tables que vous avez créées sur la base MySql.

37) Quelques liens :

Pilote ODBC MySql pour windows et linux :

<http://www.mysql.com/>

Comment installer MyODBC

http://www.mysql.com/doc/fr/Installing_MyODBC.html

Formation SQL, un très bon site en français :

<http://sqlpro.developpez.com/>

Site NeXen Pour suivre l'actualité PHP et MySQL :

<http://www.nexen.net/index.php>

Documentation MySQL 4.1.0 en Français

<http://dev.nexen.net/docs/mysql/>

Le langage R présentation de P. Raffinat, Département Statistique et Traitement Informatique des Données IUT de Pau et des Pays de l'Adour :

http://www.univ-pau.fr/~artouste/fr/composants/R/aide_R.html

Data mining avec R :

<http://sic.epfl.ch/SA/publications/FI01/fi-sp-1/sp-1-page45.html>

The Comprehensive R Archive Network

<http://lib.stat.cmu.edu/R/CRAN/>

38) Corrigé des exercices :

◆ Exercice 1 :

Créez une liaison ODBC avec le fichier de données senegal.mdb. Copiez le dans un data.frame.

```
library(RODBC)
odbcCloseAll()
sen <- odbcConnect("senegal", "", "")
odbcGetInfo(bdd)
senegal<-sqlFetch(sen, "senegal")
names(senegal)
```

◆ Exercice 2 :

Représentez graphiquement la relation entre le nombre d'actifs et la surface possédée.

```
plot(senegal$Actifs*senegal$SAP, main="Relation entre Nb  
d'Actifs \n et Surface possédée")
```

◆ Exercice 3 :

Sélectionner les exploitations ayant un nombre d'actifs > 2
et une surface possédée < 4

```
sqlQuery(bdd, "SELECT * FROM senegal  
WHERE Actifs > 2 AND SAP<4")
```

◆ Exercice 4 :

Combien d'exploitations répondent à ces critères ?

```
sqlQuery(bdd, "SELECT Count(Actifs) AS Nb FROM senegal  
WHERE Actifs > 2 AND SAP<4")
```

Nb

1 35

➡ Exercice 5 :

Combien d'exploitations par Village ?

```
sqlQuery(bdd, "SELECT Village, Count(Village) AS NbDeVill  
FROM senegal GROUP BY Village")
```

	QU	NbDeVill
1	1	58
2	2	15
3	3	22
4	4	9
5	5	68
6	6	31
7	7	19
8	8	14
9	9	9

➡ Exercice 6 :

Quelle est la surface moyenne des exploitations par Quartier ?

```
sqlQuery(bdd, "SELECT Village, Avg(SAP) AS MoyDeSP  
FROM senegal GROUP BY Village")
```

	QU	MoyDeSP
1	1	7.725000
2	2	7.430000
3	3	8.388636
4	4	6.333333
5	5	8.705882
6	6	9.137097
7	7	7.776316
8	8	7.625000
9	9	7.805556

◆ Exercice 7 :

Copiez le fichier "**senegal**" de la base de données "**senegal.mdb**" vers la base de données MySql (après avoir créé la ressource ODBC). Attention, donnez-lui un nom différent pour chaque stagiaire. Vérifier la présence des nouvelles tables.

```
#lecture du tableau de données "senegal"
```

```
library(RODBC)
sen <- odbcConnect("senegal", "", "")
don<-sqlFetch(sen,"senegal")
odbcClose(sen)
```

```
#écriture du tableau "senegal" dans la base mySQL
```

```
bdd <- odbcConnect("formation", "", "")
sqlSave(bdd, don, rownames=F, tablename="senegal", append=F)
sqlTables(bdd)
```


➡ Exercice 8 :

Calculez le nombre moyen d'actifs par ha de SAU pour chaque village de la table "**senegal**".

```
sqlQuery(bdd,"SELECT Village, AVG(Actifs/SAU) AS ActHa FROM senegal  
GROUP BY Village")
```

	Quartier	x
1	1	0.6057581
2	2	0.6287979
3	3	0.5469784
4	4	0.5520493
5	5	0.6157053
6	6	0.5334181
7	7	0.5561242
8	8	0.4365168
9	9	0.5484554

➡ Exercice 9 :

Effacez les tables que vous avez créées sur la base MySql.

```
sqlDrop(bdd, "senegal")
```

Formation Permanente

André Bouchier

CENTRE INRA de MONTPELLIER

FORMATION AU LOGICIEL



Programmation et interfaces graphiques

(durée : environ 6 heures)

version du 13 février 2006

Copyright © André Bouchier.

© 2006, André Bouchier (20 Janvier 2006)

Permission est accordée de copier et distribuer ce document, en partie ou en totalité, dans n'importe quelle langue, sur n'importe quel support, à condition que la notice © ci-dessus soit incluse dans toutes les copies. Permission est accordée de traduire ce document, en partie ou en totalité, dans n'importe quelle langue, à condition que la notice © ci-dessus soit incluse.

1-Programmer avec R

- R est à la fois un logiciel de statistique et un langage de programmation
- Avec R, on peut, par exemple, programmer des boucles afin d'analyser successivement différents jeux de données.
- On peut aussi combiner ou empiler dans le même programme plusieurs fonctions statistiques pour réaliser des analyses complexes.
- R est un langage interprété et non compilé

2-Quelques éléments à connaître sur les listes (1)

- Une liste est formée d'éléments pouvant être de types différents
- Par exemple, une liste contenant 3 éléments : le tableau de données iris, le nom des variables de ce tableau et les fréquences de chaque espèce

```
data(iris)
```

```
maliste<-list(iris, names(iris), table(iris[,5]))
```

- Pour accéder aux élément de la liste

```
maliste[1]
```

3-Quelques éléments à connaître sur les listes (2)

- Il est conseillé de nommer chaque élément de la liste

```
names(maliste) <- c("donnees", "noms", "frequence")
```

```
maliste$noms
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length"  
"Petal.Width"   "Species"
```

4-Quelques éléments à connaître sur les listes (3)

- Une autre méthode pour nommer chaque élément de la liste

```
data(iris)
```

```
maliste<-list(donnees=iris,noms= names(iris),  
frequence=table(iris[,5]))
```

- On peut alors interroger plus précisément la liste

```
typeof(maliste$noms)
```

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

```
maliste$noms[1]
```

```
[1] "Sepal.Length"
```


5-Quelques éléments à connaître sur les listes (4)

● Exercice 1 :

On utilisera le data.frame "Orange"

Créer une liste contenant

- une description des données (en une phrase),
- le tableau de données lui-même,
- les valeurs maxi des circonférences pour chaque arbre

Remarque : vous aurez besoin de la fonction aggregate()

6-Tâches complexes et répétitives, fonction `by()`

- Appliquer une fonction pour chaque niveau d'un facteur

```
data(iris)
attach(iris)
by(Sepal.Length, Species, mean)
```

- Appliquer la fonction aux éléments d'une liste

```
liste<-list(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width)
names(liste)<-c("Sepal.Length", "Sepal.Width", "Petal.Length",
"Petal.Width")
by(liste, Species, mean)
detach(iris)
```

7-Tâches complexes et répétitives, fonction `by()`

● Exercice 2 :

Calculez les coefficients de corrélation de `pearson` entre l'age et la circonférence des orangers pour chaque arbre du data.frame `Orange`

8-Tâches complexes et répétitives : **lapply()**

- **lapply()** applique une fonction à une liste de vecteurs

```
attach(iris)
```

```
liste<-list(Sepal.Length, Sepal.Width, Petal.Length,  
Petal.Width)
```

```
names(liste)<-c("Sepal.Length", "Sepal.Width",  
"Petal.Length", "Petal.Width")
```

```
lapply(liste,mean)
```

```
lapply(liste, quantile, probs = c(0, .33, 0.66, 1))
```

```
detach(iris)
```

● **Exercice 3 :**

en utilisant **lapply()**, centrer et réduire les variables quantitatives du tableau de données **Iris**. Vous devez obtenir un data.frame.

9-Tâches complexes et répétitives : replicate()

- Pour répéter l'évaluation d'une expression (met en général en oeuvre la génération de tirages aléatoires)

- **exemple :**

la moyenne d'un tirage aléatoire

```
mean(sample(iris$Sepal.Length , size=20, replace = FALSE))
```


l'histogramme de la moyenne sur 1000 tirages aléatoires

```
hist(replicate(1000,mean(sample(iris$Sepal.Length,size=20))))
```

- **Exercice 4 :**

Faites le même exercice en calculant l'écart-type sur des tailles d'échantillon de 30 individus. Pour améliorer la lisibilité du graphique, vous utiliserez la variable Sepal.Length centrée-réduite

10-Echantillonner sur plusieurs variables

- Attention, danger 

```
attach(iris)
cor(sample(Sepal.Length, size=20), sample(Petal.Length, size=20))
detach(iris)
```

11-Echantillonner sur plusieurs variables

- Pour échantillonner des paires de valeurs

```
data (swiss)
don<-as.data.frame (swiss)
echant<-sample (1:dim(don) [1], size=10)
don[echant, ]
```

12-Tâches complexes et répétitives : replicate()

● Exercice 5 :

Faites 2000 corrélations linéaires entre Sepal.Length et Petal.Length sur des échantillons de 10 plantes. Dessinez l'histogramme des coefficients **r**.

13-Les conditions : ifelse()

● `ifelse(condition, "si vrai", "si faux")`

```
attach(iris)
ifelse( mean(Sepal.Length)>3, paste("moyenne: ",
mean(Sepal.Length)), "Moyenne <= 3")
detach(iris)
```

● Exercice 6 :

Faire un tirage aléatoire de 20 plantes dans le fichier iris. Selon la valeur de la moyenne de Sepal.Length, affichez le message approprié

- "moyenne > 5.8"
- "moyenne <= 5.8"

14-Les conditions : if else

- une fonction de base des langages de programmation

if (cond) expr1 else expr2

● Par exemple :

On a un scalaire x

si x est positif en calculer la racine

si x est négatif, calculer la racine de la valeur absolue

Afficher un commentaire à chaque fois

```
x<- -3
{if (is.numeric(x) & x>=0) cat("Racine de",x,"=",sqrt(x),"\n")
else cat("x négatif. Racine de |",x,"| : ", sqrt(-x), "\n")}
```

● Que se passe-t-il si x=0 ?

15-Les conditions : if else

● Exercice 7 :

Générez un vecteur de 30 valeurs aléatoires suivant une loi uniforme (`fonction runif()`). Les valeurs seront comprises entre 1 et 100.

Si les valeurs générées suivent une loi normale (`test de shapiro`) calculez et affichez la moyenne. Sinon, affichez la médiane.

16-Les boucles : for()

- Comment réaliser des boucles. Une autre fonction de base des langages de programmation

```
attach(iris)
for (i in 1:dim(iris)[1])
{
  if (Petal.Length[i] > 5) cat(as.character(Species[i]), "\n")
}
detach(iris)
```

- **Exercice 8 :** Le vecteur X est dans le mauvais sens. Je souhaite remettre la semaine à l'endroit.

```
X<-c("Dimanche", "Samedi", "Vendredi", "Jeudi", "Mercredi",
     "Mardi", "Lundi")
```

17-Quitter une boucle :

- On utilisera l'instruction `break()` pour quitter une boucle

```
for (i in 1:10000)
{
  cat(i^2, " ")
  if (i^2>100) break()
}
```

- Pour générer un message d'erreur en quittant la boucle, on utilisera la fonction `stop()`

```
options(show.error.messages=F)
for (i in 1:10000)
{
  cat(i^2, " ")
  if (i^2>100) stop("i est trop grand")
}
geterrmessage() #affiche le dernier message d'erreur
options(show.error.messages=T)
```

18-Les boucles : while()

- Comment tourner en rond jusqu'à nouvel ordre

while(cond) expr

● exemple :

obtenir un jeu de 20 données dont la corrélation entre **Sepal.Length** et **Petal.Length** est comprise entre 0.70 et 0.72

```
while(TRUE) # TRUE est toujours vrai
{
x<-sample(iris$Sepal.Length , size=20)
y<-sample(iris$Petal.Length , size=20)
if (cor(x,y) >= 0.70 & cor(x,y) <= 0.72)
{
    don<-data.frame(x,y)
    break()
}
}
cor(don)
```

19-Les boucles : while()

● Exercice 9 :

Créer un jeu simple. L'ordinateur génère un entier X de 1 à 100. A vous de le trouver.

Vous proposez un nombre et R vous indique si celui-ci est plus grand ou plus petit que X.

Continuez jusqu'à ce que votre nombre = X

remarque : vous aurez besoin de la fonction `readline()`

20-Les boucles : repeat()

- Dans les boucles `while(condition)`, rien n'est exécuté si la condition est fausse.
- Dans les boucles `repeat()`, une première exécution est effectuée avant de tester la condition.

```
j<-1
repeat
{
  j<-j+1
  if(j> 10) stop("too many iterations j")
}
```


21-Les fonctions :

● créer une fonction :

exemple 1 : calculer la norme d'un vecteur

```
norm <- function(x) sqrt(x%*%x)
```

```
norm(1:4)
```

exemple 2 : calcul de l'écart-type de l'échantillon

```
sd.ech<-function(x)
{
  sd<-sum( (mean(x)-x) ^2)
  sd<-sd/ (length(x)-1)
  sd<-sd^0.5
  return(sd)
}
```

22-Les fonctions :

● créer une fonction :

exemple 3 : calcul de l'écart-type de la population

```
sd.pop<-function(x)
{
  sd<-sum( (mean(x)-x) ^2)
  sd<-sd/ (length(x) )
  sd<-sd^0.5
  return(sd)
}
```

● Enregistrer des fonctions perso.

enregistrer les 3 fonctions à la suite dans un fichier texte. On peut ajouter des commentaires. Sauvegarder ce fichier dans un répertoire accessible. Appelez-le : **mesfonctions.R**

23-Les fonctions :

- Charger vos fonctions perso. 3 méthodes pour charger le fichier **mesfonctions.R**

1) avec le menu « File / Source R code »

2) insérer cette instruction dans votre script

```
source("c:/temp/mesfonctions.R")
```

3) avec un peu d'interactivité dans votre programme

```
source(file.choose())
```

24-Les fonctions : exercices

● Exercice 10 :

- construisez une fonction avec le jeu de devinette de nombre.
Appelez cette fonction `jeux()`
- testez cette fonction
- Si tout marche bien, sauvegardez cette fonction dans votre bibliothèque personnelle, à la suite des autres.

25-Les fonctions : le débogage

- En cas de fonction récalcitrante, on peut contrôler les valeurs des variables à un point précis de l'exécution :

```
calcul<-function(a=3)
{
  d<-a^(1/4)
  return(d)
}
calcul(-2)
[1] NaN
```

- Utilisation de la fonction `browser()`

```
calcul<-function(a=3)
{
  d<-a^(1/4)
  browser()
  return(d)
}
calcul(-2)
Browse[1]> get("a")
[1] -2
```

26-Les fonctions : exercices

● Exercice 11 :

- 1) Ajoutez un contrôle sur un nombre d'essais maximal. Celui-ci doit être passé en paramètre avec une valeur par défaut de 6. Quand le nombre maxi d'essais est atteint, la partie est perdue. Affichez un message.
- 2) Testez votre fonction pour différentes valeurs du nombre d'essais maxi.
- 3) Si tout va bien, enregistrez votre fonction dans [mesfonctions.R](#) , à la suite des autres.
- 4) Quittez puis relancez R
- 5) Charger la fonction de manière interactive avec `file.choose()`

27-Les fonctions : porté des variables

- La fonction **norm()** calcule la norme d'un vecteur. Elle utilise une variable "interne" **x**. Une variable x existe déjà...

```
x<-10  
norm <- function(x) sqrt(x%*%x)  
norm(1:5)  
x  
[1] 10
```

- Les variables "déclarées" à l'intérieur d'une fonction ont une porté locale

28-Les fonctions : valeurs renvoyées par la fonction

- La fonction `ecrit()` doit renvoyer 3 valeurs

```
ecrit<-function(x) { x ; x^2 ; x^3 }
```

- Mais `ecrit(2)` ne renvoie qu'une seule valeur (la dernière)

```
ecrit(2)  
[1] 8
```

- Pour renvoyer plusieurs valeurs, on utilisera les listes

```
ecrit<-function(x) list(simple=x, carre=x^2, cube=x^3)  
ecrit(2)$carre  
[1] 4
```


29-un peu d'interaction avec le système de fichier :

- choisir un fichier texte de manière interactive :

```
nom<-file.choose()
```

- des informations sur ce fichier

```
file.info (nom)
```

- afficher le contenu

```
file.show (nom)
```

- détruire un fichier

```
file.remove()
```

- essayez ces fonctions

```
basename(nom)
```

```
dirname(nom)
```

```
file.path(nom)
```

- contenu d'un répertoire

```
dir(dirname(nom), pattern = ".txt", recursive = TRUE)
```

Attention pattern=".txt" est différent de pattern=".TXT" !

30-Un peu d'interaction avec le système de fichier :

● Exercice 12 :

Allez visiter le répertoire R qui contient les bibliothèques de fonctions. Il se trouve vers "[C:\Program Files\R\rw2001\library](#)".

Récupérez la liste des noms de répertoires (ça correspond à la liste des bibliothèque)

31-Une instruction utile : substitute()

- Une fonction est un conteneur. Par exemple, le vecteur "normale" contient 100 valeurs suivant une loi normale

```
normale<-rnorm(100,10,1)
```

- On peut calculer la norme de ce vecteur avec une fonction

```
norm <- function(x) sqrt(x%*%x)
```

```
norm(normale)
```

Les calculs sont effectués sur le contenu du vecteur "normale"

- Si j'améliore ma fonction pour afficher un commentaire

```
norm<-function(x){ y<-sqrt(x%*%x); cat("La norme de",x,"est",y) }
```

Le résultat n'est pas celui attendu, puisque je souhaite afficher le nom du vecteur et pas son contenu.

32-Une instruction utile : substitute()

```
norm<-function(x) { y<-sqrt(x%*%x); cat("La norme de",x,  
"est",y) }
```

- Dans cette fonction, la variable x fait référence au vecteur "normale" qui contient 100 nombres de moyenne 10 et écart-type 1
- Si on veut faire référence au nom du vecteur et pas à son contenu, on utilisera la fonction substitute()

● La fonction deviendra :

```
norm<-function(x) {  
  y<-sqrt(x%*%x)  
  cat("La norme du vecteur '", substitute(x), "'  
  est", y, "\n")  
}
```

```
norm(normale)
```

```
La norme du vecteur ' normale ' est 101.5202
```

33-Une instruction utile : `substitute()`

● Exercice 13 :

Créer une fonction dessinant un graphe xy et sa droite de régression.
Donnez un titre et des labels d'axe explicites.

● Exercice 14 :

Faites le même exercice en donnant un choix interactif de couleur pour la droite de régression.

34-Des boucles et encore des boucles

● On peut utiliser les boucles itératives pour des calculs complexes. Par exemple, sur le data.frame **Orange**, on peut calculer le gain moyen journalier de la circonférence des troncs pour chaque arbre.

```
data(Orange)
arbres<-as.integer(names(summary(Orange$Tree)))
arbres<-arbres[sort.list(arbres)]
d<-length(arbres)
result<-data.frame(arbre=c(1:d), gmj=c(1:d))
for (i in arbres)
{
  mini<-min(Orange[Orange$Tree==i, 3])
  maxi<-max(Orange[Orange$Tree==i, 3])
  diff<-(maxi-mini)/7
  result[i,1]<-i
  result[i,2]<-diff
}
```

35-Sauvegarder plusieurs graphiques

- La fonction `savePlot()` permet d'enregistrer sur disque le graphique "en cours"

- **Exercice 15** : Utilisez le data.frame `iris`. En utilisant la fonction `for()`, dessinez les boîtes à pattes des variables quantitatives (un seul graphe à la fois) et enregistrez ces graphiques dans votre répertoire de travail au format `jpeg`.

36-Vous avez un message !

- Afficher les résultats dans une boîte à message

```
library(tcltk)
norm <- function(x)
{
  n<-round(sqrt(x%*%x),2)
  texte<-tkmessageBox(type='ok', icon="info",
    message=paste("norme",n))
}
norm(1:10)
```

- Types de boîtes : ok, yesno
- Types d'icônes : info, warning

● Exercice 16 :

Créez une fonction qui efface un fichier. Cette fonction doit vous demander confirmation.

37-Ecrire les résultats dans un fichier texte

● Par défaut, les résultats des analyses sont écrits dans la fenêtre "R console". On peut les réorienter vers un fichier texte. On utilisera la fonction `sink()`

```
setwd("c:/temp")
sink("toto.txt", append=TRUE)
data(PlantGrowth)
cat("Données PlantGrowth\n")
cat("Analyse de variance effectuée le",
    format(Sys.time(), "%a %d %b %X %Y %Z"), "\n")
x<-aov(weight~group , data=PlantGrowth)
summary(x)
sink()
```

● **Exercice 17** : Dans l'exercice 2, vous avez calculé les coefficients de corrélation de `pearson` entre l'age et la circonférence des orangers pour chaque arbre du data.frame `Orange`.

Refaites ces calculs de manière à ce que les résultats soient copiés dans le fichier `correlation.txt`. Commentez ces résultats.

38-Le batch : la non-interactivité totale !

- Il est possible d'automatiser des tâches répétitives. Dans ce cas, le passage par l'interface graphique de R devient inutile. On travaillera en mode *traitement par lot* (batch)

```
setwd("c:/temp")
sink("anova.txt", append=TRUE)
data(PlantGrowth)
cat("Données PlantGrowth\n")
cat("Analyse de variance effectuée le",
    format(Sys.time(), "%a %d %b %X %Y %Z"), "\n")
x<-aov(weight~group , data=PlantGrowth)
summary(x)
sink()
```

- On copie les instructions ci-dessus dans un fichier texte *rprg.txt*

- On lance l'exécution de ce programme grâce à la commande :

"C:\Program Files\R\rw2001\bin\R.exe" CMD BATCH "c:/temp/rprg.txt"

- Cette ligne de commande est exécutée dans une *console dos*

39-Le batch : la non-interactivité totale !

- Pour ne pas avoir à retaper une ligne d'instruction compliquée (avec des risques d'erreur), on peut copier cette ligne de commande dans un fichier "batch".

- La ligne de commande :

"C:\Program Files\R\rw2001\bin\R.exe" CMD BATCH "c:/temp/rprg.r"

peut-être copiée dans le fichier `rprg.bat`

- Il suffit alors de cliquer (ou double-cliquer) sur ce fichier pour lancer l'exécution du programme R

- Les résultats seront lisibles dans le fichier `c:\temp\anova.txt`

- **Exercice 18** : Automatisez intégralement l'exercice 17

40-Le système d'exploitation

● On peut lancer des instructions vers le système d'exploitation :

- par exemple, pour exécuter un programme `batch` :

```
shell.exec("c:/temp/rprg.bat")
```

- ou ouvrir une page html locale

```
system(paste('"c:/Program Files/Internet Explorer/IEXPLORE.EXE"',  
'C:/Program Files/R/rw2001/doc/html/rwin.html'), wait = FALSE)
```

- ou une page html distante

```
system(paste('"c:/Program Files/Internet Explorer/IEXPLORE.EXE"',  
'-url cran.r-project.org'), wait = FALSE)
```

- ou lancer un éditeur de texte

```
system("notepad c:/temp/prog.R")
```

41-Le système d'exploitation

- On peut utiliser les associations de fichiers du système d'exploitation

- Ouvrir un fichier html avec le butineur par défaut

```
shell.exec('C:/Program Files/R/rw2001/doc/html/rwin.html')
```

- Ouvrir un fichier pdf avec l'acrobat reader

```
shell.exec('C:/temp/lea_book.pdf')
```

- Ouvrir un fichier MS-Excel (avec Open Office si c'est l'application par défaut)

```
shell.exec('C:/temp/bledur.xls')
```

42-Développer une interface graphique

- Tcl (Tool Command Language) est un langage interprété, disponible *gratuitement* et qui fonctionne sous de très nombreux systèmes d'exploitation

- Un exemple de fenêtre de saisie :

```
library(tcltk)
tt<-tktoplevel()
tktitle(tt)<-"Tcl/Tk"
lab<-tklabel(tt, text="Entrez votre nom")
unnom<-tclVar("Anonyme")
nom<-tkentry(tt, width="20", textvariable=unnom)
b1<-tkbutton(tt, text="Quitter", command= function()
{
  tkdestroy(tt)
  tkmessageBox(icon="info", type="ok", message=
    paste(tclvalue(unnom), ": joli nom !"))
})
tkpack(lab, nom, b1)
```

43-Quelques adresses utiles :

- Des exemples de l'utilisation de Tcl/Tk :

<http://bioinf.wehi.edu.au/~wettenhall/RTclTkExamples/>

- Le site officiel de R

<http://lib.stat.cmu.edu/R/CRAN/>

- Introduction à Tcl/Tk

<http://www-lipn.univ-paris13.fr/~hamon/SJ-TclTk/index.html>

- Une présentation de Tcl/Tk

<http://www.uppp.free.fr/Tcltk.htm>

44-Correction des exercices :

● Exercise 1 :

```
data(Orange)
descript<-"The Orange data frame has 35 rows and 3 columns of
records of the growth of orange trees."

valeurs.maxi<-aggregate(Orange$circumference ,
list(valmaxi=Orange$Tree), max)

Orange.liste<-list(Orange, descript, valeurs.maxi)
names(Orange.liste)<-c("Orange", "descript", "valeurs.maxi")
```


● Exercise 2 :

```
attach(Orange)
by(Orange[,c(2,3)], Tree, cor, method = "pearson")
detach(Orange)
```

ou

```
attach(Orange)
liste<-list(age, circumference)
names(liste)<-c("age", "circumference")
by(liste, Tree, cor, method = "pearson")
detach(Orange)
```

● Exercice 3 :

```
attach(iris)
liste<-list(Sepal.Length, Sepal.Width, Petal.Length,
Petal.Width)
names(liste)<-c("Sepal.Length", "Sepal.Width", "Petal.Length",
"Petal.Width")
cr<-as.data.frame(lapply(liste, scale, center=T, scale=T))
mean(cr) ; sd(cr)
detach(iris)
```

● Exercice 4 :

```
hist(replicate(1000, sd(sample(scale(iris$Sepal.Length),  
                                size=40))))
```

● Exercice 5 :

```
attach(iris)
hist(replicate(2000,
  {
    echant<-sample(1:dim(iris)[1], size=10)
    tirage<-iris[echant,]
    cor(tirage$Sepal.Length, tirage$Petal.Length )
  }
), main="2000 corrélations", xlab="")
detach(iris)
```

● Exercice 6 :

```
attach(iris)
text1<-"moyenne > 5.8"
text2<-"moyenne <= 5.8"
ifelse(mean(sample(Sepal.Length , size=20))>5.8,text1, text2)
detach(iris)
```

● Exercice 7 :

```
x<-runif(30,0,100)
shapiro.test(x)$p
if (shapiro.test(x)$p >= 0.05)
cat("moyenne x=",mean(x),"\\n") else
cat("mediane x=", median(x),"\\n")
```

● Exercice 8 :

```
X<-c("Dimanche", "Samedi", "Vendredi", "Jeudi", "Mercredi",  
"Mardi", "Lundi")
```

```
Y<-X
```

```
for (i in 1:length(X)) {Y[length(X)-(i-1)]<-X[i]}
```

```
Y
```

Remarque : on peut aussi, plus simplement utiliser la commande suivante :

```
Y<-X[7:1]
```

● Exercice 9 :

```
X<-round( runif(1,1,100))
X<-as.integer(X)
while(TRUE)
{
nb<-readline(prompt = "Proposez un nombre de 1 à 100 : ")
nb<-as.integer(nb)
if (nb==X)
{
cat("Vous avez gagné, la solution est : ",X, "\n")
break()
}
if (nb>X) cat("Essayez encore : nombre est trop grand \n")
if (nb<X) cat("Essayez encore : nombre est trop petit \n")
}
```


● Exercice 10 :

```
jeu<-function()  
{  
  X<-round( runif(1,1,100))  
  X<-as.integer(X)  
  while(TRUE)  
  {  
    nb<-readline(prompt ="Proposez un nombre de 1 à 100 : ")  
    nb<-as.integer(nb)  
    if (nb==X)  
    {  
      cat("Vous avez gagné, la solution est : ",X, "\n")  
      break()  
    }  
    if (nb>X) cat("Essayez encore : nombre trop grand \n")  
    if (nb<X) cat("Essayez encore : nombre trop petit \n")  
  }  
}
```

● Exercice 11 :

```
jeu2<-function(maxi=6)
{
  X<-round( runif(1,1,100))
  X<-as.integer(X)
  nbessai<-0
  while(TRUE)
  {
    nbessai<-nbessai+1
    nb<-readline(prompt = "Proposez un nombre de 1 à 100 : ")
    nb<-as.integer(nb)
    if (nb==X) {
      cat("Vous avez gagné, la solution est : ",X, "\n")
      break()
    }
    if (nbessai<maxi) {
      if (nb>X) cat("Essayez encore : nombre trop grand \n")
      if (nb<X) cat("Essayez encore : nombre trop petit \n")
    }
    if (nbessai>=maxi) {
      cat("Dommage, vous n'avez droit qu'à",maxi,"essais\n")
      break()
    }
  }
}
```

● Exercice 12 :

```
nom<-file.choose()  
liste<-dir(dirname(nom), recursive = F, full.names = F)
```

● Exercice 13 :

```
graphxy<-function(x,y,donnees)
{
  attach(donnees)
  plot(x,y,main=paste("relation entre", substitute(x), "et",
substitute(y)), xlab=substitute(x),ylab=substitute(y))
  abline(lm(y~x))
  detach(donnees)
}
```

● Exercice 14 :

```
graphxy<-function(x,y,donnees)
{
  attach(donnees)
  plot(x,y,main=paste("relation entre", substitute(x), "et",
    substitute(y)), xlab=substitute(x),ylab=substitute(y))
  message<-"Couleur du trait 1 bleue, 2 vert, 3 noir :"
  couleur<-readline(prompt=message)
  couleur<-as.integer(couleur)
  if (couleur==1) coul<-"blue"
  if (couleur==2) coul<-"green"
  if (couleur==3) coul<-"black"
  abline(lm(y~x),col=coul)
  detach(donnees)
}
```

● Exercise 15 :

```
data(iris)
for (i in 1:4)
{
  boxplot(iris[,i], xlab="", main=names(iris)[i])
  savePlot(filename=paste("c:/temp/boxplot",i,sep=""),
    type="jpeg")
}
```

● Exercice 16 :

```
detruire<-function()  
{  
  fichier<-file.choose("Choisissez un fichier à effacer")  
  texte<-tkmessageBox(type='yesno', icon="warning",  
    message=paste("détruire",fichier,"?"))  
  if (as.character(texte)=="yes")  
  {  
    file.remove(fichier)  
    cat("fichier",fichier,"détruit.\n")  
  }  
}  
detruire()
```

● Exercice 17 :

```
attach(Orange)
setwd("c:/temp")
sink("correlation.txt", append=TRUE)
  cat("Données Orange, Formation R \n")
  cat(format(Sys.time()), "%a %d %b %X %Y %Z"), "\n")
  by(Orange[,c(2,3)], Tree, cor, method = "pearson")
sink()
detach(Orange)
```


● Exercice 18 :

- Copier les lignes suivantes dans le fichier `correlation.txt`

```
attach(Orange)
setwd("c:/temp")
sink("correlation.txt", append=TRUE)
  cat("Données Orange, Formation R \n")
  cat(format(Sys.time()), "%a %d %b %X %Y %Z"), "\n")
  by(Orange[,c(2,3)], Tree, cor, method = "pearson")
sink()
detach(Orange)
```

- Copier la ligne suivante dans le fichier `correlation.bat`

```
"C:\Program Files\R\rw2001\bin\R.exe" CMD BATCH
"c:/temp/correlation.txt"
```

- Dans l'explorateur de fichier, cliquez sur le fichier `correlation.bat`. Le programme s'exécute sans ouvrir de fenêtre R

Table des matières

1-Programmer avec R.....	3
2-Quelques éléments à connaître sur les listes (1).....	4
3-Quelques éléments à connaître sur les listes (2).....	5
4-Quelques éléments à connaître sur les listes (3).....	6
5-Quelques éléments à connaître sur les listes (4).....	7
6-Tâches complexes et répétitives, fonction by().....	8
7-Tâches complexes et répétitives, fonction by()	9
8-Tâches complexes et répétitives : lapply().....	10
9-Tâches complexes et répétitives : replicate().....	11
10-Echantillonner sur plusieurs variables.....	12
11-Echantillonner sur plusieurs variables.....	13
12-Tâches complexes et répétitives : replicate().....	14
13-Les conditions : ifelse().....	15
14-Les conditions : if else.....	16
15-Les conditions : if else.....	17
16-Les boucles : for()	18
17-Quitter une boucle :.....	19
18-Les boucles : while().....	20
19-Les boucles : while().....	21
20-Les boucles : repeat().....	22
21-Les fonctions :.....	23
22-Les fonctions :.....	24
23-Les fonctions :.....	25

24-Les fonctions : exercices.....	26
25-Les fonctions : le débogage.....	27
26-Les fonctions : exercices.....	28
27-Les fonctions : porté des variables.....	29
28-Les fonctions : valeurs renvoyées par la fonction.....	30
29-un peu d'interaction avec le système de fichier :.....	31
30-Un peu d'interaction avec le système de fichier :.....	32
31-Une instruction utile : substitute().....	33
32-Une instruction utile : substitute().....	34
33-Une instruction utile : substitute().....	35
34-Des boucles et encore des boucles.....	36
35-Sauvegarder plusieurs graphiques.....	37
36-Vous avez un message !.....	38
37-Ecrire les résultats dans un fichier texte.....	39
38-Le batch : la non-interactivité totale !.....	40
39-Le batch : la non-interactivité totale !.....	41
40-Le système d'exploitation.....	42
41-Le système d'exploitation.....	43
42-Développer une interface graphique.....	44
43-Quelques adresses utiles :.....	45
44-Correction des exercices :.....	46