PREMIERE PARTIE. PREMIERS CONTACTS AVEC R

R est un environnement intégré d'un ensemble de softwares (logiciels) pour la manipulation de données, le traitement statistique de celles-ci, l'affichage graphique …et bien plus encore. R existe en version Windows et Linux (32/64) ainsi que Mac OS. Nous n'envisageons que la version Windows. Il y a 3 façons au moins de travailler avec R

1. En mode « console ». Ouvrir une console (= « invite de commande »). Passer dans le répertoire de R (en principe C:\Programmes\R\bin) puis taper simplement R.
2. En mode Graphique. Clicker sur l'icône crée par l'installateur de R.
3. En utilisant l'interface Tinn-R. Une fois Tinn-R lancée, une icône ou bien à partir du menu R>Start/close connection>Rgui.

Remarquez bien que les modes 2 et 3 provoquent tous deux l'ouverture d'une fenêtre « R console ». La seule différence étant la présence de la fenêtre tinn-R dans le dernier cas. En mode 2 vous devrez entrer successivement vos commandes dans la console, alors qu'en mode 3 vous pourrez préparer directement un « script » dans la fenêtre de Tinn-R puis au moyen d'un seul click envoyer vers R l'ensemble du script qui s'exécutera alors. Tinn-R a bien d'autres fonctionnalités à découvrir…
Pour quitter R utiliser le menu de la console ou bien entrer la commande **q( )** [les parenthèses indiquent que la commande q est en fait un appel de fonction].
On supposera désormais que vous êtes en mode 2 ou 3. La première des choses est de spécifier le répertoire de travail courant (sauvegarde des scripts, fichiers de données et résultats). Utiliser le menu fichier de R et choisissez votre répertoire.

Obtenir de l'aide. Menu <Aide> de la console R: FAQ, Manuels et une aide en html très pratique avec une recherche automatique dans l'ensemble des packages de R du mot souhaité (essayez « **rnorm** »). Autre façon : dans la console de R entrez la commande : « **help(rnorm)** » ou encore « **?rnorm** ». Essayer aussi « **?help** ». Si vous souhaitez avoir des informations sur un sujet, il suffit d'entrer quelque chose comme « **help.search("linear models")» .** Des exemples sont aussi disponibles. On peut y accéder avec une commande du type example(thème). Essayer : « **example(lm)** ».

R est un *interpréteur* d'expressions simples. Techniquement il reconnaît 3 types d'expressions : Les **commandes** (= fonctions) à exécuter, les **assignations** (sauvegarde d'un résultat de commande dans une variable), les **commentaires, des instructions de contrôle** (if, while, for, repeat…). Il faut y ajouter l'affichage du contenu de variable qui s'obtient en entrant simplement le nom de la variable.

A connaître : search( ) -> montre le contenu du « search path ». Cette fonction est très utile pour connaître les listes, les data frames et les packages qui sont actuellement attachés/détachés (attach ( )). Pour connaître le contenu d'un package il convient alors d'utiliser la fonction ls, avec le numéro de position donné par search(). Par exemple : ls (3). Vous pourrez constater que le package « base » contient 1186 éléments !

EX1

```
# Génère vecteur de nombre pseudo-aléatoires = c'est un commentaire
> rnorm(50, 5, 2.1)          # affiche les 50 nombres mais le résultat est perdu !
> x <- rnorm(50,5, 2.1)      # On sauvegarde le résultat dans une variable x
> x                          # et on affiche son contenu…
> y <- rnorm(50,-3,0.8)
> y
> plot(x, y)                  # un simple scatter plot = diagramme de dispersion
> x11(w=4, h=4)               # défini une fenêtre graphique de 4 X 4
> bringToTop(stay = TRUE);    # en haut de l'écran
> plot(x, y)                  # A nouveau le graphique…
> rm(x,y)                     # détruit des données  (vecteurs) x et y
> graphics.off()
```

Les structures de données

Dans le monde R on parle d'**Objets**. Ces objets sont très divers (vecteurs, graphiques, données, fonctions, etc.). Pour ce qui concerne les structures de données, il faut bien avoir à l'esprit qu'elles se différentient par leur mode : null (=vide), logical, numeric, complex, character. Et leur classe. Les classes plus utilisées sont au nombre de 3 : **vectors**, **matrix** et **data.frame**. Mais il existe aussi les array (tableaux), factor, time-series et les list. *A noter que list et data.frame peuvent être hétérogènes*.

Vecteurs (mode quelconques)

```
> a=c(5,5.6,1,4,-5)     # Création de l'objet a recevant un vecteur numérique
                        # de dimension 5 et de cordonnées (5,5.6,1,4,-5).
> a[1]                  # Affichage de la première coordonnée du vecteur a
> b = a[2:4]            #Création du vecteur numérique b de dimension 3
                        # et de coordonnées (5.6, 1, 4)
> d=a[c(1,3,5)]         # Création du vecteur numérique d de dimension 3
                        # et de coordonnées (5, 1,-5)
> e=3/d                 # Création du vecteur numérique e de dimension 3
                        # et de coordonnées (3/5, 3,-3/5)
length(e)               # dimension de e
t(e)                     # transposition de e -> vecteur ligne
t(e)%*% b               # produit scalaire entre un vecteur ligne(e) et colonne(b)
rep(c(1,2,3), each = 2) # fonctions utiles
seq(1.575, 5.125, by=0.05)
order(d)
```

Matrices

```
Syntaxe de création : matrix(vec, nrow=n, ncol=p, byrow=(T ou F))

> a    = 1:20 ; a
> b    = sample(1:10,10) ; b
> mat1 = matrix(a, nrow=5, byrow=F) # Création de la matrice numérique 5 × 4 ayant
> mat1                               # pour première ligne 1, 6, 11, 16
> mat2 = matrix(a, nrow=5, byrow =T)# Création de la matrice numérique 5 × 4 ayant
> mat2                               # pour première ligne 1, 2, 3, 4
> mat3 = t(mat2)                     # Transposition de mat2
> mat3
> mat4 = matrix(b, ncol=2)
> mat4
> b = mat3 %*% mat2 ; b             # produit matriciel : %*%
> c = mat2 %*% mat3 ; c              # le produit matriciel n'est pas commutatif !
> dim(mat1)                         # dimension de mat1 : lignes X colonnes
> mat1[3,2]                         # affichage de l'élément en i=3, j=2 de mat1
> mat5 <- mat1[c(3,4),]             # sélection des lignes 3 et 4 et stockage dans
                                    # une nouvelle matrice mat5
> mat5 <- mat1[, -c(1,3)]      # surpressions des col. 1 et 3 et stockage dans mat5
> apply(mat1, 2, sum)          # appliquer la somme à mat1 par colonnes (2)
> apply(mat1, 1, sum)          # puis par ligne (1)
> rbind(mat1,mat2)             # concaténation (bind) par ligne des deux matrices,
> cbind(mat1,mat2)             # puis par colonnes

# Encore plus fort, toujours plus de dimensions…
```

On crée un tableau **array**(**vec**,**c**(**n**,**p**,**q**...)) où **vec** est le vecteur contenant les éléments de la matrice qui seront rangés en colonne et l'argument **c**(**n**,**p**,**q**...) désigne les dimensions : **n** est le nombre de lignes, **p** le nombre de colonnes, **q** le nombre de matrices.

```
> x=array(1:30,c(2,3,5))
> x[1,2,2]        # affichage de l'élément de coordonnées (1,2,2)
> dim(x)
```

Data frame

Fondamentalement c'est une matrice à colonnes pouvant être hétérogène. Il s'agit en fait d'une classe particulière de listes utilisées pour stocker des données. Chaque colonne correspond à une variable (avec éventuellement un nom), alors que chaque ligne correspond à un individu. Ça se présente ainsi (Exemple fictif d'une analyse de sol) :

Prélèvement	pH	%MatOrga	C/N	Aluminium meq/100g
1	5.5	12	60	4.10
2	4.5	25	75	7.32
…	…	…	…	…
n	6.1	15	32	2.6

Syntaxe usuelle de création: data.frame(nom1= var1, nom2 = var2, ...)

```
as.data.frame(mat)                      # formation à partir d'une matrice
MesData <- read.table("path du fichier",sep = "\t", header = TRUE) #déjà vu…
> v1=sample(1:12,30,rep=T)              #Echantillonnage avec remise dans les entiers
                                        #de 1 à 12
> v2=sample(LETTERS[1:10],30,rep=T)     #30 lettres entre A et J
> v3=runif(30)                          #30 réalisations d'une loi uniforme sur [0,1]
> v4=rnorm(30)                          #30 réalisations d'une loi normale(0,1)
> xx = data.frame(v1,v2,v3,v4)
> xx
> xx$v2                                 # pour accéder à une colonne
> summary(xx)                           # pratique, non ?
> plot(xx)                              # surprise !

NB. Des données sont livrées avec R, à partir desquelles sont bâtis les exemples du
logiciel.
> data()                                #la liste des datas disponibles
> ?women                                # description des données
> data(women)                           # on charge women
> attach(women); women                  # women est un data.frame
```

Conseil : Il ya dans le manuel "An Introduction to R" (dans le menu <Aide>) un exemple de session (Appendix A, p. 76) sous R. L'appliquer intégralement.

DEUXIEME PARTIE. DES DONNEES : QUE FAIRE ?

1. On a mesuré sur 43 enfants diabétiques la concentration en peptide C que l'on souhaite étudier en fonction de l'âge (diabete.txt). Proposer un protocole d'analyse statistique entre la Concentration et l'Age, puis établissez un modèle de type : [C] = f(age). (*Le peptide C fait partie du précurseur de l'insuline. A maturité, on obtient : insuline + peptide C. Mesurer le peptide C revient à estimer le potentiel du pancréas à secréter de l'insuline, même en présence d'injection d'insuline exogène, ou bien en présence d'anticorps anti-insuline qui gênent son dosage*). Conseil : pour une bonne visualisation des données brutes essayez les fonctions stripchart(data frame) et hist(data frame$variable).

2. On considère aussi le facteur Sexe. Quelle analyse envisager ?

3. L'influence d'un traitement grossissant, à base de vitamines, est étudiée sur des animaux de races différentes. Pour cela nous disposons d'animaux de trois races i = 1; 2; 3, et nous avons effectué trois traitements j = 1; 2; 3, utilisant respectivement 5, 10 et 15 g de vitamines B12 par cm3. Le gain moyen de poids par jour est mesuré, à l'issue d'un traitement de 50 jours dans chaque cas (vitamines.txt). Un seul animal est utilisé pour chaque couple « race-traitement ». De quel type de plan d'expérience s'agit-il ? Quelle analyse envisager et pourquoi ? Pour tester les normalités des résidus : shapiro.test (residus). Egalité des variances : barlett.test (residuals~races, data = vitamines) ou bien barlett.test (residuals~doses, data = vitamines). Terminer par un test HSD de Tuckey.