T.P. 1 d'Analyse des Données

Lors de nos T.P., nous utiliserons le logiciel R grâce à l'interface RStudio, qui est disponible sur toutes les machines. Vous pouvez télécharger R gratuitement à l'adresse `www.r-project.org` *et RStudio à l'adresse* `www.rstudio.com`*. Attention, pour utiliser RStudio, R doit être installé sur la machine.*

Le but du T.P. 1 est d'apprendre à se familiariser avec le logiciel R et de présenter un certain nombre de commandes de base.

1 Une calculatrice

R est un logiciel de calcul numérique orienté vers l'analyse des données en statistique. Il s'agit avant tout d'une « grosse calculatrice ».

Ainsi, R permet de faire les opérations de *calcul* élémentaire. Essayer les commandes suivantes :

```
> 3+5
> 2*4
> 3-8
> 2/3
> 2.1-6*2.5
> 3*2-5*(2-4)/6.02
> 2^2
```

R permet aussi de faire des calculs plus élaborés, à l'aide de plusieurs fonctions prédéfinies. Que font les fonctions suivantes ?

```
> sqrt(4)
> abs(-4)
> log(4)
> sin(0)
> exp(1)
```

R connaît la valeur de quelques constantes mathématiques :

```
> pi
> cos(2*pi)
```

R manipule des opérateurs et valeurs *logiques* :

`<` : inférieur
`>` : supérieur
`<=` : inférieur ou égal
`>=` : supérieur ou égal
`==` : égal
`!=` : différent
`!` : non
`&` ou `&&` : et
`|` ou `||` : ou
`xor(,)` : ou exclusif

Essayer les commandes :

```
> 1<2
> 1==3
> TRUE
> F
```

Pour définir une *chaîne de caractères*, on emploie des guillemets.

```
> 'a'
> 'essai'
```

Pour une *affectation*, on utilise les commandes `<-` ou `=`.

Remarque 1.
- Si une ligne contient plusieurs instructions, elles doivent être séparées par « ; ».
- R distingue minuscules et majuscules.
- Les lignes de commande commençant par « # » ne sont pas exécutées par R.
- `ls()` affiche la liste des objets créés et installés dans l'environnement courant.

```
> # Un joli calcul
> a=52
> b <- 50
> A=0
> A-b ; a-b
```

2 L'aide

Lorsque vous recherchez une fonction, que vous en avez oublié le nom ou la syntaxe, l'aide du logiciel vous apportera sûrement toutes les informations souhaitées. S'il ne vous est pas demandé de retenir par coeur toutes les commandes possibles et imaginables de R, il est en revanche *très important de savoir utiliser l'aide* pour les retrouver. Celle-ci doit vous permettre d'acquérir une réelle autonomie. On peut y accéder de différentes manières.

- `help.start()` lance l'aide en ligne.
- L'instruction `?obj` (ou `help(obj)`) affiche la documentation associée à l'objet `obj`.
- `apropos('mot')` affiche la liste des fonctions et objets qui contiennent le mot-clé `mot` dans leur nom, tandis que `??mot` affiche cette liste dans l'aide en ligne.

Essayer les commandes :

```
> ?plot
> help(plot)
> help.start()
> apropos('plot')
> ??plot
```

3 Vecteurs, matrices, tableaux

L'objet de base en R est le *vecteur*. Un vecteur peut contenir des valeurs numériques, mais aussi des valeurs logiques (`TRUE` ou `FALSE`, `T` ou `F`) ou encore des chaînes de caractères...

La commande permettant de créer un vecteur en énumérant ses valeurs est `c`, ce qui signifie *concaténation*. Plusieurs vecteurs peuvent d'ailleurs être concaténés à l'aide de cette même commande.

```
> v1=c(1,2,3,4,5)
> v1
> v2= c(3.14, 2.71, 1.414, 1.732)
> v2
> v3= c(2,4, 3.14)
> v3
> v4=c(v1,v2,v3)
> v4
> length(v1); length(v2); length(v3); length(v4)
```

```
> v5=c(T, T, F, F)
> v5
> v6=c('a', 'b')
> v6
```

On peut aussi fabriquer une suite à l'aide de `seq` en donnant les valeurs minimale et maximale et le pas ou la longueur. Si le pas est 1, il suffit même d'écrire `min : max`. La fonction `rep` donne un vecteur dans lequel un nombre ou une suite de nombres est répété autant de fois que désiré. Enfin, la fonction `scan('')`, qui permet de saisir directement le contenu d'un vecteur, peut s'avérer très pratique.

```
> seq(1,10,2)
> seq(1,10,by=2)
> seq(1,9,length.out=5)
> seq(1,10,by=.1)
> 1:10
> rep(55,10)
> rep(c(1,2), 10)
> notes=scan('')
1: 5 12 14 3 6 10 12 6 5 11 10 9 2 6.5 9 17 8 9.5 10
> notes
```

La fonction `vector(mode=,length=)` crée un vecteur avec des entrées de type et longueur donnés.

```
> vector('numeric',6)
> vector('logical',5)
> vector('character',3)
```

La fonction `factor` donne à un vecteur le statut de variable qualitative ou facteur. On peut aussi construire un facteur en spécifiant le nombre de modalités et le nombre d'occurrences de la même modalité à l'aide de `gl`.

```
> factor(c('brun','blond','roux','blond','roux','brun','brun'))
> factor(c(1,2,3,2,3,1,1),labels=c('brun','blond','roux'))
```

On accède aux éléments d'un vecteur en mettant entre crochets l'indice ou un vecteur d'indices :

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

On peut donner des noms aux éléments d'un vecteur :

```
> v=c(1,2,3,4)
> names(v)=c('alpha','beta','gamma','delta')
> v['beta']
> a='alpha'; b='beta'; c='gamma'; d='delta'
> v[b]
```

La fonction `matrix`, prenant en argument un vecteur, le nombre de colonnes et le nombre de lignes, permet d'obtenir une *matrice* (tableau à deux dimensions). En mettant les indices entre crochets, on peut extraire un élément, une colonne, une ligne ou une sous-matrice. Il est possible de donner des noms aux lignes et colonnes de la matrice avec `colnames` et `rownames`. Les fonctions `cbind` et `rbind` permettent de réunir deux matrices.

```
> mat=matrix(1:20, 4, 5)
> mat1=matrix(1:20, 4, 5, byrow=TRUE)
> id=diag(5)
> mat[1,5]
> mat[1,]
> mat[,5]
> mat[1:2,1:3]
> colnames(mat)=c('C1', 'C2', 'C3', 'C4', 'C5')
> rownames(mat)=c('L1', 'L2', 'L3', 'L4')
> cbind(mat,mat1)
> rbind(mat,mat1)
```

La longueur d'un vecteur est donnée par `length` et la dimension d'une matrice par `dim`. La fonction `which` retourne le vecteur des indices pour lesquels la condition logique donnée en argument est vraie. L'option `arr.ind=T` permet de traiter le cas des matrices.

```
> length(v)
> dim(mat)
> which(notes==10)
```

D'autres objets de R :

- `list` : liste, ensemble de vecteurs.
- `array` : vecteur avec dimensions ; il peut y en avoir une (vecteur), deux (matrice) ou plus.
- `data.frame` : tableau de données, matrice dont chaque colonne, qui possède un nom, correspond à une variable, les lignes correspondant aux individus.

```
> a=array(1:20,dim=c(4,5))
> a[2,4]
>
> b=list(alpha=1:3, beta=c('a','b','c','d'))
> names(b)
> b$alpha
> b$beta
> b[1]
> b[2]
>
> c=data.frame(a=gl(2,5,10), b=1:10, x=seq(1,20,2))
> c
> c$a ; c[,1]
> c$b ; c[,2]
> c$x ; c[,3]
> names(c)
> rownames(c)
```

On peut faire afficher le type d'un objet et convertir un objet en un objet de type différent.

```
> class(mat)
> is.data.frame(mat)
> as.vector(mat)
```

Exercice 1.

1. Construire un vecteur allant de 2 à 100 par pas de 2.
2. Construire la matrice de taille 10×3 dont la première colonne ne contient que des 1, la deuxième que des 2 et la troisième que des 3.
3. Donner les noms I, II et III aux colonnes et A,B,C,D,E,F,G,H,I,J aux lignes.
4. Transformer cette matrice en dataframe.
5. Demander à R d'afficher la dernière colonne du dataframe, et la première ligne.

4 Quelques paramètres graphiques

La fonction principale permettant de tracer un *graphe* est `plot`. Elle possède de nombreuses options concernant le type de tracé, la couleur, les axes, le titre du graphique, ou encore la légende.

`type` : points, ligne, les deux, bâtons, escalier...
`lwd` : épaisseur du trait
`lty` : type de trait
`cex` : taille des points
`pch` : forme des points
`col` : couleur
`main` : titre du graphique
`xlab, ylab` : titres des axes
`asp` : échelle y/x
`xlim, ylim` : limites des axes

Les fonctions `lines` et `points` sont utilisées pour ajouter des lignes ou des points à un graphe existant. On peut ajouter une légende à l'aide de `legend`, fonction possédant elle aussi plusieurs options. Plus généralement, du texte peut être ajouté sur le graphique grâce à `text`. Lorsqu'il s'agit d'une expression mathématique, il est préférable utiliser la fonction `expression` plutôt qu'une chaîne de caractère. `x11()` ouvre une nouvelle fenêtre graphique, tandis que `dev.cur()` donne le numéro de la fenêtre courante et `dev.list()` la liste des fenêtres ouvertes. `dev.off()` ferme la fenêtre courante ou celle dont le numéro est donné en argument et `graphics.off()` les ferme toutes. On peut diviser la fenêtre graphique grâce à `par(mfrow=)`.

```
> x=1:10
> y=(1:10)^2
> a=cbind(x,y)
> par(mfrow=c(2,2))
> plot(a,cex=1.5)
> plot(a,type='l')
> plot(a,type='b',lwd=2,pch=2)
> plot(a,type='l',lty='dotted',col=2,main='Titre du graphe',xlab='Abscisses',
  ylab='Ordonnées')
> lines(x,x,col=3)
> points(x,y+2*rnorm(10),col=4,pch=16)
> legend('topleft',expression(y==x^2,y==x,'Données'),inset=0.05,
  lty=c('dotted','solid',NA),pch=c(NA,NA,16),col=c(2,3,4 ))
```

Exercice 2.

1. Construire un graphique représentant la fonction racine carré.
2. Tracer la courbe en rouge
3. Donner un titre au graphique, ainsi qu'aux axes
4. Superposer sur la courbe des points
5. Mettre une légende
6. Dans la même fenêtre, faire la même avec la fonction log.

5 Fonctions

Remarque 2. Pour afficher le code d'une fonction de R, il suffit d'en entrer le nom dans la console.

On peut définir ses propres fonctions, avec la syntaxe suivante :

```
mafonction <- function(x) {mes instructions}.
```

En écrivant une fonction, on peut être amené à utiliser une boucle. La syntaxe d'une boucle « for » (pour i variant de 1 à n, faire telle action) est par exemple

```
for (i in indices) {action}.
```

Il peut être utile aussi de manipuler une condition « if », ce qui se fait avec la syntaxe

```
if (condition) {action 1} else {action 2}.
```

Lorsque l'on écrit une fonction, il est préférable de la rédiger dans un éditeur de texte prévu à cet effet plutôt que dans la console R. L'éditeur installé sur les machines du SCRIPT s'appelle « Tinn-R ». Un fichier contenant un ensemble de fonctions et commandes R constitue un programme, qui peut être enregistré au format « .R ».

Exercice 3. Construire une fonction qui à un vecteur associe la moyenne de ses éléments négatifs ou nuls.

6 Génération de nombres aléatoires

R peut générer des séquences de nombres aléatoires indépendants, suivant certaines distributions courantes comme la loi normale, binomiale, ou du chi-deux. La commande générique est `rnomdeladistribution(paramètres)`, par exemple `rnorm(50)` pour générer 50 nombres tirés suivant la loi normale.

Essayer les commandes suivantes

```
> rnorm(10);rnorm(10,1,0.1)
> rbinom(1,20,0.5);rbinom(10,20,0.5)
```

Il est aussi possible d'avoir accès à d'autres fonctions de la distribution souhaitée, comme la densité ou les quantiles, la commande est alors du type `dnomdeladistribution(paramètres)` pour la densité et `qnomdeladistribution(paramètres)` pour les quantiles. De manière plus générale, taper `?Nomdeladistribution` donnera la liste des fonctions disponibles associées. Par exemple

```
>?Normal;?Binomial
```

Exercice 4 (facultatif).

1. Générer une matrice de taille 100×10 dont les éléments sont indépendants et tirés suivant la loi normale.
2. Construire le vecteur de taille 100 correspondant au maximum par ligne de la matrice précédente (on pourra utiliser `apply`), et calculer la moyenne de ce vecteur (on pourra utiliser `mean`).
3. Répéter les points 1 et 2 pour des matrices normales de taille $100 \times k$ et stocker le résultat dans un vecteur $m(k)$, pour k variant entre 1 et 50 (on pourra construire une liste de matrice, construire une fonction, utiliser la fonction `lapply` et construire le vecteur à partir de la liste obtenue en utilisant `unlist`).
4. Tracer la courbe de $exp(m^2/2)$ et interpréter.