Université Joseph Fourier **Frédérique Leblanc**
L2/STA401 **leblanc@imag.fr**

TP1 : Prise en main de R

R est un logiciel libre développé par une très large communauté scientifique et basé sur "des objets". RStudio est une interface très pratique pour utiliser simplement les fonctionnalités de R. Vous pouvez le télécharger gratuitement depuis le site **www.rstudio.com**, en ayant au préalable installé le logiciel R lui même, disponible sur le site **www.r-project.org.** Attention, dans le langage R, on distingue majuscule et minuscule.

RStudio est séparé en 4 fenêtres graphiques :

- en bas à gauche : fenêtre de commandes où sont exécutées les instructions tapées après le prompt `>` avec la touche retour chariot du clavier (Entrée $\hookleftarrow$).
- en haut à gauche : où l'on écrit son script afin de pouvoir sauver ses instructions dans un fichier `nomfichier.R`.
- en bas à droite : où sont, entre autres, affichés les graphiques.
- en haut à droite : indique l'historique des instructions exécutées pendant la session, affiche les objets créés au fur et à mesure en précisant leur nature.

Vous trouverez sur la "toile" un grand nombre de cours ou "tutorials" et bien sur vous avez toujours à votre disposition l'aide en ligne de R que l'on pourra appeler avec la fonction `help(·)`.

Objectifs de la séance : *Commencer à utiliser le logiciel comme une calculatrice et savoir faire quelques manipulations élémentaires sur une série de données rangées dans un vecteur. Savoir tracer le graphe d'une fonction.*
.

1 Premiers pas avec R et l'environnement R Studio :

Avant toute chose créer dans son dossier personnel un répertoire STAT et un sous-répertoire TP dans lequel seront rangés tous les documents de Tps : fiches de Tps .pdf, scripts .R, graphiques .pdf, jeux de données analysés ....

Une fois R (R.base) et Rstudio installés sur la machine une icône R est disponible sur le bureau. Cliquer sur cette icône et l'environnement de R studio s'ouvre et se présente avec trois ou quatre fenêtres.

Exercice 1 : méthode de travail

1. Lancer R studio
2. Ouvrir un nouveau script en allant dans **File** > **New file** > **Rscript** : la quatrième fenêtre en haut à gauche s'ouvre.

3. Commencer par y inscrire en ligne 1 le commentaire : `#------------Exercice 1-----------` puis exécuter cette ligne de commande en cliquant sur le numéro de ligne du script indiqué à l'extrémité gauche de la ligne (la sélection de la ligne à exécuter est surlignée en bleue) et cliquer sur l'onglet **Run**. Observer la sortie retournée par R en bas à gauche. En fait comme la ligne est précédée du signe #, R voit la suite comme un commentaire et l'affiche.

4. Pour des raisons pratiques et d'économie de syntaxe la session R sera lancée directement dans le répertoire de travail TP précédemment créé dans le répertoire STAT. Pour cela aller dans le menu décrit dans le bandeau supérieur **Session** > **Set Working Directory** > **Choose directory** puis sélectionner le répertoire TP du répertoire STAT (ne pas oublier de sélectionner le répertoire choisi en cliquant sur ouvrir).Observer la syntaxe de la commande qui vient de s'exécuter en bas à gauche et la recopier sur la ligne 3 du script (tandis que la ligne 2 recevra le commentaire #`instruction pour travailler dans le répertoire de travail TP`).

5. Taper à présent la ligne de commande `1:10` et l'exécuter. Que retourne R ?

6. Nommer la séquence précédente init avec :
   `init<-1:10`
   ou `1:10->init` ou encore `init=1:10` . Observer ce qui s'affiche en haut à droite. Pour afficher le contenu de l'objet appelé `init` dans la fenêtre de commande en bas à gauche il suffit d'exécuter la commande :
   `init`

7. A tout moment pour comprendre ce que fait une fonction ou une commande il suffir d'appeler l'aide de R avec la fonction `help()`. Par exemple
   `help(":")`
   retourne les instructions pour utiliser l'opérateur `:`.

8. Sauvegarder votre script en activant la fenêtre où est affichée le script (il suffit de cliquer dans la fenêtre et le curseur clignote) puis aller dans **File** > **Save as** puis taper TP1 (inutile de rajouter l'extension .R qui sera automatiquement ajoutée puisqu'il s'agit d'un script ouvert avec Rstudio).

9. Fermer Rstudio qui propose alors une sauvergarde du Workspace qui n'est pas indispensable puisque sont conservées toutes les instructions dans le script précédemment sauvegardé.

2 Les vecteurs et les matrices

L'objet élémentaire est le vecteur constitué d'une suite de mots (de type alpha-numériques) d'éléments logiques (TRUE ou FALSE) ou de nombres (de type numériques). Une collection de vecteurs de mêmes types et de mêmes longueurs constitue une matrice.

Exercice 2 : créer, manipuler des vecteurs ou des matrices

1. Lancer R, changer de répertoire de travail (choisir TP de STAT) et ouvrir le script TP1.R

2. *Créer* : la suite de données $(1, 2, 3, 4, 5)$ en tapant la commande `c(1,2,3,4,5)` puis en l'exécutant avec Entrée $\hookleftarrow$:

   `c(.)` est la fonction de concaténation qui permet de créer une suite d'éléments (logiques, numériques ou alpha-numériques) énoncés les uns à la suite des autres et séparés par des virgules. L'objet ainsi produit est un vecteur colonne.

   Ici la réponse de R est [1] 1 2 3 4 5 qui indique que l'objet ainsi créé est constitué d'une colonne numérotée [1] et qui contient la suite des valeurs $(1, 2, 3, 4, 5)$ (par souci d'ergonomie ces valeurs sont listées sur une seule ligne).

3. *Affecter* : la suite précédemment créée dans un vecteur nommé `x` en exécutant la commande : `c(1,2,3,4,5)→ x`
(ou bien `x←c(1,2,3,4,5)`). R ne retourne rien sinon le traditionnel début de ligne > pour vous redonner la main. C'est normal et si vous voulez vérifier que le vecteur `x` existe bien quelque part et contient les informations saisies il vous suffit de l'appeler en exécutant seulement
`x`

4. *Extraire* : Exécuter
`x[2]` # pour obtenir l'élément d'indice 2 du vecteur x et
`x[c(2,4)]` # pour obtenir les éléments d'indices 2 et 4.
Que fait la commande
`x[-2]` ?

5. Fabriquer le vecteur `y` contenant la suite des valeurs $(2, 4, 6, 8, 10)$ et le vecteur `label` contenant la suite de caractères a,b,c,d,e avec `label<-c("a","b","c","d","e")`.

6. *Créer une matrice* avec 5 lignes et 2 colonnes contenant x et y en l'affectant à `A` avec :
`matrix(c(x,y),ncol=2,byrow=F)->A`
Consulter l'aide de la fonction `matrix()` et indiquer ce que fait l'option `nrow`. On peut aussi utiliser la commande `cbind`:
`cbind(x,y)`.

7. *Extraire ou compléter* : comme pour les vecteurs on peut extraire des éléments d'une matrice ou construire une matrice à partir de plusieurs autres matrices ou vecteurs. Que font les commandes :
`A[1,2]`
`A[,1]`
`A[2,]`
et commenter les deux commandes suivantes
`un<-rep(1,5)`
`matrix(c(x,y,un),ncol=3,byrow=F)->B`

3 Calculer avec R : manipulations de base

Il s'agit là d'expérimenter les calculs élémentaires que l'on doit savoir faire avec un ou plusieurs vecteurs numériques de même longueur ou une ou plusieurs matrices de mêmes dimensions.

Exercice 3 : Avec un seul vecteur

1. *Multiplier ou diviser par un scalaire* : Par exemple pour produire le résultat du calcul $\frac{1}{5}(1, 2, 3, 4, 5)$ (resp. du calcul $5 * (1, 2, 3, 4, 5)$) on exécutera :
`x/5` (resp. `x*5`)

2. *Ajouter un scalaire* : par exemple pour calculer $(1 + 2, 2 + 2, 3 + 2, 4 + 2, 5 + 2)$ on exécute :
`x+2`

3. *Sommer en cumulant ou pas* : on obtient la somme des éléments de x (resp. les sommes cumulées) avec la fonction `sum()` (resp.`cumsum()`) :
`sum(x)`
`cumsum(x)`

4. *Dimension* : on obtient la longueur de x avec la fonction `length`
`length(x)`

5. *Calculer la racine* : avec la fonction `sqrt()`
`sqrt(x)`

6. *Calculer une puissance* : avec le symbole ^, par exemple :
   `x ^ 3`

7. *Agrandir* (ajouter des éléments à la liste décrite par un vecteur) : si on veut produire un vecteur x' constitué de x complété par une sixième valeur 6 par exemple on utilise la fonction de concaténation `c()` :
   `c(x,6)`
   Si on veut compléter x par $(1, 1, 1, 1, 1)$ on peut utiliser l'une ou l'autre des commandes suivantes :
   `c(x,1,1,1,1,1)`
   `c(x,rep(1,5))`
   `c(x,c(1,1,1,1,1))`
   La fonction `rep()` permet de créer une suite d'éléments de même valeur (premier argument) et de longueur (deuxième argument) fixées. Essayer également de manipuler la fonction `seq()`, bien utile en pratique. Tester par exemple les commandes :
   `seq(1,10,2)`
   `seq(from=1,to=10,by=2)`
   `seq(from=1,to=2,by=.2)`

8. *Tester :* Si on souhaite savoir si les composantes de x vérifient une condition logique par exemple si elles sont positives on exécute :
   `x>0`
   Si on veut savoir pour quel indices les composantes de x sont positives on exécute :
   `which(x>0)`

9. Que fait la commande :
   `length(which(x>0))`

10. Que font les commandes :
    `which(x!=2)`
    `x>0&x<2`
    `sum(x>0&x<2)`

Exercice 4 : Avec plusieurs vecteurs

1. *Ajouter des vecteurs* : pour ajouter (ou soustraire) `x` et `y` (l'élément en position i du résultat est la somme des éléments en position i, de `x` et de `y`) on exécute : `x+y` ou `x-y`

2. *"Multiplier des vecteurs"* : pour multiplier (ou diviser) deux à deux les éléments des vecteurs `x` et `y` (attention il ne s'agit pas de la multiplication usuelle matricielle entre un vecteur ligne et un vecteur colonne ou entre un vecteur colonne et un vecteur ligne. Nous y reviendrons selon le besoin) :
   `x*y` ou `x/y`

3. *Réunir deux vecteurs* : dans un même vecteur colonne avec la fonction `c()` ou dans une matrice à cinq lignes et deux colonnes (une pour `x` et l'autre pour `y`) avec la fonction `cbind()` ou dans une matrice à 2 lignes et cinq colonnes avec `rbind()`. Tester les commandes:
   `cbind(x,y)`
   `rbind(x,y)`
   `t(cbind(x,y))`
   Que fait la fonction `t()` ?

Exercice 5 : calculs sur des matrices

1. *Somme :* essayer les commandes suivantes et les commenter
   `sum(A) #calcule la somme des éléments de A`
   `A+1 # ajoute 1 à chaque élément de A`
   `A+B`

2. *Produit :* Que font les commandes :
   `A*B`
   `A*A`

3. *Produit matriciel* : Pour faire le produit matriciel de A avec le vecteur ligne $(1, 2)$ on utilise l'opérateur %*%. Exécuter et commenter les commandes suivantes :
   `c(1,2)->a`
   `A%*%a`
   `t(a)%*%t(A)`

Exercice 6 : pour s'entrainer

1. Calculer la moyenne empirique et la variance empirique de x
2. Calculer la proportion des coordonnées de x inférieures ou égale à 2.
3. Calculer le nombre de fois où la ième coordonnée de x dépasse la ième coordonnée de y
4. Construire une matrice (2,2) symétrique ayant des 2 sur la diagonale et des 1 ailleurs. La nommer `C` et calculer le produit matriciel de C avec A.

4 Graphiques

La fonction graphique de base sous R permettant de tracer des nuages de points que l'on peut ou pas relier entre eux est la fonction `plot()`.

Exercice 7 : tracé d'une fonction.
Soit la fonction $f(x) = sin(x)$. On se propose ici de tracer son graphe.

1. Créer le vecteur des nombres allant de 0 à 7 par pas de 0.1 et le nommer `abs`
2. Créer le vecteurs des ordonnées obtenues en appliquant la fonction sinus aux abscisses listées dans `abs` et les affecter à `ord`
3. Tracer le graphe de f sur $[0, 7]$ en utilisant la fonction `plot()`. Il est fortement recommandé de consulter au préalable l'aide en ligne de cette fonction. Ajouter titre et légendes. Refaire ce graphique à l'aide de la fonction de R `curve()`.
4. Ajouter la droite d'équation $y = 1$ en rouge et celle d'équation $y = x$ en vert avec la fonction `abline()`.
5. Ajouter le point de coordonnées $(3, 1)$ sur le graphique précédent.