Université Paris I, Panthéon - Sorbonne

Deuxième Année Licence M.I.A.S.H.S. 2016 – 2017

TP de Méthodes Numériques n^0 1 : Initiation au logiciel R

L'objectif de ce TP est de vous familiariser avec les objets et les commandes du logiciels R, notamment de tout ce qui a trait aux manipulations des réels, vecteurs et matrices. Le logiciel R est un logiciel de mathématiques appliquées, plutôt spécialisé en statistiques mais pas seulement, qui a la particularité d'être libre (donc gratuit) et qui s'enrichit au fur et à mesure de l'apport gracieux de chercheurs du monde entier. Il est issu d'un langage de programmation S permettant la programmation orientée objet et des algorithmes, dont les racines sont essentiellement le FORTRAN (1958), le langage APL (1967) et le langage C (1972). Une version commercialisée, Splus, plus conviviale et unifiée mais payante et moins développée, a également existé. On peut télécharger le logiciel R sans risque à partir des sites `http://www.r-project.org/` ou `http://cran.cict.fr/` pour des systèmes d'exploitation Windows, Mac ou Linux.

On commence par se connecter à l'ordinateur, puis à ouvrir le logiciel en "cliquant" sur l'icône correspondante. Il faut savoir que R se développe rapidement par d'ajonction de nombreux "packages". Un certain nombre de packages (une trentaine) sont automatiquement installés avec une installation de base, mais il en existe un nombre près de 8000 actuellement que l'on peut télécharger à loisir. Ces packages contiennent des fonctions adaptées à certains traitements, nous verrons plus tard comment s'en servir.

Cela pourra être un réflexe utile dans toute la suite, on "clique" sur `Aide`; deux formes d'aide sont alors proposées. Une aide html est proposée, puis en cliquant sur `Search Engine & Keywords`, on peut rechercher des commandes à partir de mots-clés (essayer avec *variance* ou *inverse*). Ensuite, quand le nom d'une commande est connue, on peut avoir tous les détails la concernant en "cliquant" sur `R functions`, puis en écrivant le nom de cette commande (essayer avec `var` ou `inv`).

Dans la suite, sont écrites à gauche des commandes à taper, et à droite des commentaires sur ces commandes ou des questions relatives. Vous pouvez donc ainsi apprendre les commandes du logiciel par la pratique.

Une façon intéressante d'utiliser le logiciel est d'ouvrir des "Scripts" dans lesquels vous allez pouvoir écrire l'ensemble des commandes. Il sera alors possible d'exécuter une partie des commandes tapées. Ce sera donc très pratique plutôt que de retaper les commandes à chaque fois. Il faut donc aller dans la colonne `Fichier`, et cliquer sur nouveau `script`. Puis, après avoir taper les commandes choisies, avec la souris vous sélectionner la ou les commandes particulières que vous désirer faire tourner, vous allez ensuite dans la colonne `Edition` et vous cliquer sur `Executer une selection`. Les fichiers `script` peuvent enregistrer dans le répertoire que vous choisissez sous la forme d'un fichier `.R`.

L'objectif de ce TP est vous familiariser avec le langage R, sa structure, ses objets et ses fonctions, ainsi que d'avoir un aperçu de ses possibilités graphiques et des commandes Entréee/Sorties.

Commandes de type calculatrice pour nombres réels

$6 + 3.4$	Effectue l'opération demandée.
$x = -2.34/21$	Le résultat de l'opération est affecté à la variable x.
x	Permet d'afficher la variable x.
$Y = x * 3$	Création de Y.
Y	Affiche la valeur de Y.
$Y = 1 + Y\hat{}3$	Opération de mise à une certaine puissance; remarquer que Y a changé, sa première écriture est oubliée. **On utilisera très souvent cette réaffectation d'une variable**.
$ls(all = TRUE)$	Liste les objets utilisés dans la session.
$JOIE = x + Y$	Affiche le résultat de $x + Y$ affecté à la variable $JOIE$.
$ls(all = TRUE)$	Vérifier que $JOIE$ a été rajoutée. Noter que l'on peut appeler les variables comme on le veut En pratique, on utilisera pour noms des variables les abréviations de ce qu'elles désignent. Par exemple, le temps pourra être t, T ou $temps$.
$rm(Y)$	Suppression de la variable Y.
Y	Vérification...
$rm(list = ls(all = TRUE))$	Permet d'effacer toutes les variables de la session.
$x = 3.14235;$	Nouvelle variable x.
$y = cos(x)$	Utilisation de la fonction cosinus.
$z = exp(y)$	Utilisation de la fonction exponentielle. Essayer avec les autres fonctions usuelles (quel logarithme?).
$v = floor(x)$	Partie entière de x.
v	Résultat?
$v = floor(x, 2);\ v$	Approximation par défaut à 10^{-2} près. Notez que l'on a mis deux commandes sur la même ligne.
$w = round(x, 3);\ w$	Approximation la plus proche à 10^{-3} près. Et à 10^{-4} près? Conclusion.
$x = 12.34567890123456789;\ x$	On choisit un nombre avec beaucoup de décimales.
$x - 12.34567890123456$	Résultat? Quel est donc l'approximation faite? Faire d'autres essais.
$.Machine\$double.eps$	Donne la précision demandée (voir l'aide).
$options(digits = 20);\ x$	Permet d'afficher plus de décimales. Qu'en pensez-vous?

Nous verrons que ces approximations nécessaires (même avec un ordinateur plus puissant, il faudrait de toute manière s'arrêter à un moment dans la décomposition décimale d'un nombre) sont une des sources des limitations d'un logiciel de calcul numérique.
Important: tout ce qui se rapporte à l'infini, comme la décomposition décimale d'un nombre réel ou rationnel, mais aussi une limite, une dérivée, ..., ne peut pas être obtenu par un logiciel comme R. Un logiciel de mathématique formel comme Maple ou Matematica peut en revanche y avoir accès dans certaines limites.

Autre point qui peut être utile:

Si vous voulez rappeler une commande précédemment tapée, ici par exemple $ls(all = TRUE)$, il vous suffit d'aller dans la fenêtre de commandes et d'utiliser les flèches ↑ (revient aux commandes précédentes) et ↓ (affiche les commandes suivantes). Faire plusieurs essais pour maîtriser ces commandes très utiles (évite surtout de recopier).

Manipulations et opérations sur les vecteurs

$x = c(1.23, -4.3, log(2), pi)$	Crée un vecteur. Lequel?
$length(x)$	Taille du vecteur.
$y = c(c(-2, sqrt(3)), x)$	Crée un nouveau vecteur. Lequel?
$z = rep(y, 3)$	Nouveau vecteur utilisant une répétition de y.
$X = 1 : 100$	Quel est le vecteur X?
$Y = 2 + X\hat{}0.5 - 10 * cos(log(X))$	Qu'a-t-on fait ici?

Les deux dernières commandes tapées sont typiques de ce que l'on va faire avec le logiciel R. Le calcul direct sur les vecteurs va économiser beaucoup de temps de calcul dans de très nombreux cas de figure.

$Y[3]$	Permet d'obtenir une composante de Y.
$Y[c(3, 6 : 17, 99)]$	Permet d'obtenir des commposantes choisies de Y.
$S = sum(Y)$	Calcul de la somme des composantes de Y **Commande très importante...**
$m = mean(X)$	Calcul de la moyenne des composantes du vecteur X. Montrer que le résultat est bon!
$M = max(Y);\ M$	Détermine le maximum des composantes du vecteur Y.
$u = which(z == min(z));\ u$	Expliquer cette commande.
$z[u]$	Résultat.
$plot(2 * X, Y)$	Trace l'ensemble des points $(2 * X_i, Y_i)$
$title('Une\ courbe')$	Met un titre à la figure
$plot(Y)$	Trace le graphe de Y en prenant par défaut en absisse ... X.
$r = runif(10, 0, 1)$	Produit 10 nombres "indépendants" entre 0 et 1 suivant une loi uniforme (nous détaillerons beaucoup cela plus tard).
$s = runif(100000, -5, 5)$	Produit 100000 nombres "indépendants" entre 0 et 1 suivant une loi uniforme Que pensez-vous du temps de calcul?
$hist(s)$	Trace un histogramme des composantes de s.

Un problème peut être que lorsque l'on refermera la session, toutes les variables créées et les commandes tapées vont être effacées. Un moyen de les sauvegarder va être d'aller dans l'onglet `Fichier`, et de choisir `Sauver l'environnement de travail` et/ou `Sauver l'historique des commandes`. Noter cependant que cette dernière action n'est pas utile quand tout a été écrit et sauvé dans un script.

Exercice 1. On désire calculer des approximations de $I = \sum_{j=1}^{\infty} 1/j^2$. Rappeler pourquoi cette série converge. En minorant j^2 par $j(j-1)$, puis en utilisant une série télescopique, montrer que $1 < I < 2$. Calculer $I_n = \sum_{j=1}^{n} 1/j^2$ pour $n = 100$, $n = 1000$ et $n = 10^6$. Faire un tracer de I_n en fonction de n pour $n = 10^r$ avec $r = 2, 2.1, 2.2, \cdots, 5.9, 6$. Un calcul théorique obtenu à l'aide des séries de Fourier (fait en L3...) permet de montrer que $I = \pi^2/6$. Représenter graphiquement le reste de la série pour les valeurs de n précédentes. Comment peut-on majorer théoriquement l'erreur d'approximation de I par I_n (penser à des intégrales)? Quel est l'ordre de grandeur en n de cette approximation?

Programmes: boucles et conditionnement

On va maintenant explorer les principales commandes permettant de réaliser des programmes en R. Nous allons nous restreindre à des programmes effectuant des boucles avec de possibles conditionnements (nous verrons plus tard l'utilisation de sous-programmes et fonctions).

	Premier programme: termes d'une suite
$rm(list = ls(all = TRUE))$	On initialise. Important pour ne pas utiliser des variables déjà affectées.
$x = sqrt(2)/2;$	On initialise la variable du programme.
$for\ (n\ in\ c(1 : 100))$	On va répéter faire prendre à n les valeurs de 1 à 100.
{	Début des commandes.
$x[n + 1] = 4 * x[n] * (1 - x[n])$	On itère sur x la fonction logistique: on obtiendra donc une suite $u_{n+1} = 4u_n(1 - u_n)$.
}	On termine la boucle.
$n;\ x$	Pour vérifier.
	On recommence en ne s'arrétant que lorsque x est supérieur à 0.999
$rm(list = ls(all = TRUE))$	On initialise.
$x = sqrt(2)/2;$	On initialise la variable du programme.
$i = 1;$	On utilise un compteur i pour connaître le nombre de termes nécessaires.
$while\ (x < 0.999)$	Boucle conditionnée.
{	Début des commandes.
$x = 4 * x * (1 - x)$	La suite "déroule".
$i = i + 1$	Le compteur tourne.
}	On termine la boucle conditionnée. Noter le résultat. Recommencer en changeant le conditionnement, puis en changeant le premier terme.

Exercice 2. Reprendre le calcul des termes de la suite "logistique" dans un fichier script (par exemple *Suitelogistique.R*). Introduire le premier terme avec *input*, puis calculer les 10^5 premiers termes de la suite, mémorisés dans un vecteur X. Tracer les points $(X[n], X[n+1])$, pour $n = 1, 2 \cdots, 10^5$. Que constatez vous?

	Nouveau programme Syracuse.R avec test La suite de Syracuse
$rm(list = ls(all = TRUE))$	Nouveau programme.
$N = as.numeric(readline("EntierN?"))$	Demande d'un entier N que l'on écrit dans la fenêtre de commande. Le *as.numeric* permet de transformer en nombre la réponse
$i = 1$ # *Compteur*	Noter également que le caractère # permet de mettre un commentaire
while $(N > 1)$	Boucle conditionnée.
$\{if\ (N\%\%2 == 0)$	On conditionne suivant la parité de N. Noter que = devient == dans un test.
$\{N = N/2\}$	
else $\{N = 3 * N + 1\}$	Donc dans le cas où N est impair.
$i = i + 1\}$	Le compteur tourne.
i	Nombre d'itération nécessaire. Recommencer en changeant de N. Vérifier que l'algorithme sur N le ramène toujours à 1 (proposition non encore démontrée et peut-être indécidable!!!).

Pour améliorer le programme précédent, on peut créer une fonction Syracuse (dans un script) et taper les commandes suivantes:

```
## Fonction suite de Syracuse ##
#################################

Syracuse <- function(N)
{
M=N
i=1
while (M[i]>1)
{
if (M[i]%%2==0)
{M[i+1]=M[i]/2}
else {M[i+1]=3*M[i]+1}
i=i+1
}
return(M)
}
```

On lance la fonction en écrivant par exemple *Syracuse*(9347) dans la fenêtre de commande. Attention cependant à avoir bien sauvé ce script sous le nom `Syracuse.R` dans le répertoire courant qui a été choisi en cliquant sur l'onglet `Fichier`.

Exercice 3. Mettre en place une fonction permettant lorsqu'un entier quelconque N est donné, de vérifier s'il est ou non un nombre premier. On commencera par démontrer qu'il n'est pas utile de chercher à diviser N par un nombre entier plus grand que $\sqrt{N}$...

Exercice 4. Utiliser le programme précédent pour illustrer une autre proposition non encore démontrée: la fameuse conjecture de Goldbach qui affirme que tout nombre entier pair peut s'écrire comme la somme de 2 nombres premiers. Faire des essais pour tous les entiers jusqu'à $N = 10^5$.

Exercice 5. Définir une fonction appelée F telle que $F : \ x \in]0, 10] \mapsto 1/\sqrt{x} - \cos(x)$. En considérant des x entre 0 et 10 espacés d'un paramètre *pas* à régler (par exemple $pas = 0.01$), tracer la courbe représentative de F. Déterminer théoriquement le nombre de x_0 tels que $F(x_0) = 0$. Essayer de trouver un moyen pour déterminer les valeurs approchées de ces x_0 (on pourra par exemple utiliser la valeur absolue d'un réel, commande *abs* en R...). Peut-on améliorer l'approximation?