TP0 : introdution au langage R

1 Introduction

Le langage R a été développé au début des années 1990 par Ross Ihaka et Robert Gentleman. Il est actuellement maintenu et mis à jour par une équipe de développeurs au sein du ***R Project***. Cette structure garantit des mises à jour fréquentes avec une communauté d'utilisateurs qui s'occupe du développement de nouvelles fonctionnalités (librairies). Le langage R permet de manipuler des jeux de données et de visualiser les résultats facilement avec des graphiques paramétrables. Il est entièrement gratuit et multi-plateforme. Vous pourrez en effet l'utiliser sous Windows, Mac OS ou Linux.

2 Installation et lancement

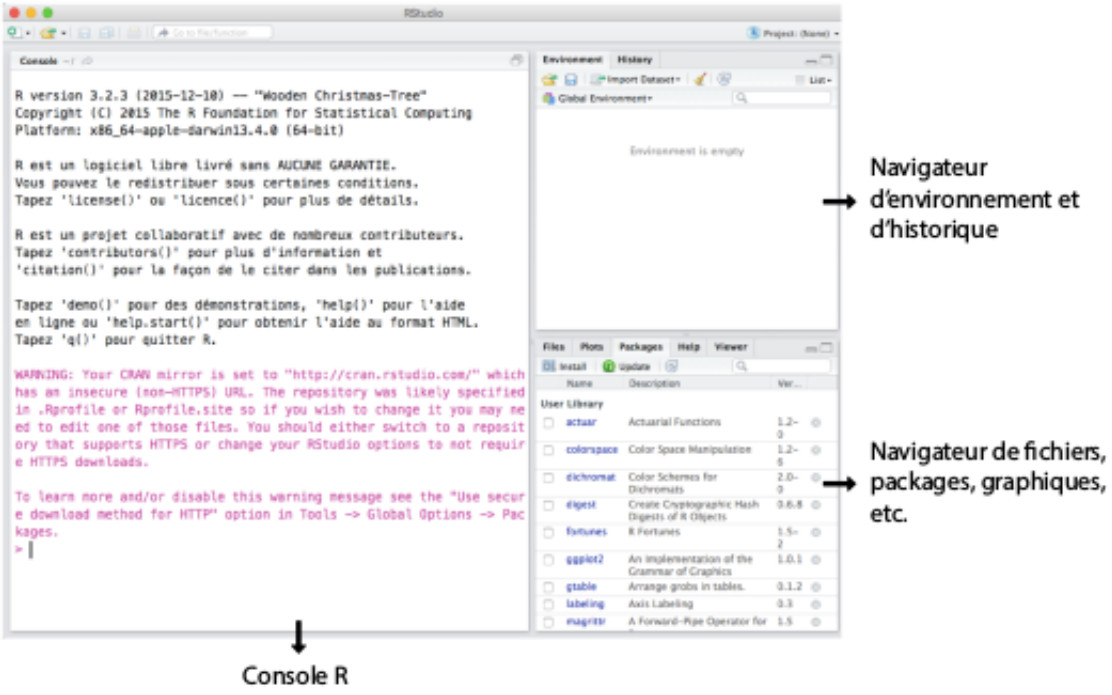
Voici un miroir de téléchargement disponible sur : http ://cran.irsn.fr/

- **Windows :** executable disponible en cliquant sur : "Download R for Windows".
- **Linux :** commande : sudo apt-get install r-base
- **Mac OS X :** programme disponible sur : "Download R for (Mac) OS X"

En salle TP, on travaille sous Linux. Pour lancer R, il faut ouvrir un terminal et taper la commande : R. Tapez et analyser le résultats des lignes suivantes :

```
> 2 + 3
> x <- exp(2)
> x
> pi
> cos(pi/4)
```

On va utiliser ***RStudio*** comme un environnement de développement intégré (integrated development environment, IDE). Pour l'installer sur vos ordinateurs personnels, vous devez le télécharger à partir du lien : `https://www.rstudio.com/products/rstudio/download/#download`. En salle TP, ***RStudio*** est déjà installé. Vous pouvez le lancer et créer un nouveau projet. Lors de la création d'un projet, ***RStudio*** crée dans le dossier visé un fichier avec une extension .Rproj. Ajouter ensuite un script R et recopier le code précédent ensuite exécuter le ligne par ligne.


RStudio
Console
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.
R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.
Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.
WARNING: Your CRAN mirror is set to "http://cran.rstudio.com/" which has an insecure (non-HTTPS) URL. The repository was likely specified in .Rprofile or Rprofile.site so if you wish to change it you may need to edit one of those files. You should either switch to a repository that supports HTTPS or change your RStudio options to not require HTTPS downloads.
To learn more and/or disable this warning message see the "Use secure download method for HTTP" option in Tools -> Global Options -> Packages.
Environment
History
Import Dataset
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
User Library
actuar
Actuarial Functions
colorspace
Color Space Manipulation
dichromat
Color Schemes for Dichromats
digest
Create Cryptographic Hash Digests of R Objects
fortunes
R Fortunes
ggplot2
An implementation of the Grammar of Graphics
gtable
Arrange grobs in tables.
labeling
Axis Labeling
magrittr
Navigateur d'environnement et d'historique
Navigateur de fichiers, packages, graphiques, etc.
Console R

3 bases du langage R

Les rubriques d'aide des diverses fonctions de R contiennent une foule d'informations ainsi que des exemples d'utilisation. Leur consultation est tout à fait essentielle. Pour consulter la rubrique d'aide de la fonction cos :

```
> ?cos
ou
> help(cos)
```

Toute commande R est soit une expression, soit une affectation. Une expression est immédiatement évaluée et le résultat est affiché à l'écran :

```
> 2 + 3
[1] 5
```

Lors d'une affectation, une expression est évaluée, mais le résultat est stocké dans un objet (variable) et rien n'est affiché à l'écran.

```
> a <- 5
> a
[1] 5
> b <- a
> b
[1] 5
```

Tout dans le langage R est un objet. Les objets possèdent au minimum un mode et une longueur et certains peuvent être dotés d'un ou plusieurs attributs. Testez les commandes suivantes :

```
> v <- c(1, 2, 5, 9)
> mode(v)
> length(v)
```

Mode	Contenu de l'objet
numeric	qui peut etre soit entier (integer) ou double (double)
complex	nombres complexes
logical	booléen (TRUE ou FALSE)
character	chaînes de caractères
function	fonction
list	données quelconques
expression	expressions non évaluées

Voici un exemple d'utilisation des entrées/sorties standards

```
> scan()
> a<-scan()
> b<-scan()
> c<-a+b
> print(c)
> c
```

3.1 Structures de contrôle

— if (condition) {expressions1} else {expressions2}
— for (variable in suite) {expressions}
— while (condition) expression

3.2 fonctions

```
> carre <- function(x) { return (x*x) }
> carre(3)
```

4 les vecteurs

4.1 création et accès

En R, tout est un vecteur. Contrairement à certains autres langages de programmation, il n'y a pas de notion de scalaire en R ; un scalaire est simplement un vecteur de longueur 1. Voici quelques exemples pour créer des vecteurs. Testez les et afficher le contenu du vecteur à chaque fois .

```
> v1 <- vector("numeric", 10)
> v2 <- vector("logical", 8)
> v3 <- c (1,3,4,8)
> v4 <- rep(1,10)
> v5 <- seq(1,10)
> v6 <- (1:10)
> v7 <- seq(1,10,3)
```

Pour plus de lisibilité, on peut être amené à vouloir donner des noms aux éléments d'un vecteur. Voici un exemple de vecteur appelé poids dans lequel nous noterons le poids de 4 individus : Marc, Julie, Sophie et Blaise

```
> poids <- c(77, 58, 66, 82)
> poids
> names(poids)
> names(poids) <- c("Marc", "Sophie", "Julie", "Blaise")
> poids
```

Chaque élément d'un vecteur est accessible grâce à un index numérique ou par un nom qui peut lui être attribué.

```
> poids [1]
> poids["Sophie"]
```

Il est possible d'extraire plus d'un élément d'un vecteur à la fois. Il suffit pour cela de spécifier plusieurs index sous la forme d'un vecteur. Le vecteur passé comme index peut contenir des booléens.

```
> poids[2:3]
> poids[c(1,4)]
> poids[c(TRUE,FALSE,TRUE,FALSE)]
```

Exercice : créer et afficher le vecteur suivant puis sélectionner les éléments supérieurs à 7 :

```
> v <- c(1:12, 8:5, rep(2,4), 8:15, 16:12)
```

4.2 opérations sur les vecteurs

Analyser et commentez chaque ligne

```
> poids <- c(85,78,54,98,66,78,77,72,99,102,54,66,98,75,82,83,75)
> poids
> poids <- poids + 1
> poids
> sqrt(poids)
> length(poids)
> poids <- poids[2:length(poids)]
> poids
> length(poids)
> sort (poids)
> sort(poids, decreasing=TRUE)
> order(poids)
> poids[order(poids)]
```

La fonction ***summary*** renvoie une série d'informations sur la distribution numérique : la valeur minimale, le premier quartile (25%), la médiane (ou second quartile, 50%), la moyenne, le troisième quartile (75%) et la valeur maximale.

```
> summary(order(poids))
```

4.3 comparer plusieurs vecteurs

Télécharger le fichier de test.RData puis tester le script suivant. qui importe l'objet contenant les données et affiche le nom des objets qui ont été importés

```
> objets <- load("test.RData")
> objets
```

- La fonctions ***load*** permet de charger 4 vecteurs
  - *tailleG* : vecteur contenant la taille des garçons.
  - *performanceG* : vecteur contenant la meilleure de 3 performances en saut en hauteur pour les garçons.
  - *tailleF* : vecteur contenant la taille en centimètres des filles.
  - *performanceF* : vecteur contenant la meilleure de 3 performances (en centimètres) en saut en hauteur pour les filles.
- Afficher la taille de chaque vecteur
- Afficher les noms associés aux différentes mesures.
- Quel est l'ensemble des noms présents dans l'étude (utilisez la fonction union)
- Quels noms sont communs aux deux ensembles (utilisez la fonction intersect)
- Créer de nouveaux vecteurs (tailleG2, performanceG2, tailleF2 ,performanceF2) contenant seulement les personnes pour lesquels nous avons toutes les données nécessaires (taille + performance).
- Créer des vecteurs triés à partir de (tailleG2, performanceG2, tailleF2, performanceF2) sans modifier ces derniers vecteurs (utiliser la fonction order)

5 les facteurs

les facteurs sont généralement utilisés pour y stocker des variables **qualitatives**.

```
> vent <- factor(c("fort","faible","moyen","faible","faible","fort"))
> vent
[1] fort   faible moyen  moyen  faible faible fort
Levels: faible fort moyen
```

Notez la présence de l'attribut "Levels" qui est un vecteur ordonné des valeurs uniques présentes dans le facteur.

6 les matrices

Pour créer des matrices. Copier et analyser les commandes suivantes :

```
> m1 <- matrix(1:6, nrow = 2, ncol = 3)
> m1
> m2 <-matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)
> m2
> m3 <- matrix(c(40, 80, 45, 21, 55, 32),nrow = 2, ncol = 3)
> m3
```

Pour récupérer un élément, toute une ligne ou toute une colonne :

```
> m3[1, 2]
> m3[2, ]
> m3[,3]
```

7 les listes

La liste est le mode de stockage le plus général et polyvalent du langage R. Il s'agit d'un type de vecteur spécial dont les éléments peuvent être de n'importe quel mode. La liste x ci-dessous contient un vecteur numérique, une chaîne de caractère et un booléen

```
> x <- list(size = c(1, 5, 2), user = "Joe", new = TRUE)
> x
```

Il est recommandé de nommer les éléments d'une liste. En effet, les listes contiennent souvent des données de types différents et il peut s'avérer difficile d'identifier les éléments s'ils ne sont pas nommés.

La liste demeure un vecteur. On peut donc l'indicer avec l'opérateur [].Cependant, cela retourne une liste contenant le ou les éléments indicés. C'est rarement ce que l'on souhaite. Pour indicer un élément d'une liste et n'obtenir que cet élément, et non une liste contenant l'élément, il faut utiliser l'opérateur d'indiçage [[ ]]. Comparer

```
> x[1]
$size
[1] 1 5 2
> x[[1]]
[1] 1 5 2
```

Une autre façon d'indicer un seul élément d'une liste est par son étiquette avec l'opérateur $

```
> x$size
[1] 1 5 2
```

L'utilité d'une liste est de regrouper dans un même objet une série d'objets appartenant par exemple à une même expérience ou observation. Le code suivant permet de créer une liste d'athlètes avec différentes mesures de performances

```
athletes <- list(Didier=c(630, 625, 628, 599, 635, 633, 622),
 Jules=c(610, 590, 595, 582, 601, 603),
 Pierre=c(644, 638, 639, 627, 642, 633, 639),
 Matthieu=c(622, 625, 633, 641, 610),
 Georges=c(561, 572, 555, 569, 653, 549, 558, 561),
 Khaled=c(611, 621, 619, 618, 623, 614, 623),
 Guillaume=c(599, 601, 612, 609, 607, 608, 594),
 Hermann=c(624, 630, 631, 629, 634, 618, 622),
 Carlos=c(528, 531, 519, 533, 521), Keith=c(513))
```

Extraire pour chaque athlète, en utilisant la fonction lapply(liste, fonction, arguments de la fonction) les informations suivantes :

— la meilleure et la pire performance,

— le nombre de mesures effectuées,

— la performance moyenne.

Remplacez lapply() par sapply(). Que constatez vous ?

8 Data frames

Un data frame est une liste dont tous les éléments sont de la même longueur (ou comptent le même nombre de lignes si les éléments sont des

matrices). Il est généralement représenté sous la forme d'un tableau à deux dimensions. Chaque élément de la liste sous-jacente correspond à une colonne.

On peut créer un dataFrame avec la fonction ***data.frame()*** ou bien à partir d'un fichier grâce à la fonction ***read.csv()***.

```
> resultats <- data.frame(taille=c(185,178,165,171,172),
 poids=c(82,81,55,65,68),
 QI=c(110,108,125,99,124),
 sexe=c("M","M","F","F","F"),
 row.names=c("Paul","Matthieu",
 "Camille","Mireille","Capucine"))
> resultats
```

Extraire les informations suivantes de data frame resultats

1. la taille de Camille.
2. le QI et le sexe des trois premiers individus.
3. toutes les données relatives à Paul et Capucine.
4. le vecteur des poids

Télécharger le fichier "day.csv" à partir du lien `https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset`. Il s'agit d'une base de données sur le nombre de locations quotidiennes entre 2011 et 2012 d'un système de vélos partagés. Analyser le résultat des commandes suivantes :

```
> dataBikes <- read.csv("day.csv", header=TRUE);
> head(dataBikes);
> summary(dataBikes);
```

9 Graphiques

Il existe plusieurs commandes permettant de tracer différents type de graphiques. Voici quelques exemples :

— La commande **plot** : L'utilisation de base est plot(x, y), où x et y sont 2 vecteurs de même longueur. On construit alors un nuage de points dont le i-ème point est de coordonnées (x[i], y[i]). Il existe des options dans plot permettant de changer les paramètres graphiques.

— La commande **curve** : sert à confectionner rapidement certaines courbes représentatives de fonctions. On peut notamment l'utiliser pour représenter la densité et la fonction de répartition des lois d'une variable à densité. Exemple : $curve(sin(x), -3, 10)$

— La commande **parplot** : Pour un vecteur x à n éléments, les commandes barplot(x) donnent n barres verticales, la i-ème barre étant de hauteur proportionnelle à x[i].

Exercice : analyser les commandes suivantes :

```
>  x <- rnorm(18)
>  y <- rnorm(18)
>  plot(x, y, xlab = "", ylab = "",
 xlim = c(-2, 2), ylim = c(-2, 2), pch = 22,
>  col = "red", bg = "yellow", bty = "l",
tcl = 0.4, main = "Nuage de points")
>  grid(lwd = 1)
```