Prise en main de R

Maud Thomas

Table des matières

1 Prise en main de R **1**
1.1 Interfaces . . . 1
1.1.1 Installation . . . 1
1.2 Ma première session . . . 3
1.2.1 R comme calculatrice . . . 3
1.2.2 L'aide dans R . . . 5
1.2.3 Scripts . . . 5
1.2.4 Répertoire de travail . . . 6
1.2.5 Créer des objets R . . . 6
1.2.6 Supprimer des objets . . . 8
1.2.7 Quitter RStudio . . . 8

1 Prise en main de R

Le logiciel R est un logiciel d'étude statistique. Ce logiciel sert à manipuler des données, à tracer des graphiques et à faire des analyses statistiques sur ces données. Il s'agit d'un logiciel *opensource* et multi-plates-formes qui fonctionne sous UNIX (et Linux), Windows et Macintosh. Il est distribué gratuitement et vous pouvez le télécharger et installer sur une machine personnelle.

Cette initiation à R s'appuie sur la fiche de Tabea Rebafka, les livres *Le logiciel R* de Lafaye de Micheaux et al., et *Statistiques avec R* de Cornillon *et al.* qui fournissent des introductions plus complètes au logiciel R.

1.1 Interfaces

L'interface graphique RStudio permet une utilisatoin plus commode et plus jolie que l'interface standard de R.

On peut également utiliser un notebook qui est un document interactif qui contient des paragraphes de texte ainsi que des bouts de code, que l'on peut modifier et exécuter. Les notebooks se prêtent bien pour remplacer les feuilles de TP en papier ou comme un appui pour une réunion. En revanche, pour développer du code il est préférable d'utiliser l'interface standard de R ou RStudio.

1.1.1 Installation

Pour installer R et les autres programmes sur votre ordinateur personnel voici les démarches (qu'il vaut mieux exécuter dans l'ordre). Notez que tout est gratuit.

1. **Installer** `R` à partir de la plateforme du CRAN à l'adresse
   http://cran.r-project.org
   Installer toujours la dernière version.
2. **Installer** `RStudio` à partir du site
   https://www.rstudio.com/products/rstudio/download2/
3. **Installer Jupyter Notebook**. Quelques instructions sont disponibles à l'adresse
   http://jupyter.readthedocs.io/en/latest/install.html
   Le plus simple pour obtenir Jupyter est de passer à travers Anaconda que l'on peut télécharger sur le site
   https://www.continuum.io/downloads
   Le choix de la version de Python (Python 2.7 ou Python 3.5) n'a pas d'importance pour nous.
4. **Installer le kernel `R` pour Jupyter Notebook**.
   Pour plus d'informations sur l'installation du kernel voir
   https://github.com/IRkernel/IRkernel/blob/master/README.md
   Sous Windows il faut essentiellement ouvrir `R` et exécuter les instructions suivantes :

```
install.packages(c('pbdZMQ', 'repr', 'devtools'))
devtools::install_github('IRkernel/IRkernel')
IRkernel::installspec()
```

   Sous Mac, il faut faire pareil, mais il faut lancer `R` à partir d'un terminal. Pour cela, on tape simplement

   `R`

   dans un terminal, puis on exécute les commandes ci-dessus et pour sortir de `R` on tape

   `q()`

   Ensuite, on lance Jupyter depuis le terminal par la commande

   `jupyter notebook`

   ou l'on peut ouvrir Anaconda et lancer « jupyter notebook ».
   Sous Ubuntu, taper la commande suivante dans le terminal (avec votre mot de passe)

   `sudo apt-get install libcurl4-openssl-dev libssl-dev libzmq3-dev`

   ensuite lancer `R` depuis le terminal par l'instruction

   `R`

   puis tapez

```
install.packages(c(‘crayon‘, ‘pbdZMQ‘, ‘devtools‘))
devtools::install_github(paste0(‘IRkernel/‘, c(‘repr‘, ‘IRdisplay‘,
  ‘IRkernel‘)))
IRkernel::installspec()
```

   Enfin quitter `R` par la commande

   `q()`

   et taper

   `jupyter notebook`

   dans le terminal pour lancer Jupyter.

1.2 Ma première session

Lancer `RStudio` sur votre machine.

Vous voyez apparaître trois fenêtres. La plus importante est celle à gauche : la fenêtre de commandes ou ligne de commandes (*Console*). La console permet d'exécuter des instructions ou commandes. À la fin de l'affichage qui se déroule dans la console, le caractère d'invite de commande `>` vous invitant à taper votre première instruction en langage `R`.

```
>
```

Il s'agit du symbole d'incitation à donner une instruction (*prompt symbol*).

1.2.1 `R` comme calculatrice

Tout d'abord, on peut utiliser `R` comme une calculette classique. Essayons de calculer $2 + 3$:

```
> 2+3

[1] 5
```

`R` nous a compris et affiche le bon résultat. Essayons un exemple légèrement plus difficile :

```
> 3+8^2*(1-3)

[1] -125
```

Nous voyons que `R` connaît bien les règles de calcul et les priorités opératoires. Quand `R` ne comprend pas l'instruction, parce qu'elle est erronée, il affiche un message d'erreur :

```
2+*2

Error: <text>:1:3: '*' inattendu(e)
1: 2+*
     ^
```

Si l'instruction est incomplète, `R` retourne le signe `+` au lieu du prompteur `>`. Il faut alors compléter l'instruction ou sortir de cette situation et récupérer la main en tapant Ctrl + c ou Echap :

```
> 2*(3+1
+ )

[1] 8
```

Il existe de nombreuses fonctions mathématiques prédéfinies sous `R`. Exemples :

```
> exp(1)

[1] 2.718282

> sqrt(2)

[1] 1.414214

> abs(-5.7)

[1] 5.7

> round(-5.7)

[1] -6
```

Lorsqu'on effectue un calcul inadmissible, `R` renvoie la valeur `NaN` (pour *not a number*) et affiche un warning :

```
> sqrt(-2)

Warning in sqrt(-2): production de NaN

[1] NaN
```

La valeur `Inf` représente l'infini. Exemples :

```
> 1/0

[1] Inf

> -exp(exp(100))

[1] -Inf
```

Il faut jamais oublier que des calculs numériques ne sont pas forcément exacts à cause de la discrétisation inévitable des nombres sur une machine. Comparer les sorties suivantes :

```
> sin(0)

[1] 0

> sin(pi)

[1] 1.224647e-16

> sin(2*pi)

[1] -2.449294e-16
```

La précision de machine est de l'ordre de 10^{-16}. Cela semble petit, mais dans certains cas, les erreurs s'amplifient de façon étonnante. Exemple :

```
> sin(pi*10^16)

[1] -0.3752129
```

1.2.2 L'aide dans R

Le logiciel R possède un système d'aide efficace. Cette aide très complète, en anglais, est accessible au moyen de la fonction `help()` ou la commande `?` suivi du nom de la fonction pour laquelle vous cherchez de l'aide. Vous pouvez par exemple taper

```
> help(abs)
```

où de façon équivalente

```
> ?abs
```

pour obtenir de l'aide sur la fonction `abs()`. Vous constatez que chacune de ces commandes ouvre la page d'aide de la fonction `abs()` dans la fenêtre en bas à droite dans RStudio. On y trouve une description de la fonction, la liste des arguments à renseigner avec éventuellement leur valeur par défaut et la liste de résultats retournée par la fonction. En fin d'aide, un ou plusieurs exemples d'utilisation de la fonction sont présents et peuvent être copiés-collés dans R afin de comprendre son utilisation.

Prenez le plus possible l'habitude d'utiliser le système d'aide de R.

1.2.3 Scripts

Au lieu de travailler directement dans la ligne de commandes, vous pouvez écrire des scripts. Un script est un fichier texte qui contient une succession de commandes. Vous pouvez le sauvegarder et le rouvrir lors d'une session ultérieure. Pour développer du code, l'utilisation de scripts est indispensable.

Pour créer un nouveau script et le sauvegarder, on utilise le menu déroulant. La terminaison de fichier d'un script R est `.R`.

Afin d'exécuter une partie des commandes d'un script, on sélectionne les lignes voulues et on appuie sur le bouton Run dans la barre à outils ou on appuie sur Ctrl + R (Cmd + Enter pour les Mac). Vous pouvez aussi exécuter une seule ligne d'instructions R du script en tapant Ctrl + R lorsque le curseur clignotant se trouve sur la ligne en question dans la fenêtre de script. Pour exécuter l'ensemble des instructions d'un script, on clique sur le bouton Source. Cela évitera de surcharger inutilement votre console.

Il est possible d'insérer des commentaires dans vos programmes en les faisant précéder du caractère # :

```
> 'ce qui suit' # ne s'affiche pas

[1] "ce qui suit"
```

Tout ce qui est écrit après le caractère # jusqu'à la fin de la ligne n'est pas interprété par R. Des commentaires permettent de fournir des informations utiles sur votre programme pour les futurs utilisateurs du script.

1.2.4 Répertoire de travail

Le répertoire de travail (en anglais *working directory*) est le répertoire par défaut. C'est par exemple le répertoire qui s'ouvre quand vous cliquez sur le bouton pour enregistrer un script. La commande pour connaître le répertoire de travail actuel est

```
> getwd()

[1] "/Users/tabea/Enseignement/statistique_de_base/poly"
```

Pour le changer, aller dans *Session* R*ightarrow Set Working Directory* R*ightarrow Choose Directory* et sélectionner le dossier souhaité. On peut également changer le répertoire de travail en utilisant la fonction `setwd()` (au risque de ne pas y arriver à cause des nombreuses fautes de frappes possibles...) :

```
> setwd("/Users/tabea/Enseignement/statistique_de_base/2016/TP")
```

Prenez l'habitude de changer le répertoire de travail à chaque ouverture de session avant tout autre chose. Par ailleurs, pour une bonne gestion des fichiers, il est fortement recommandé de créer un dossier pour chaque projet sur lequel vous travailler.

1.2.5 Créer des objets R

La force de R (comme de toute autre langage de programmation) consiste à aller au-delà de la simple utilisation de R comme calculatrice. Comme vous l'avez sans doute remarqué, R répond à vos requêtes en affichant le résultat obtenu après évaluation. Ce résultat est affiché puis perdu. Dans une première utilisation, cela peut paraître agréable, mais dans une utilisation plus poussée il apparaît plus intéressant de rediriger la sortie R de votre requête en la stockant dans une variable. En fait, pour développer des programmes, c'est-à-dire des ensembles de plusieurs d'instructions, il est très utile de pouvoir créer des objets qui stockent de l'information, qu'on peut modifier et avec lesquels on peut faire des calculs.

La création d'un objet R est tout simple. Dès qu'on affecte une valeur à un nom de variable, l'objet – s'il n'existe pas déjà – est créé et prend la valeur qu'on est en train de lui affecter. Cette opération s'appelle affectation du résultat dans une variable. Une affectation évalue ainsi une expression, mais n'affiche pas le résultat qui est en fait stocké dans un objet. Exemple : On crée une variables de nom x qui prend la valeur 74 :

```
> x <- 70 + 4
```

L'exécution de cette instruction ne produit pas de sortie, mais l'objet a bien été créé. Pour afficher la valeur d'un objet, on tape son nom dans la console :

```
> x

[1] 74
```

Maintenant on peut faire des calculs avec cette variable :

```
> 3*x

[1] 222
```

On change la valeur de `x`, en lui affectant une nouvelle valeur :

```
> x <- x+6
```

On vérifie que la valeur de `x` a bien changée :

```
> x

[1] 80
```

Le symbole pour assigner une valeur est soit `<-` soit `=` Exemple :

```
> y = 17
> y

[1] 17
```

Pour les noms d'objet vous avez le droit d'être créatif! En fait, les noms comme `x, y, z` sont de très mauvais exemples. Il vaut mieux utiliser des noms parlants. Si les noms de variables sont bien choisis, les commentaires deviennent presque superflus pour expliquer le code. Voyez l'exemple suivant :

```
> poids <- 52
> taille <- 1.65
> IMC <- poids/taille^2
> IMC

[1] 19.10009
```

Un nom de variable admissible sous `R` doit commencer par une lettre. Il peut aussi inclure des chiffres ou les caractères `.` et `_`

1.2.6 Supprimer des objets

L'**environnement de travail** (ou *work space* en anglais) est une sorte d'espace de stockage d'objets `R` créés lors d'une session, c'est-à-dire c'est la liste d'objets actuellement connus par `R`. Pour connaître tous les objets créés depuis le début de la session, on utilise la commande

```
> ls()

[1] "IMC"    "poids"  "taille" "x"      "y"
```

Pour en supprimer par exemple l'objet `x`, on tape

```
> rm(x)
```

et on vérifie que `x` a bien disparu de l'environnement de travail :

```
> ls()

[1] "IMC"    "poids"  "taille" "y"
```

Afin de supprimer tous les objets d'un seul coup, on écrit

```
> rm(list=ls())
> ls()

character(0)
```

1.2.7 Quitter `RStudio`

À la fermeture de `RStudio`, le logiciel vous propose de sauvegarder une image de la session, ce qui permet d'enregistrer l'ensemble d'objets créés lors de cette session, ou plus précisément d'objets actuellement dans l'environnement de travail. On peut les charger dans une session ultérieure pour reprendre son travail au même endroit.

Notebook n1

Maintenant vous devez travailler le Notebook n1. Les notebooks contiennent également des explications importantes sur le logiciel `R` ainsi que les exercices à faire avec `RStudio`.