# Introduction à R

*Florence Vallée-Dubois*

## Table des matières



### Bienvenue à votre introduction à R !

R a vu le jour en 1993. Ce langage de programmation est de plus en plus utilisé en analyse statistique. Il permet, entre autres, de visualiser des données quantitatives et de faire des analyses statistiques sur ces données. Étant donné que R est un logiciel libre, de nombreux contributeurs et de nombreuses contributrices travaillent à la création de nouvelles fonctions, qui sont rendues disponibles à tous les utilisateurs de R.

Mes notes de cours sont inspirées de celles d'Alexandre Blanchet. Je vous conseille de consulter son site pour une introduction très complète à l'inférence statistique en R. Sinon, il existe de nombreuses autres ressources en ligne.

Sites en français :

- Même si la plupart des forums d'aide sur R sont en anglais, il existe tout de même des blogs francophones qui permettent de démystifier certaines fonctionnalités. Entre autres, on retrouve ElementR et R-atique.
- Le Groupe des utilisateurs du logiciel R est un forum francophone sur R.
- Le site FUN Mooc est une plateforme internet gratuite d'enseignement supérieur. On y offre le cours "Introduction à la statistique avec R".
- L'université Lyon 2 offre le cours "Programmation sous R". Le contenu du cours est disponible sur ce site.

En anglais :

- Vous tomberez probablement assez vite sur Stack Overflow en cherchant des réponses concernant R sur Google. La communauté d'utilisateurs du langage R est très active sur ce site, qui est organisé sous forme de questions-réponses.
- Je consulte régulièrement le site R-bloggers.
- Pour suivre des formations en ligne, ma ressource préférée est DataCamp. Les exercices sont pensés pour un apprentissage graduel et on y offre aussi des formations plus avancées.
- J'ai aussi suivi quelques cours sur Coursera. Les cours ont le format de cours universitaires, mais avec la version gratuite, pas besoin de suivre l'entièreté des plans de cours (ni de remettre les travaux).

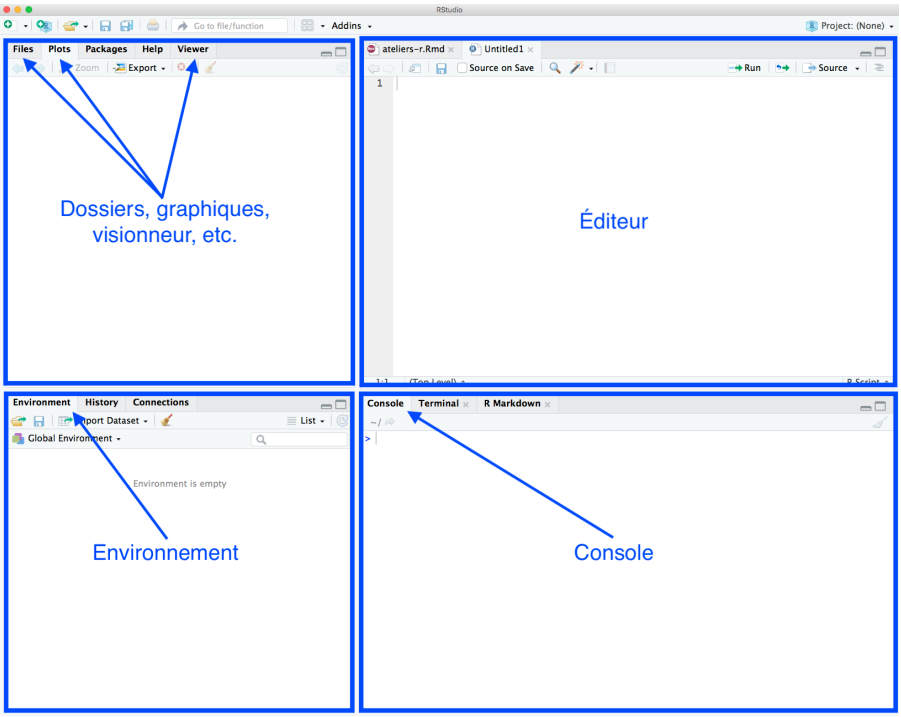



FIGURE 1 – L'interface R Studio

### Un aperçu de R Studio

R Studio est un environnement permettant de travailler avec le langage R dans un espace plus convivial et intuitif. Voici ce à quoi ressemble l'interface de R Studio.

Il est possible, si vous le désirez, de réorganiser l'apparence de R Studio en allant dans l'onglet *R studio –> preferences... –> Appearance/Pane Layout*.

## Pour commencer

### Ouvrez et enregistrez un nouvel éditeur R (*R script*).

C'est dans l'éditeur que vous entrerez toutes vos commandes. En entrant les commandes dans un éditeur (et en ayant enregistré cet éditeur sur votre ordinateur !), vous vous assurez de conserver une trace de votre travail. Évitez d'entrer des lignes de codes directement dans la console, puisque vous n'enregistrerez pas la console à la fin de votre travail.

### Définissez un répertoire

Dans votre éditeur, commencez par définir votre répertoire (en anglais : *working directory*) à l'aide de la fonction `setwd()`. Il est aussi possible d'enregistrer votre répertoire en allant dans *Session –> Set working directory –> Choose directory...*

```
> setwd("~/université/cours/métho/r")
```

Le *working directory* est l'endroit où R ira chercher les données que vous importerez dans votre environnement (il faut donc placer vos banques de données à cet endroit sur votre ordinateur). C'est aussi à cet endroit que seront placés vos modèles, tableaux, graphiques, etc. lorsque vous demanderez à R de les enregistrer.

## Commandes de base

### Faire des commentaires/prendre des notes

À tout moment, vous pouvez prendre des notes pour vous-même dans votre éditeur. Les notes sont tout ce qui est précédé d'un ou plusieurs signes de dièse (`#`). Ces lignes ne seront pas "lues" par R au moment d'exécuter les commandes.

```
> # Ceci est une note -- elle ne sera pas exécutée par R. Annoter votre
> # éditeur vous permet de revenir à votre code plus tard, et de comprendre ce
> # que vous avez fait. Un code bien annoté vous fera gagner du temps! Il vous
> # permettra aussi de facilement partager vos analyses.
```

### Exécution des commandes

Pour exécuter une commande, placez votre curseur sur la ligne que vous souhaiter exécuter, puis appuyez sur CMD + ENTER (en mac) ou sur CTRL + ENTER (en windows). Vous pouvez aussi sélectionner une (ou plusieurs) lignes et cliquer sur le bouton *Run* en haut de votre éditeur.

### Explorer une commande

Lorsque que vous n'êtes pas certain de savoir à quoi sert une commande ou comment elle doit être utilisée, la première étape est de l'explorer dans l'assistant R à l'aide du `?`.

```
> ?setwd
```

Le résultat apparaîtra dans l'onglet *Help*.

### Commandes mathématiques

On peut se servir de R comme d'une calculatrice. Par exemple, on peut faire des additions (`+`), des soustractions (`-`), des divisions (`/`) et des multiplications (`*`). De nombreuses autres opérations mathématiques peuvent y être utilisées.

```
> 4 + 2
## [1] 6
>
> 4 * 2
## [1] 8
>
> # Moyenne: je demande à R de calculer la moyenne des chiffres 4 à 10 (le
> # symbole ':' signifie 'à')
> mean(4:10)
## [1] 7
>
> # Médiane
```

```
> median(20:27)
## [1] 23.5
```

Attention, R est sensible aux majuscules !

#### Assigner des valeurs à des objets

R est un langage "orienté objet" (*object oriented*). En R, on crée des "objets" et on leur assigne des données. Prenons un exemple simple : nous créons deux variables, nommées `variable.x` et `variable.y`, et leur assignons à chacun une valeur. On assigne une valeur à un objet à l'aide du symbole `<-` ou `=`.

```
> variable.x <- 2
> variable.y = 3
>
> # En tapant le nom de l'objet, R nous renvoit son contenu
> variable.x
## [1] 2
>
> # On peut ensuite faire des opérations mathématiques sur ces objets
> variable.x - variable.y
## [1] -1
>
> # On peut même créer un objet qui sera le résultat d'une opération
> operation = variable.x * variable.y
>
> operation
## [1] 6
```

### Types de données

Les trois types de données les plus utiles sont les caractères, les numériques et les booléens.

#### Caractères

Il s'agit de texte. Ce type de données sera entouré de guillemets.

```
> "ceci est ma première variable"
## [1] "ceci est ma première variable"
>
> # Pour conserver cette donnée, il faut l'assigner à un objet: elle apparaît
> # maintenant dans notre environnement.
> coucou <- "ceci est ma première variable"
>
> # Affichons le contenu de l'objet 'coucou'.
> coucou
## [1] "ceci est ma première variable"
```

#### Numériques

Les données numériques sont des nombres, elles ne sont pas entourées de guillemets.

```
> une.donnee.num <- 2.5
> une.donnee.num
## [1] 2.5
```

#### Booléens

Les booléens sont des expressions logiques. Ils prennent la forme `TRUE` ou `FALSE`.

```
> 3 < 2
## [1] FALSE
> # R retourne 'FALSE' parce que 3 n'est pas plus petit que 2
>
> # On peut assigner un énoncé logique à un objet
> boule = 4 * 2 > 3 + 1
>
> # Affichons cet objet
> boule
## [1] TRUE
```

La valeur `TRUE` est assignée à l'objet `boule`, puisque `4*2` est plus grand que `3+1`.

### Structures de données

Les données peuvent avoir différentes structures. Plus haut, chaque objet que nous avons défini ne contenait qu'une seule donnée. L'objet `coucou` ne contient qu'une seule donnée de type caractère, soit la phrase `ceci est ma première variable`. L'objet `une.donnee.num` ne contient qu'une seule donnée de type numérique, soit `2.5`. Et ainsi de suite. La plupart du temps, on voudra assigner plusieurs données à un même objet. Il faudra donc organiser ces données selon une certaine structure.

#### Vecteur

Les vecteurs sont une série de données du même type.

```
> fibonacci <- c(1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987,
+     1597, 2584)
> fibonacci
##  [1]    1    1    2    3    5    8   13   21   34   55   89  144  233  377
## [15]  610  987 1597 2584
>
> ma.famille <- c("Nathalie", "Jean-Francois", "Anthony")
> ma.famille
## [1] "Nathalie"      "Jean-Francois" "Anthony"
```

Rappelez-vous que les vecteurs sont des données du même type, donc si on mélange les types, toutes les données seront converties en un seul type.

```
> fibonacci2 <- c(1, 1, 2, 3, 5, 8, 13, 21, "trente-quatre")
>
> fibonacci2
## [1] "1"             "1"             "2"             "3"
## [5] "5"             "8"             "13"            "21"
## [9] "trente-quatre"
```

```
> # Ici, tout a été converti en caractère (entouré de guillemets)
>
> # On peut appliquer des commandes mathématiques aux vecteurs
> median(fibonacci)
## [1] 44.5
>
> # Pour sélectionner des données dans un vecteur, on peut utiliser les
> # crochets: je demande à R de sortir la 4e donnée du vecteur 'fibonacci'
> fibonacci[4]
## [1] 3
>
> # Et la 1ère et 2e données du vecteur 'ma.famille'
> ma.famille[c(1, 2)]
## [1] "Nathalie"      "Jean-Francois"
>
> # Je pourrais assigner les deux premiers éléments du vecteur 'ma.famille'' à
> # un autre vecteur, que je nommerai 'mes.parents'.
> mes.parents <- ma.famille[c(1, 2)]
> mes.parents
## [1] "Nathalie"      "Jean-Francois"
```

### Liste

Les listes peuvent contenir différents types de données. Elles peuvent aussi contenir des structures de données, comme des vecteurs (mise en abîme des données!).

```
> famille.taille = list("Nathalie", 5.4, "Jean-Francois", 6.4, "Anthony", 5.7)
> famille.taille
## [[1]]
## [1] "Nathalie"
##
## [[2]]
## [1] 5.4
##
## [[3]]
## [1] "Jean-Francois"
##
## [[4]]
## [1] 6.4
##
## [[5]]
## [1] "Anthony"
##
## [[6]]
## [1] 5.7
>
> # Organisons une liste contenant des vecteurs (mise en abîme!)
> famille.taille2 = list(c("Nathalie", "Jean-Francois", "Anthony"), c(5.4, 6.4,
+     5.7))
> famille.taille2
## [[1]]
## [1] "Nathalie"      "Jean-Francois" "Anthony"
##
```

```
## [[2]]
## [1] 5.4 6.4 5.7
> # Le 1er élément de cette liste est un vecteur de 3 prénoms, tandis que le
> # 2e élément de la liste est un vecteur de 3 nombres.
>
> # Je pourrais faire une liste qui associe chaque membre de ma famille à leur
> # taille
> famille.taille3 = list(Nathalie = 5.4, `Jean-Francois` = 6.4, Anthony = 5.7)
> famille.taille3
## $Nathalie
## [1] 5.4
##
## $`Jean-Francois`
## [1] 6.4
##
## $Anthony
## [1] 5.7
>
> # De cette façon, je peux accéder directement à la taille d'un des membres
> # de ma famille en utilisant les double [ ]
> famille.taille3[["Nathalie"]]
## [1] 5.4
>
> famille.taille3[1]
## $Nathalie
## [1] 5.4
> # Cela me renvoit le premier élément de la liste. Il s'agit de Nathalie, et
> # de sa taille.
```

### *Data frame*

Un *data frame* (désolée, je n'ai pas trouvé de traduction satisfaisante) contient des données organisées en rangées et en colonnes, comme dans un tableau. Chaque colonne peut avoir un type différent, mais les données *au sein* d'une même colonne doivent être du même type. Généralement, les colonnes seront associées aux variables, tandis que chaque rangée correspondra à une observation.

```
> famille.info = data.frame(Noms = c("Nathalie", "Jean-Francois", "Anthony", "Florence",
+     "Michelle", "Eric"), Taille = c(5.4, 6.4, 5.7, 5.4, 5.3, 6), Yeux = c("bruns",
+     "bleus", "verts", "bruns", "bruns", "bruns"), Naissance = c(1964, 1974,
+     1989, 1992, 1942, 1991))
>
> # Affichons cet objet
> famille.info
##            Noms Taille  Yeux Naissance
## 1      Nathalie    5.4 bruns      1964
## 2 Jean-Francois    6.4 bleus      1974
## 3       Anthony    5.7 verts      1989
## 4      Florence    5.4 bruns      1992
## 5      Michelle    5.3 bruns      1942
## 6          Eric    6.0 bruns      1991
```

Cette *data frame* compte 6 observations (`Nathalie, Jean-Francois, Anthony, Florence, Michelle, Eric`) et 4 variables (`Noms, Taille, Yeux, Naissance`).

On peut extraire les données qui se trouvent dans une *data frame*.

```
> # À l'aide du symbole '$', R nous donnera le contenu de cette colonne sous
> # forme de vecteur
> famille.info$Taille
## [1] 5.4 6.4 5.7 5.4 5.3 6.0
>
> # On peut sélectionner plus d'une colonne à l'aide de la notation [ , ]: Ce
> # qui est inscrit avant la virgule correspond à la rangée, ce qui vient
> # après la virgule correspond à la colonne
> famille.info[, c("Noms", "Yeux")]
##            Noms  Yeux
## 1      Nathalie bruns
## 2 Jean-Francois bleus
## 3       Anthony verts
## 4      Florence bruns
## 5      Michelle bruns
## 6          Eric bruns
>
> # Ici, on n'a rien précisé pour la rangée, R nous a donc renvoyé le contenu
> # entier des deux colonnes qu'on a précisées. On peut obtenir la même chose
> # en précisant le numéro des colonnes que l'on veut obtenir (la 1ere et la
> # 3e colonne).
> famille.info[, c(1, 3)]
##            Noms  Yeux
## 1      Nathalie bruns
## 2 Jean-Francois bleus
## 3       Anthony verts
## 4      Florence bruns
## 5      Michelle bruns
## 6          Eric bruns
>
> # On peut faire la même chose pour les rangées. Par exemple, le contenu des
> # rangées 1 à 3
> famille.info[1:3, ]
##            Noms Taille  Yeux Naissance
## 1      Nathalie    5.4 bruns      1964
## 2 Jean-Francois    6.4 bleus      1974
## 3       Anthony    5.7 verts      1989
>
> # Les possibilités sont (presque) infinies! Ici, je demande à R de trouver
> # tous les yeux 'bruns' (le double '=' signifie 'correspond à'). R a extrait
> # toutes les rangées où c'est le cas.
> famille.info[famille.info$Yeux == "bruns", ]
##       Noms Taille  Yeux Naissance
## 1 Nathalie    5.4 bruns      1964
## 4 Florence    5.4 bruns      1992
## 5 Michelle    5.3 bruns      1942
## 6     Eric    6.0 bruns      1991
>
> # On peut spécifier plusieurs conditions
> famille.info[famille.info$Yeux == "bruns" & famille.info$Taille > 5.8, ]
##   Noms Taille  Yeux Naissance
## 6 Eric      6 bruns      1991
```

```
> # Une seule personne (Eric) a les yeux bruns et mesure plus de 5.8
>
> # Ici, je demande à R d'extraire tous les individus qui ont les yeux bruns
> # ou (symbole |) qui mesurent plus de 5.8
> famille.info[famille.info$Yeux == "bruns" | famille.info$Taille > 5.8, ]
##            Noms Taille  Yeux Naissance
## 1      Nathalie    5.4 bruns      1964
## 2 Jean-Francois    6.4 bleus      1974
## 4      Florence    5.4 bruns      1992
## 5      Michelle    5.3 bruns      1942
## 6          Eric    6.0 bruns      1991
>
> # Je veux savoir le noms et les dates de naissance (rien d'autre) des
> # individus qui ont les cheveux bruns
> famille.info[famille.info$Yeux == "bruns", c("Noms", "Naissance")]
##       Noms Naissance
## 1 Nathalie      1964
## 4 Florence      1992
## 5 Michelle      1942
## 6     Eric      1991
>
> # On peut faire des opérations mathématiques sur les data frames: calculons
> # la moyenne des tailles
> mean(famille.info$Taille)
## [1] 5.7
```

#### Exercice 1

Créez une *data frame* contenant des données d'au moins 2 types différents. Votre *data frame* doit contenir au moins 4 rangées et 3 colonnes. En utilisant la syntaxe de votre choix, faites l'extraction de deux colonnes. Faites ensuite l'extraction d'une rangée.

## Importer et préparer les données

#### Comment importer

La plupart du temps, les banques de données que vous utiliserez seront structurées en *data frame*. Comme dans un tableau Excel, les colonnes correspondront aux variables et les rangées correspondront aux observations. Si on retourne à l'exemple utilisé plus haut, `Nathalie` est notre première observation. La rangée qui correspond à cette observation est remplie par différentes variables concernant Nathalie, soit sa taille, la couleur de ses yeux et son année de naissance.

Très souvent, vous importerez des banques de données enregistrées en format .csv (données séparées par des virgules). Il se peut que vous utilisiez des banques de données enregistrées sous d'autres formats (ou que vous importiez des banques de données qui se trouvent sur internet). L'avantage avec R, c'est qu'il existe probablement une fonction pour ouvrir ces données. Pour trouver la fonction dont vous avez besoin, je vous conseille de chercher en ligne. Il existe une grande communauté d'utilisateurs et d'utilisatrices du langage R : si vous vous posez une question, il est fort probable que quelqu'un d'autre se la soit posée auparavant ! Vous n'êtes pas ignorant.e !

Avant d'importer une banque de données, il faut d'abord s'assurer qu'elle se trouve dans notre répertoire (*working directory*). Rappelez-vous : c'est avec ce répertoire que R communique. Tout ce que vous "allez chercher"

doit se trouver dans le répertoire que vous avez déjà spécifié.

Pour importer un fichier .csv, on se sert de la fonction `read.csv()`.

```
> travail.femmes = read.csv("women-labour.csv", sep = ",")
```

Il est possible d'importer des données de nombreux formats différents. Par exemple :

```
> baseball = read.table("https://raw.githubusercontent.com/chadwickbureau/
+                       baseballdatabank/master/core/BattingPost.csv",
+     header = TRUE, sep = ",")
```

### Présentation des données que nous utiliserons

Avant d'aller plus loin, voici une brève description de la banque de données que nous utiliserons (source : Mroz 1987, données obtenues de Vincent Arel-Bundock). Il s'agit de la banque *U.S. Women's Labor-Force Participation*, obtenu en 1975 via le *Panel Study of Income Dynamics*. Elle contient des informations 753 femmes américaines mariées. Voici les variables contenues dans cette enquête :

| Variable | Description | Code |
|---|---|---|
| lfp | participation au marché de l'emploi | Yes, No |
| k5 | nombre d'enfants âgés de 5 ans et moins | N.A. |
| k18 | nombre d'enfants âgés de 6 à 18 ans | N.A. |
| age | âge, en années | N.A. |
| wc | fréquentation universitaire par l'épouse | Yes, No |
| hc | fréquentation universitaire par l'époux | Yes, No |
| lwg | salaire (log) des femmes sur le marché du travail (pour les femmes qui ne sont pas sur le marché du travail, une valeur a été imputée sur la base des autres variables) | N.A. |
| inc | revenu du ménage excluant celui de l'épouse | N.A. |

### Explorer vos données

On peut explorer nos données à l'aide de certaines fonctions.

```
> # Voir les premières rangées de la banque de données
> head(travail.femmes)
##   X lfp k5 k618 age  wc hc       lwg    inc
## 1 1 yes  1    0  32  no no 1.2101647 10.910
## 2 2 yes  0    2  30  no no 0.3285041 19.500
## 3 3 yes  1    3  35  no no 1.5141279 12.040
## 4 4 yes  0    3  34  no no 0.0921151  6.800
## 5 5 yes  1    2  31 yes no 1.5242802 20.100
## 6 6 yes  0    0  54  no no 1.5564855  9.859
>
> # Voir les dernières rangées
> tail(travail.femmes)
##       X lfp k5 k618 age  wc  hc       lwg    inc
## 748 748  no  0    2  36  no  no 0.8895015  5.330
## 749 749  no  0    2  40 yes yes 1.0828638 28.200
## 750 750  no  2    3  31  no  no 1.1580402 10.000
## 751 751  no  0    0  43  no  no 0.8881401  9.952
## 752 752  no  0    0  60  no  no 1.2249736 24.984
## 753 753  no  0    3  39  no  no 0.8532125 28.363
```

### Recoder

Recoder des variables est une action fréquente pour tout.e chercheur.se ayant recours à des données quantitatives. Avec le temps, vous trouverez votre propre stratégie pour recoder des variables de la manière qui vous convient le mieux. Ici, l'essai-erreur deviendra votre meilleur ami (et parfois pire ennemi).

```
> # Nous aimerions créer une variable nommée "pgm" qui prendrait la valeur de
> # 1 pour les femmes ayant vécu la Première Guerre mondiale (i.e. qui ont plus
> # de 57 ans), et de 0 pour les autres
> travail.femmes$pgm = ifelse(travail.femmes$age > 57, "1", "0")
>
> # Explorons la fonction ifelse
> ?ifelse
>
> # J'ai demandé à R de créer une nouvelle variable, la variable `pgm`. J'ai
> # ensuite demandé à R d'attribuer la donnée "1" à toutes les observations où
> # la variable "age" était supérieure à 57, et la donnée "0" à toutes les
> # autres observations.
>
> # Nous aimerions créer des catégories d'âge pour chaque décennie, i.e. 30-39
> # ans, 40-49 ans et 50 ans et +
> travail.femmes$decennie = NA
> travail.femmes$decennie[travail.femmes$age <= 39] = "30-39 ans"
> travail.femmes$decennie[travail.femmes$age >= 40 & travail.femmes$age <= 49] =
+   "40-49 ans"
> travail.femmes$decennie[travail.femmes$age >= 50] = "50 ans et plus"
```

Un truc pour vérifier que votre recodage a bel et bien fonctionné : faire un tableau croisé.

```
> table(travail.femmes$decennie, travail.femmes$age)
##
##                 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
##   30-39 ans     38 33 34 25 32 24 38 27 26 21  0  0  0  0  0  0  0  0  0
##   40-49 ans      0  0  0  0  0  0  0  0  0  0 20 27 21 37 24 35 31 38 29
##   50 ans et plus  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
##                 49 50 51 52 53 54 55 56 57 58 59 60
##   30-39 ans      0  0  0  0  0  0  0  0  0  0  0  0
##   40-49 ans     28  0  0  0  0  0  0  0  0  0  0  0
##   50 ans et plus  0 18 28 22 16 26 12 13  7  8  9  6
```

### Diviser une banque de données

```
> # Éliminer la première colonne, qui est une relique du document importé
> travail.femmes = travail.femmes[, -1]
>
> # Rappel: ce qui vient avant la virgule concerne les lignes, ce qui vient
> # après la virgule concerne les colonnes
>
> # Créons une nouvelle banque avec les colonnes 1 à 8 (notez la virgule avant
> # le c)
> femmes2 = travail.femmes[, c(1:8)]
>
```

```
> # Gardons seulement les variables sur le nombre d'enfants et le revenu
> # familial
> femmes3 = travail.femmes[, c("k5", "k618", "inc")]
>
> # Gardons toutes les répondantes qui n'ont pas d'enfants en bas de 5 ans
> # (pour lesquelles il y a un '0' à la variable 'k5')
> aucun.k5 = travail.femmes[travail.femmes$k5 != 0, ]
```

## Statistiques descriptives et analyses univariées

Une fois votre banque de données bien "nettoyée", vous pouvez vous lancer dans l'analyse statistique. Une première étape pertinente est d'explorer vos données de façon descriptive ou de faire des analyses sur une variable à la fois (univariées). Par exemple :

```
> # Tableaux de fréquences
> table(travail.femmes$k618)
##
##   0   1   2   3   4   5   6   7   8
## 258 185 162 103  30  12   1   1   1
```

Ce type de tableau, avec les fréquences à l'horizontal, n'est peut-être pas le plus intuitif, surtout si la variable qui vous intéresse a beaucoup de catégories. Pour remédier à la situation, nous utiliserons la fonction `count()`.

```
> # Exécutons la fonction suivante
> count(travail.femmes$k618)
```

R devrait renvoyer un message d'erreur. La fonction `count()` est introuvable. La raison est la suivante : R contient en tout temps certaines fonctions de base, comme celles que nous avons utilisées depuis le début de cet atelier (`setwd()`, `table()`, `mean()`, etc.). Or, pour utiliser d'autres fonctions, il est nécessaire d'installer des paquets (*packages*). Il existe des centaines de paquets, que les collaborateurs et collaboratrices du langage R travaillent à développer. Ces paquets contiennent des fonctions qui vous permetteront de réaliser diverses actions. Souvent, la seule manière pour vous de connaître l'existence d'une fonction (et de son paquet) sera de faire des recherches sur internet.

Ma recherche internet m'indique que la fonction `count()` se trouve dans le paquet nommé `plyr`. Installons-le.

```
> install.packages("plyr")
```

La deuxième étape essentielle est d' "appeler" le paquet à l'aide de la fonction `library()`.

```
> library(plyr)
```

Il faut voir la fonction `library()` comme un service de bibliothécaire (littéralement). Après avoir installé le *package* `plyr`, ce dernier se trouvait en mémoire de R. Cependant, il vous était encore impossible de l'utiliser parce que vous ne l'aviez pas "fait venir à la surface".

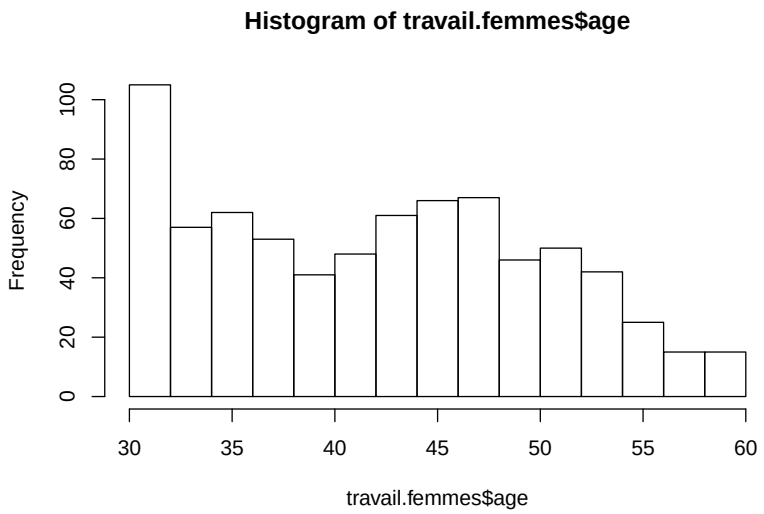
Maintenant que tout est installé, retournons à nos statistiques descriptives :

```
> count(travail.femmes$k618)
##   x freq
## 1 0  258
## 2 1  185
## 3 2  162
## 4 3  103
## 5 4   30
## 6 5   12
## 7 6    1
```

```
## 8 7    1
## 9 8    1
```

Il est également possible de visualiser nos variables à l'aide de graphiques.

```
> hist(travail.femmes$age)
```




Il est possible de calculer des mesures de tendances centrales, comme la moyenne et la médiane.

```
> # Moyenne d'âge des répondantes
> mean(travail.femmes$age)
## [1] 42.53785
>
> # Calcul de la moyenne de la variable lfp, pour connaître le nombre de
> # femmes qui sont sur le marché du travail
> mean(travail.femmes$lfp)
## Warning in mean.default(travail.femmes$lfp): argument is not numeric or
## logical: returning NA
## [1] NA
>
> # R renvoit un message d'erreur parce que la variable n'est pas numérique
> is.numeric(travail.femmes$lfp)
## [1] FALSE
>
> # Je peux calculer la moyenne de la variable lfp en la dédoublant en
> # variable numérique
> travail.femmes$lfp.num = as.numeric(travail.femmes$lfp)
> table(travail.femmes$lfp.num, travail.femmes$lfp)
##
##      no yes
##   1 325   0
##   2   0 428
>
> mean(travail.femmes$lfp.num)
## [1] 1.568393
>
> # Âge moyen des femmes qui travaillent seulement
> mean(travail.femmes$age[travail.femmes$lfp == "yes"])
## [1] 41.97196
```

```
>
> # Les femmes qui travaillent sont en moyenne plus jeunes que les femmes qui
> # ne travaillent pas.
>
> # Calcul de l'écart type
> mean(travail.femmes$k618)
## [1] 1.353254
> sd(travail.femmes$k618)
## [1] 1.319874
>
> # En moyenne, les répondantes ont 1.35 enfant de 6 à 18 ans. L'écart-type
> # associé à cette statistique est 1,32.
>
> # Je voudrais savoir le nombre total d'enfants qu'ont ces répondantes
> travail.femmes$enfants = travail.femmes$k618 + travail.femmes$k5
> count(travail.femmes$enfants)
##   x freq
## 1 0  229
## 2 1  161
## 3 2  167
## 4 3  117
## 5 4   56
## 6 5   15
## 7 6    5
## 8 7    1
## 9 8    2
```

## Analyses bivariées et statistiques inférentielles

R peut servir à analyser la relation entre deux ou plusieurs variables. De nombreux tests, comme la corrélation de Pearson, sont disponibles sur R. Après avoir analysé la force de la relation entre deux variables, vous pouvez également tester la signification statistique de cette dernière à l'aide, par exemple, de tests comme le khi carré.

Première question : La participation des femmes au marché de l'emploi (`lfp`) est-elle liée au fait qu'elles aient fréquenté l'université (`wc`) ?

```
> # Faisons un tableau croisé pour visualiser la relation (potentielle)
> table(travail.femmes$lfp, travail.femmes$wc)
##
##        no yes
##   no  257  68
##   yes 284 144
>
> # Faisons des variables numériques pour lfp et wc
> travail.femmes$wc.num = ifelse(travail.femmes$wc == "yes", 1, 0)
> travail.femmes$hc.num = ifelse(travail.femmes$hc == "yes", 1, 0)
>
> # Corrélation
> cor(travail.femmes$lfp.num, travail.femmes$wc.num)
## [1] 0.1401022
```

La corrélation entre la participation au marché de l'emploi et la fréquentation universitaire est très faible (0.14). Pour connaître le niveau de signification statistique, utilisons la fonction `rcorr()` du paquet `Hmisc`.

```
> install.packages("Hmisc")
```

```
> library(Hmisc)
>
> rcorr(travail.femmes$lfp.num, travail.femmes$wc.num)
##      x    y
## x 1.00 0.14
## y 0.14 1.00
##
## n= 753
##
##
## P
##   x     y
## x       1e-04
## y 1e-04
> # rcorr nous donne la statistique p associée à chaque corrélation
> # L'association statistique entre lpf et wc est significative à ~0.0001
>
> # Pour connaître la signification statistique, on peut aussi utiliser le khi
> # carré
> chisq.test(travail.femmes$lfp.num, travail.femmes$wc.num)
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  travail.femmes$lfp.num and travail.femmes$wc.num
## X-squared = 14.158, df = 1, p-value = 0.0001681
```

## Régression linéaire et analyses multivariées

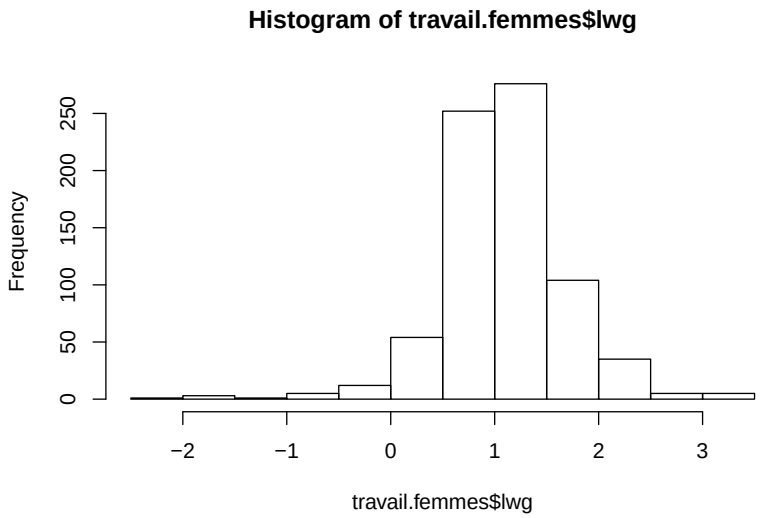
Enfin, R nous permet de faire des analyses de régression par les moindres carrés ordinaires (*OLS*). Passons en revue les éléments les éléments à inclure dans la fonction `lm()`.

```
> ?lm
```

Est-ce que le nombre d'enfants est associé au salaire des femmes ? On pourrait poser l'argument selon lequel les femmes qui ont plus d'enfants ont un salaire moins élevé que les autres, puisqu'elles ont dû s'absenter du marché du travail pour une plus longue période de temps. Or, l'ancienneté et l'expérience sont généralement associées positivement au salaire, et les législations relatives à l'équité salariale et aux congés de parentalité étaient encore moins répandues en 1975.

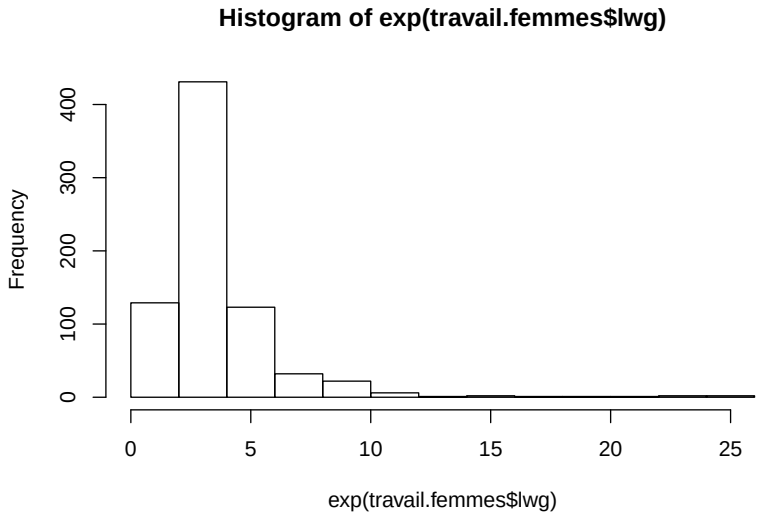
Le salaire des femmes (`lwg`) est la variable dépendante, tandis que le nombre d'enfants (`enfants`) est la variable indépendante, que nous avons créée plus tôt. N'oubliez pas que la variable `lwg`, dans cet échantillon, est le log du salaire. Elle va de -2.05 à 3.2. Traçons l'histogramme pour voir.

```
> hist(travail.femmes$lwg)
```




La variable du salaire est souvent exprimée en log, parce que dans la population, la distribution des salaires ne suit souvent pas une distribution normale. Dans la population, il y a généralement quelques individus très fortunés qui représentent une proportion très limitée des travailleurs, alors que la majorité des personnes ont un revenu beaucoup plus bas. Voyez ce à quoi ressemble la distribution réelle des salaires des répondantes de cette enquête :

```
> hist(exp(travail.femmes$lwg))
```




Les chercheurs ayant mené cette étude ont donc utilisé le log pour donner une distribution normale à la variable “salaire”.

Fermons la parenthèse et revenons à notre modèle linéaire. Le nombre d’enfants affecte-t-il négativement le salaire de nos répondantes ?

```
> # Estimation du modèle
> modelelineaire = lm(lwg ~ enfants, data = travail.femmes)
>
> # Résultats
> modelelineaire
##
## Call:
## lm(formula = lwg ~ enfants, data = travail.femmes)
##
```

```
## Coefficients:
## (Intercept)      enfants
##     1.16198     -0.04077
>
> # Résultats avec statistiques inférentielles
> summary(modelelineaire)
##
## Call:
## lm(formula = lwg ~ enfants, data = travail.femmes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.09379 -0.27197 -0.02544  0.29874  2.21997
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.16198    0.03153  36.851  < 2e-16 ***
## enfants     -0.04077    0.01460  -2.791  0.00538 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5849 on 751 degrees of freedom
## Multiple R-squared:  0.01027,    Adjusted R-squared:  0.008951
## F-statistic: 7.792 on 1 and 751 DF,  p-value: 0.00538
```

Portez votre regard sur la colonne `Estimate`. Le coefficient associé à la variable `enfants` est égal à -0.04.

Interprétons : dans cet échantillon de femmes américaines, une augmentation d'un enfant est associé avec une diminution (du log de) de son salaire de 0.04. En bref, une augmentation dans le nombre d'enfants est associé à une diminution du salaire pour ces femmes américaines.

La statistique `P` associée à cette relation nous indique qu'il s'agit d'une relation statistiquement significative (à < 0.01). Le coefficient de -0.04 se trouve dans une intervalle allant d'environ -0.05 à -0.03, au moins 99 fois sur 100. Ici, je vous réfère à vos livres de statistiques inférentielles pour vous familiariser avec l'interprétation. Mais pour résumer, si nous répétions la même analyse avec une infinité d'échantillons, nous obtienderions un coefficient se trouvant entre -0.05 et -0.03 au moins 99 fois sur 100. Il s'agit d'un niveau de signification statistique accepté dans la littérature en science politique (généralement, on accepte < 0.05).

Il est possible que d'autres facteurs expliquent aussi la relation négative entre le nombre d'enfants et le salaire.

- Par exemple, il est possible que les femmes plus âgées aient plus d'enfants (entre autres, parce qu'elles sont nées à une époque où les familles étaient plus nombreuses) *et* qu'elles aient un salaire moins élevés (parce qu'au moment où elles ont intégré le marché du travail, les salaires des femmes étaient encore plus bas).
- D'autre part, il est possible que le niveau d'éducation affecte le nombre d'enfants *ainsi* que le salaire.

La relation entre le nombre d'enfants et le salaire est donc peut-être fallacieuse. Pour vérifier cette possibilité, ajoutons des variables de contrôle à notre modèle : l'**âge de la répondante** et le **fait qu'elle soit allée à l'université**.

```
> modelecontrole = lm(lwg ~ enfants + age + wc.num, data = travail.femmes)
>
> summary(modelecontrole)
##
## Call:
## lm(formula = lwg ~ enfants + age + wc.num, data = travail.femmes)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.13243 -0.24015  0.00586  0.27144  2.33185
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.129968   0.141871   7.965 6.15e-15 ***
## enfants     -0.044857   0.016098  -2.787  0.00546 **
## age         -0.001764   0.002918  -0.605  0.54564
## wc.num       0.403371   0.045212   8.922  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5563 on 749 degrees of freedom
## Multiple R-squared:  0.1072, Adjusted R-squared:  0.1036
## F-statistic: 29.97 on 3 and 749 DF,  p-value: < 2.2e-16
```

Après avoir ajouté ces deux variables, on note que la relation entre l'âge et le salaire est légèrement négative (-0.001) et non-significative (p=0.54). Le fait d'être allée à l'université est quant à lui positivement corrélé avec le salaire des répondantes. En moyenne, le (log du) salaire des femmes ayant fréquenté l'université est 0.4 plus élevé que le (log du) salaire des femmes n'ayant pas fréquenté l'université. Si on menait cette analyse à nouveau, notre coefficient se trouverait dans l'intervalle de 0.35 à 0.45 au moins 99.9% du temps.

Est-ce que la relation négative entre le nombre d'enfants et le salaire tient encore, malgré l'introduction des variables "âge" et "fréquentation de l'université"? Il semblerait que oui. Même en contrôlant l'âge et la fréquentation universitaire, chaque enfant supplémentaire fait diminuer le salaire des répondantes de ce sondage. De plus, cette relation peut être généralisée à la population dont est tiré cet échantillon (p<0.01).

#### Exercice 2

Testez une hypothèse de recherche à l'aide des données du *U.S. Women's Labor-Force Participation*. Composez un modèle de régression linéaire avec au moins une variable de contrôle et interprétez les coefficients ainsi que les niveaux de signification statistique.

## Pour vos projets futurs

Cette introduction à R a effectué un survol des principales fonctionnalités de ce langage, très utile en analyse statistique. Il vous aura appris, je l'espère, à utiliser les fonctions de base en R, à importer des bases de données, à recoder des variables, à utiliser des *packages*, à visualiser vos données et à faire des analyses univariées, bivariées ainsi que des régressions linéaires multivariées. Vous pouvez maintenant utiliser R pour mener d'autres types d'analyses statistiques, comme les régressions logistiques. Je vous invite à explorer les sources citées plus haut si vous avez des questions en cours de cheminement. R est un outil très flexible, vous serez surpris.e des possibilités qui s'ouvrent à vous en termes de modélisation et de visualisation de données!