Tutoriel d'introduction au langage R et à RStudio

Par Christophe Lalanne - Bruno Falissard

Date de publication : 3 juillet 2018

Ce cours regroupe des informations utiles pour démarrer avec le langage R en utilisant le logiciel **RStudio**. Les instructions R utilisées dans chaque session (« lab ») sont reprises en détail.

Commentez

I - Introduction

Avant de commencer, il est nécessaire de s'assurer que le logiciel R (1) fonctionne correctement sur votre machine. L'installation du logiciel **RStudio** (2) ne devrait pas poser non plus de difficulté particulière.

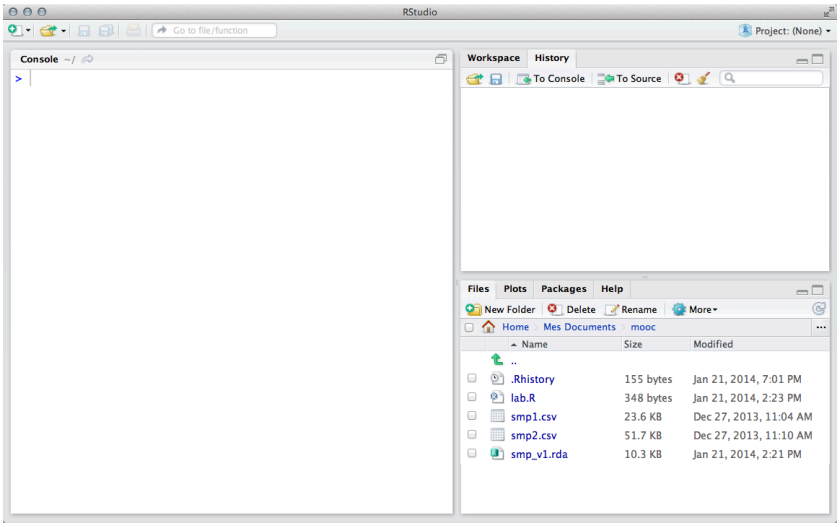
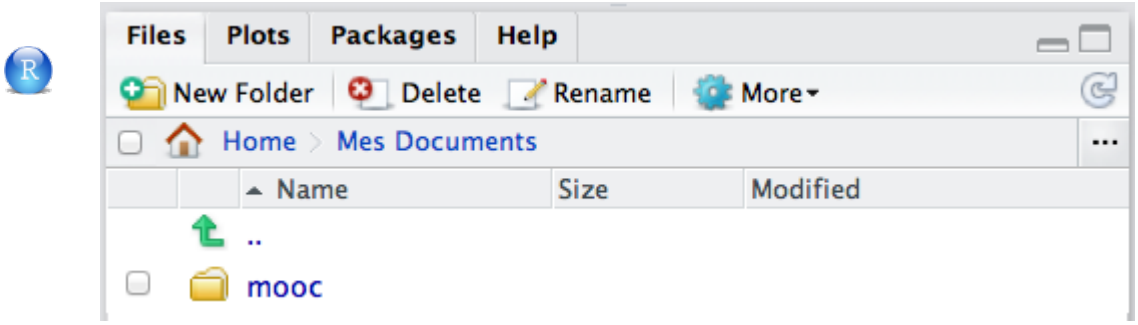


Figure 1 – L'interface RStudio est composée de différents panneaux, dont l'arrangement peut être reconfiguré, incluant une console, un navigateur de fichiers et graphiques, l'espace de travail et l'historique des commandes.

Gestion de fichiers

On utilisera le menu Files. Celui-ci permet également de créer de nouveaux répertoires (New Folder) et de définir le répertoire courant comme répertoire de travail (More > Set Working Directory).



II - Lab 1 : Importation de données et manipulation de variables

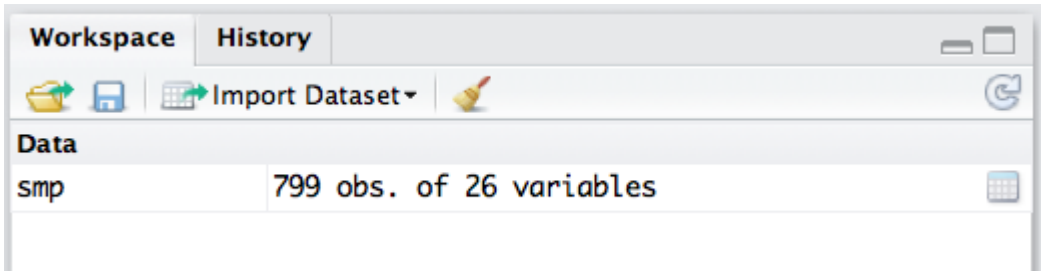
Le chargement du fichier ***smp2.csv*** se fait à l'aide de la commande `read.csv2()`, qui permet de lire des fichiers CSV pour lesquels le séparateur de champ est un point-virgule, et le séparateur décimal une virgule. Les données seront associées à la variable `smp` à l'aide de l'opérateur d'affectation `<-`. Dans ce qui suit, on supposera que l'utilisateur a bien défini le répertoire contenant le fichier *smp2.csv* comme répertoire de travail, soit à l'aide de la commande `setwd()` soit via le menu de **Rstudio**.

```
1. smp <- read.csv2("smp2.csv")
```

II-A - Représentation d'un tableau de données

Gestion de données

Le contenu de l'espace de travail peut être visualisé directement depuis le panneau intitulé « Workspace ». Si l'on double-clique sur le nom d'un data frame, celui-ci est affiché dans un tableau en mode lecture seule.



Les données sont généralement représentées sous la forme d'un tableau rectangulaire dans lequel les variables sont arrangées en colonnes, et les observations en lignes. Sous R, on parlera de *data frame*.

Une fois que les données sont importées, la variable smp sera disponible dans l'espace de travail (« workspace »), comme on pourra le vérifier avec la commande ls(). Les dimensions du tableau de données peuvent être vérifiées à l'aide de dim(), et la commande names() renvoie le nom des variables.

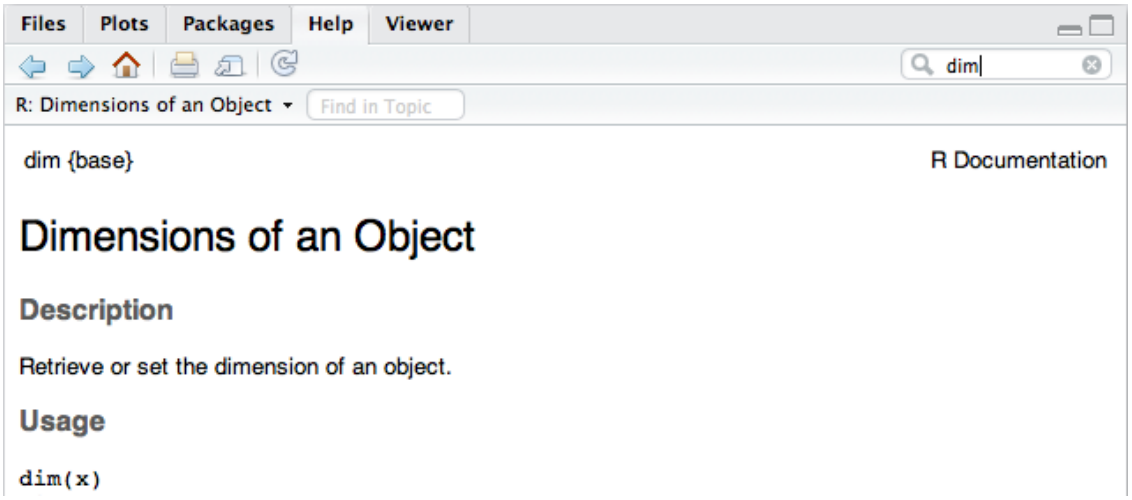
```
> ls()
[1] "abus"          "help_console" "m"            "res"
[5] "smp"           "smpb"         "tab"

> dim(smp)
[1] 799  26

> names(smp)
 [1] "age"          "prof"         "duree"        "discip"
 [5] "n.enfant"     "n.fratrie"    "ecole"        "separation"
 [9] "juge.enfant"  "place"        "abus"         "grav.cons"
[13] "dep.cons"     "ago.cons"     "ptsd.cons"    "alc.cons"
[17] "subst.cons"   "scz.cons"     "char"         "rs"
[21] "ed"           "dr"           "suicide.s"    "suicide.hr"
[25] "suicide.past" "dur.interv"
```

Aide en ligne

***RStudio** offre un panneau spécifique permettant d'accéder à l'aide en ligne, incluant un moteur de recherche et un index des commandes associées.*



Pour accéder à l'aide en ligne pour n'importe quelle commande R, on utilise la commande help(). Sous **RStudio**, il existe en plus un moteur de recherche intégré, fonctionnant par mots-clés ou à l'aide d'un système d'index. Chaque page d'aide est organisée de la même manière et contient une description des principales options de la commande R, incluant les valeurs par défaut, ainsi que des exemples d'utilisation.

```
> help(dim)
Dimensions of an Object

Description:

     Retrieve or set the dimension of an object.

Usage:

     dim(x)
     dim(x) <- value

Arguments:

       x: an R object, for example a matrix, array or data frame.

   value: For the default method, either ' NULL ' or a numeric vector,
          which is coerced to integer (by truncation).

Details:
```

La commande str() fournit un résumé de l'ensemble des variables avec leur type (ou mode de représentation) — *int* pour les variables numériques, *factor* pour les variables catégorielles — et un aperçu des dix premières observations.

```
> str(smp)
'data.frame'  : 799 obs. of 26 variables:
$ age         : int 31 49 50 47 23 34 24 52 42 45 ...
$ prof        : Factor w/ 8 levels "agriculteur",..: 3 NA 7 6 8 6 3 2 6 6 ...
$ duree       : int 4 NA 5 NA 4 NA NA 5 4 NA ...
$ discip      : int 0 0 0 0 1 0 0 0 1 0 ...
$ n.enfant    : int 2 7 2 0 1 3 5 2 1 2 ...
$ n.fratrie   : int 4 3 2 6 6 2 3 9 12 5 ...
$ ecole       : int 1 2 2 1 1 2 1 2 1 2 ...
$ separation  : int 0 1 0 1 1 0 1 0 1 0 ...
$ juge.enfant : int 0 0 0 0 NA 0 1 0 1 0 ...
$ place       : int 0 0 0 1 1 0 1 0 0 0 ...
$ abus        : int 0 0 0 0 0 0 0 0 1 1 ...
$ grav.cons   : int 1 2 2 1 2 1 5 1 5 5 ...
$ dep.cons    : int 0 0 0 0 1 0 1 0 1 0 ...
$ ago.cons    : int 1 0 0 0 0 0 0 0 0 0 ...
$ ptsd.cons   : int 0 0 0 0 0 0 0 0 0 0 ...
$ alc.cons    : int 0 0 0 0 0 0 0 0 1 1 ...
```

```
19. $ subst.cons  : int 0 0 0 0 0 0 1 0 1 0 ...
20. $ scz.cons    : int 0 0 0 0 0 0 0 0 0 0 ...
21. $ char        : int 1 1 1 1 1 1 1 1 4 1 ...
22. $ rs          : int 2 2 2 2 2 1 3 2 3 2 ...
23. $ ed          : int 1 2 3 2 2 2 3 2 3 2 ...
24. $ dr          : int 1 1 2 2 2 1 2 2 1 2 ...
25. $ suicide.s   : int 0 0 0 1 0 0 3 0 4 0 ...
26. $ suicide.hr  : int 0 0 0 0 0 0 1 0 1 0 ...
27. $ suicide.past: int 0 0 0 0 1 0 1 0 1 0 ...
28. $ dur.interv  : int NA 70 NA 105 NA NA 105 84 78 60 ...
```

Une autre commande utile pour inspecter la structure de données est la commande summary() qui fournit un résumé descriptif univarié pour chaque variable. Dans le cas des variables numériques, R indique les principaux indicateurs de tendance centrale (moyenne et médiane) et de dispersion (étendue, intervalle interquartile), ainsi que le nombre de valeurs manquantes représentées par le symbole NA. Dans le cas des variables catégorielles, R fournit un tableau d'effectifs, c'est-à-dire le nombre d'observations associées à chacune des modalités de la variable.

```
1. summary(smp)
```

II-B - Travailler avec des variables numériques

Pour afficher l'ensemble des observations recueillies pour une variable donnée, on tapera le nom de cette variable préfixé du nom du data frame suivi du signe $. L'expression smp$age désignera ainsi les valeurs prises par la variable age dans le data frame smp. Plutôt que d'afficher l'ensemble des valeurs, on peut vouloir limiter l'affichage à certaines observations, que l'on désignera par leur numéro. On peut ainsi choisir d'afficher, la première, les deux premières, ou la première et la troisième observation à l'aide des instructions suivantes :

```
1. > smp$age[1]
2. [1] 31
3.
4. > smp$age[c(1,2)]
5. [1] 31 49
6.
7. > smp$age[c(1,3)]
8. [1] 31 50
```

Il est également possible d'afficher un ensemble d'observations consécutives en utilisant la notation [i:j] où i et j désignent, respectivement, le premier et le dernier numéro d'observation. Pour afficher les dix premières observations, on utilisera donc la commande :

```
1. > smp$age[1:10]
2. [1] 31 49 50 47 23 34 24 52 42 45
```

En fait, il existe une commande, head(), qui produit exactement le même résultat que la dernière instruction et qui permet d'afficher les premières observations d'une variable. Par défaut, head() affichera les six premières observations, comme on pourra le vérifier dans l'aide en ligne. Cette valeur par défaut peut être modifiée en ajoutant l'option n=10, par exemple.

```
1. > head(smp$age)
2. [1] 31 49 50 47 23 34
3.
4. > head(smp$age, n=10)
5. [1] 31 49 50 47 23 34 24 52 42 45
```

Notons que si l'on respecte l'ordre de présentation des options d'une commande, il n'est pas nécessaire de nommer explicitement les arguments. Ainsi, la dernière instruction est équivalente à :

```
1. head(smp$age, 10)
```

La commande summary() fonctionne également avec une variable et fournit, comme dans le cas d'un data frame, un résumé numérique de la distribution de la variable selon son type.

```
> summary(smp$age)
Min.    1st Qu.    Median    Mean    3rd Qu.    Max.    NA's
19.0       28.0      37.0    38.9       48.0    83.0       2
```

Le minimum observé peut être obtenu avec la commande min(). Or dans le cas précis de la variable age, on constate que R renvoie la valeur **NA**.

```
> min(smp$age)
[1] NA
```

En effet, dans le cas où une variable contient des données manquantes, il faut indiquer explicitement à R ce que l'on souhaite faire des valeurs manquantes. Par défaut, R ne supprime pas les valeurs manquantes, comme on peut le vérifier dans l'aide en ligne (na.rm=**FALSE**), et se contente de renvoyer la valeur **NA** pour signaler à l'utilisateur que certaines données sont manquantes.

```
> min(smp$age, na.rm=TRUE)
[1] 19
```

L'étendue des valeurs observées s'obtient avec range() qui, contrairement à min(), est une commande qui renvoie des résultats multiples (dans ce cas, le minimum et le maximum de la variable age).

```
> range(smp$age, na.rm=TRUE)
[1] 19 83
```

II-C - Travailler avec des variables binaires

La variable abus est une variable qui ne prend que des valeurs 0 et 1, à l'exception des données manquantes (**NA**), comme le confirme le résultat renvoyé par la commande unique() qui permet d'énumérer les valeurs distinctes observées dans une variable.

```
> head(smp$abus)
[1] 0 0 0 0 0 0

> unique(smp$abus)
[1]  0  1 NA
```

Le nombre total d'observations contenues dans la variable abus peut être obtenu avec la commande length(), mais la valeur retournée correspond en fait au nombre total d'éléments contenus dans abus, qui est identique au nombre de lignes du tableau de données. Un simple tri à plat de la variable confirmera que le nombre total de données observées (792), c'est-à-dire non manquantes, est inférieur au nombre total d'individus dans la base de données (799).

```
> length(smp$abus)
[1] 799

> nrow(smp)
[1] 799

> table(smp$abus)
  0   1
572 220

> sum(table(smp$abus))
[1] 792
```

Pour afficher le nombre de valeurs manquantes via la commande table(), il est nécessaire d'ajouter l'option useNA="always".

```
> table(smp$abus, useNA="always")
   0   1 <NA>
 572 220   7
```

On peut très bien compter manuellement les valeurs manquantes à l'aide de la commande is.na() qui renvoie **TRUE** si l'observation est manquante, **FALSE** autrement (3) . La négation logique, exprimée à l'aide de l'opérateur !is.na, permet d'identifier les observations sans donnée manquante pour la variable abus. On lui préfèrera la commande complete.cases(), plus intuitive et plus simple d'utilisation.

```
> sum( is.na(smp$abus))
[1] 7

> sum(!is.na(smp$abus))
[1] 792

> sum(complete.cases(smp$abus))
[1] 792
```

II-D - Représentation des variables qualitatives

Une importante distinction concernant le codage des variables sous R est la notion de facteur (« factor »).

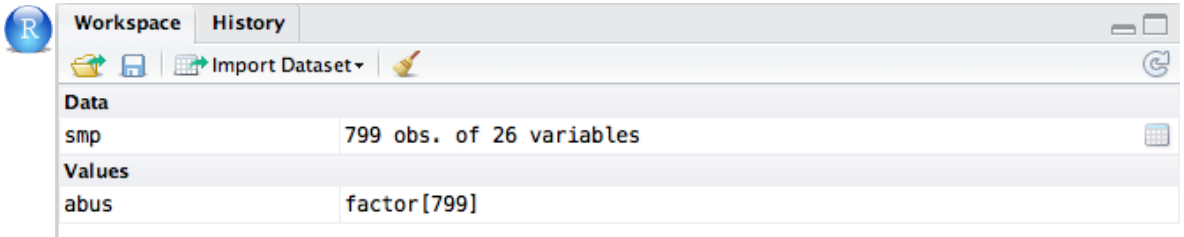
Avec la variable abus discutée précédemment, on peut vérifier que l'usage de la commande factor() ne modifie en rien le contenu de la variable, mais associe à chaque valeur observée un niveau (« level ») unique, ici 0 ou 1. On peut d'ailleurs explicitement changer les « étiquettes » associées aux niveaux du factor en spécifiant l'option labels=.

```
> head(factor(smp$abus))
[1] 0 0 0 0 0 0
Levels: 0 1

> head( factor(smp$abus, labels=c("Non", "Oui")))
[1] Non Non Non Non Non Non
Levels: Non Oui
```

Gestion du workspace

La création d'une variable en dehors d'un data frame se traduit par l'ajout d'une variable dans l'espace de travail, ce que l'on peut vérifier dans le panneau « Workspace ». Les data frames sont listés séparément des variables simples.



Pour rendre cette modification définitive, il est possible de créer une variable auxiliaire directement dans l'espace de travail. Celle-ci ne fera pas partie du data frame smp, mais il sera possible de la manipuler comme n'importe quelle variable. On peut vérifier qu'une fois la conversion en facteur effectuée, le tableau d'effectifs reste strictement identique à celui affiché **ici** et que seules les modalités ont été modifiées (Non/Oui à la place de 0/1). Enfin, la commande relevel() permet de changer la catégorie ou modalité de référence (par défaut, R suit l'ordre lexicographique ou l'ordre de présentation des niveaux dans l'option levels=.

```
> abus <- factor(smp$abus, labels=c("Non", "Oui"))
> table(abus)
```

```
3. abus
4. Non Oui
5. 572 220
6.
7. > head(relevel(abus, ref="Oui"))
8. [1] Non Non Non Non Non Non
9. Levels: Oui Non
```

II-E - Recodage de variables

Considérons la variable n.enfant, qui représente le nombre d'enfants dans la famille du répondant. Un simple tableau d'effectifs confirmera la nature discrète de cette variable, et on notera que l'on peut très bien effectuer un tableau d'effectifs à partir d'une condition logique, par exemple dénombrer les individus ayant quatre enfants ou moins et ceux ayant plus de quatre enfants, comme illustré ci-dessous :

```
1. > table(smp$n.enfant, useNA="always")
2.    0    1    2    3    4    5    6    7    8    9   10   11   13 <NA>
3.  214  220  125  101   55   31    7    7    7    2    2    1    1   26
4.
5. > head(smp$n.enfant > 4)
6. [1] FALSE  TRUE FALSE FALSE FALSE FALSE
7.
8. > table(smp$n.enfant > 4)
9. FALSE  TRUE
10.   715    58
```

Supposons à présent que l'on souhaite regrouper dans une même classe les individus ayant cinq enfants ou plus. Pour cela, il suffit de convertir la variable n.enfant en facteur, puis d'agréger ces derniers niveaux grâce à la commande levels().

```
1. > smp$n.enfant.cat <- factor(smp$n.enfant)
2. > levels(smp$n.enfant.cat)[6:13] <- "5+"
3. > nlevels(smp$n.enfant.cat)
4. [1] 6
5.
6. > table(smp$n.enfant.cat)
7.   0   1   2   3   4  5+
8. 214 220 125 101  55  58
```

Il est également possible de recoder en variable qualitative des variables continues à l'origine. Par exemple, avec la variable age, il serait tout à fait possible de créer une variable age.cat à quatre classes d'effectifs à peu près équilibrés à l'aide de la commande cut() (4) . Avec cette commande, il est important de préciser l'option **include.lowest=TRUE** pour ne pas oublier d'inclure l'observation dont l'âge vaut l'âge minimal, puisque par défaut les intervalles ont des bornes ouvertes (c'est-à-dire n'incluant pas) à gauche.

```
1. > smp$age.cat <- cut(smp$age, breaks=c(19, 25, 35, 45, 83),
2. >                    include.lowest=TRUE)
3. > table(smp$age.cat)
4. [19,25] (25,35] (35,45] (45,83]
5.     136     223     207     231
```

II-F - Sauvegarde de données

Il est tout à fait possible de sauvegarder des données au format RData, qui est un format propre au logiciel R. Cela facilite l'archivage de résultats intermédiaires ou l'enregistrement d'un tableau de données nettoyé (recodage de valeurs manquantes ou modalités de variables qualitatives, correction des erreurs de saisie, etc.) ou augmenté de variables auxiliaires. Pour cela, on utilisera la commande save(). La commande load() permet quant à elle de recharger des données sauvegardées au format RData. L'extension du fichier peut être indifféremment RData ou rda.

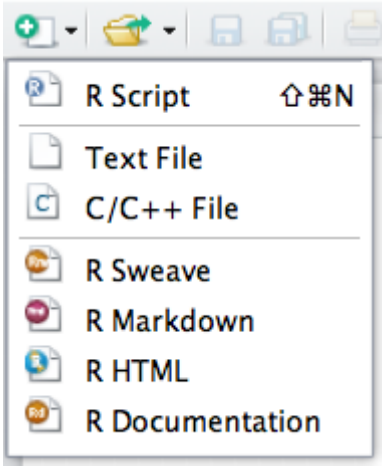
```
1. > save(smp, file="smp_v1.rda")
2. > dir(pattern="rda")
```

```
3. [1] "smp_v1.rda"
```

II-G - Enregistrement des commandes

Création d'un script R

Pour enregistrer les commandes utiles pour le projet d'analyse statistique, il suffit de créer un fichier R Script. Les commandes peuvent être copiées depuis le panneau d'historique.



Plutôt que de saisir les commandes directement dans la console, il est tout à fait possible de les enregistrer directement dans un script de commandes R, ce qui permet de reproduire les analyses ultérieurement. Dans l'éditeur interne à **RStudio**, il suffit d'écrire les commandes telles qu'on les entrerait sous la console et il est possible de transférer directement la ligne sur laquelle le curseur se situe ou une sélection multiple à la console en cliquant sur le bouton *RUN*.

Une fois enregistrées dans un fichier script R (extension .r ou .R), il est possible d'exécuter l'ensemble des instructions à l'aide de la commande source(). Attention, dans ce cas, R n'affiche pas nécessairement l'intégralité des résultats qui sont affichés en mode interactif, à moins d'ajouter l'option echo=**TRUE**.

Voici l'essentiel des commandes de ce premier Lab enregistrées dans un fichier *labs.R* dans le même répertoire que celui contenant les données.

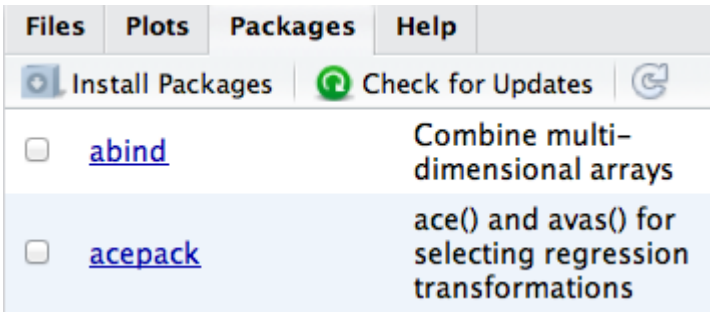
labs.R

```
1. ## Chargement des données
2. setwd("~/Mes Documents/mooc")
3. smp <- read.csv2("smp2.csv")
4.
5. ## Structure de données
6. dim(smp)
7. names(smp)
8. str(smp)
9.
10. ## Recodage de variables
11. smp$n.enfant.cat <- factor(smp$n.enfant)
12. levels(smp$n.enfant.cat)[6:13] <- "5+"
13. smp$age.cat <- cut(smp$age, breaks=c(19, 25, 35, 45, 83), include.lowest=TRUE)
14.
15. ## Sauvegarde au format R Data
16. save(smp, file="smp_v1.rda")
```

III - Lab 2 : Indexation de données et graphiques univariés

Installation de package

Les options par défaut (répertoire où installer le package et installation des dépendances) peuvent être conservées lors de l'installation d'un nouveau package. En cas d'installation multiple, il suffit d'indiquer les noms des packages séparés par des virgules ou des espaces. La commande équivalente en ligne de commande est, par exemple, install.packages("knitr").



III-A - Installation de packages

Un package R regroupe un ensemble de commandes relatives à un domaine particulier d'analyse statistique (représentations graphiques, manipulation de données, modèles spécifiques, etc.). R dispose d'un certain nombre de packages de base qui sont installés lors de l'installation de R lui-même. Il est possible d'installer des packages additionnels depuis le site principal CRAN (5) à partir de **Rstudio**.

Le chargement d'un package s'effectue à l'aide de la commande library(), en indiquant le nom du package (sans quotes). Par exemple, pour charger le package **MASS**, on tapera :

```
1. library(MASS)
```

Pour la génération de rapports automatiques, le package **knitr** (6) sera installé, avec ses dépendances. On veillera à ce que ce soit bien cet utilitaire qui soit activé par défaut pour la gestion des rapports au format R Markdown dans le menu Préférences de **RStudio**, sous l'onglet Sweave (Figure 2).

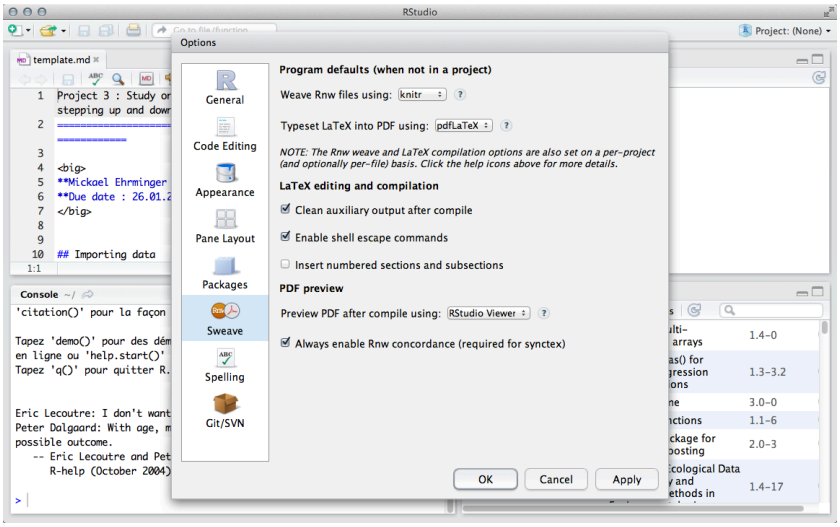


Figure 2 – Configuration requise pour générer des rapports HTML à partir du langage R Markdown.

III-B - Sélection et indexation d'observations

Pour accéder à une variable contenue dans un data frame, on utilise son nom préfixé du symbole $ et du nom du data frame.

```
> head(smp$age)
[1] 31 49 50 47 23 34
```

Une notation équivalente consiste à utiliser le numéro de position de la variable parmi les colonnes du data frame (ici, la variable age se trouve dans la première colonne) ou d'utiliser le nom de la variable (entre quotes), comme ci-dessous.

```
smp[,1]
smp[,"age"]
```

Considérons la variable prof qui indique la profession des répondants. Si l'on ne s'intéresse qu'aux individus dont la profession est « agriculteur », on peut utiliser un filtre logique avec l'opérateur == (égalité logique) pour sélectionner les observations remplissant cette condition. On pourra dresser un tableau d'effectifs en utilisant le même principe via table().

```
> head(smp$prof == "agriculteur")
[1] FALSE    NA FALSE FALSE FALSE FALSE

> table(smp$prof == "agriculteur")
FALSE  TRUE
  787     6
```

La commande which() permet de renvoyer les numéros d'observations (lignes du tableau) remplissant la condition.

```
> which(smp$prof == "agriculteur")
[1]  15 312 384 391 439 442
```

Une telle instruction peut être utilisée pour indexer les observations d'une autre variable, par exemple l'âge des personnes dont la profession est « agriculteur » :

```
> smp$age[which(smp$prof == "agriculteur")]
[1] 64 42 37 36 35 79
```

Toutefois, R offre une commande plus intéressante qui permet de sélectionner à la fois des observations remplissant une ou plusieurs conditions, et un sous-ensemble de variables. Il s'agit de la commande subset() qui prend comme premier argument le nom du data frame sur lequel on souhaite opérer, comme deuxième argument un ou plusieurs filtres logiques à appliquer aux lignes, et comme troisième argument le numéro ou le nom des variables à sélectionner.

```
> subset(smp, prof == "agriculteur", age)
     age
 15   64
312   42
384   37
391   36
439   35
442   79
```

Il est possible de sélectionner plusieurs variables en indiquant une étendue de position (par exemple, les colonnes 1 à 5) ou une liste de noms de variables, construite à l'aide de la commande c().

```
> ## subset(smp, prof == "agriculteur", 1:5)
> names(smp)[1:5]
[1] "age"      "prof"     "duree"    "discip"   "n.enfant"

> subset(smp, prof == "agriculteur", c(age, prof, duree, discip, n.enfant))
```

```
6.     age        prof duree discip n.enfant
7.  15  64 agriculteur    NA       0         3
8. 312  42 agriculteur     4       0         3
9. 384  37 agriculteur     5       1         2
10. 391  36 agriculteur     4       1         3
11. 439  35 agriculteur     3       0         0
12. 442  79 agriculteur     5       0         5
```

Les filtres sur les observations peuvent être multiples et on utilisera le symbole & pour désigner une conjonction (« et » logique) de critères de sélection des lignes du data frame.

```
1. > subset(smp, prof == "agriculteur" & n.enfant > 2,
2. >        c(age, prof, duree, discip, n.enfant))
3.     age        prof duree discip n.enfant
4.  15  64 agriculteur    NA       0         3
5. 312  42 agriculteur     4       0         3
6. 391  36 agriculteur     4       1         3
7. 442  79 agriculteur     5       0         5
```

Enfin, voici un exemple de sélection multiple de colonnes pour des individus remplissant trois conditions : « être agriculteur », « avoir plus de deux enfants » et « aucune donnée manquante sur la variable duree ».

```
1. > subset(smp, prof == "agriculteur" & n.enfant > 2 & complete.cases(duree),
2. >        c(age, prof, duree, discip, n.enfant))
3.     age        prof duree discip n.enfant
4. 312  42 agriculteur     4       0         3
5. 391  36 agriculteur     4       1         3
6. 442  79 agriculteur     5       0         5
```

III-C - Tableau d'effectifs et de fréquences relatives

Le tableau d'effectifs pour la variable n.enfant.cat construite **ici** peut être stocké directement dans une variable auxiliaire, tab.

```
1. tab <- table(smp$n.enfant.cat)
```

À partir du moment où le tableau d'effectifs a été sauvegardé dans une variable, il devient possible d'effectuer un certain nombre d'opérations élémentaires comme, par exemple, compter le nombre total d'observations à l'aide de la commande sum(). Les opérations arithmétiques s'effectuant élément par élément sous R, il est même possible d'obtenir les fréquences relatives par simple division des éléments de tab avec le total sum(tab) (7) .

```
1. > sum(tab)
2. [1] 773
3.
4. > tab / sum(tab)
5.       0       1       2       3       4      5+
6. 0.27684 0.28461 0.16171 0.13066 0.07115 0.07503
```

Cependant, il existe une commande dont le rôle est précisément de fournir les fréquences relatives, prop.table(), et qui se révélera beaucoup plus utile dans le cas des tableaux à deux entrées.

```
1. > prop.table(tab)
2.       0       1       2       3       4      5+
3. 0.27684 0.28461 0.16171 0.13066 0.07115 0.07503
```

Notons également qu'il est possible de limiter l'affichage des décimales à l'aide de round(), en précisant le nombre de décimales à afficher en deuxième argument.

```
1. > round(prop.table(tab), 3)
2.     0     1     2     3     4    5+
3. 0.277 0.285 0.162 0.131 0.071 0.075
```

Si au lieu des fréquences relatives on souhaite afficher des pourcentages, on multipliera simplement les fréquences relatives renvoyées par prop.table() par 100.

```
1. round(prop.table(tab)*100, 1)
```

III-D - Diagramme en barres

Visualisateur de graphiques

Le panneau de visualisation des graphiques dispose d'un outil d'historique qui permet de naviguer entre les différents graphiques générés, et d'une fonction d'exportation des graphiques au format PDF ou PNG.

La commande barplot() permet de construire des diagrammes en barres (encore appelés diagrammes en bâtons) pour résumer la distribution d'effectifs observée pour une variable catégorielle à k modalités. On fournit à cette commande non pas une variable, mais directement un tableau d'effectifs ou de fréquences construit avec table(). Les principaux paramètres graphiques permettant de personnaliser le rendu final de la figure 3 représentant la distribution du nombre d'enfants sont illustrés dans l'instruction suivante.

```
1. barplot(prop.table(tab) * 100, ylab="Proportion", col="cornflowerblue",
2.         border=NA, ylim=c(0, 30), las=1)
```

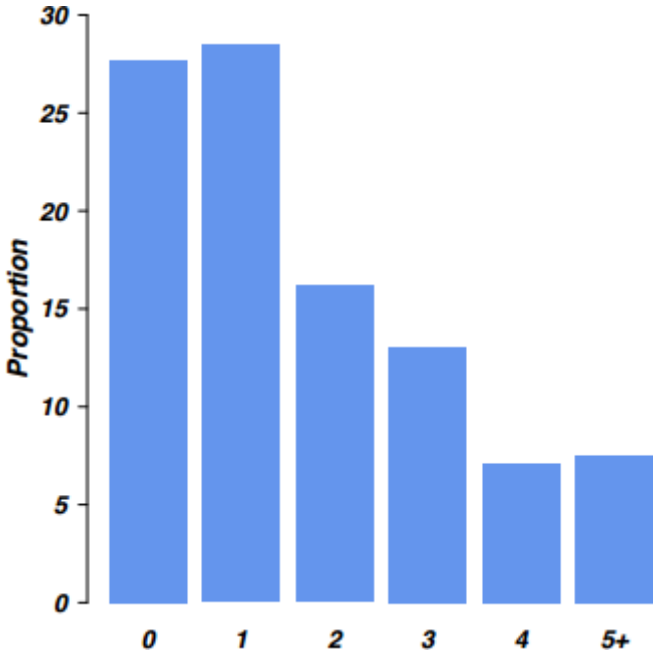


Figure 3 – Diagramme en barres.

Il est tout à fait possible d'utiliser un diagramme en points dotplot(), en utilisant à peu près la même syntaxe que celle présentée ci-dessus.

III-E - Histogramme

La commande hist() permet de représenter la distribution d'une variable numérique (continue ou discrète) sous forme d'effectif ou de densité. Le nombre de classes, ainsi que la largeur des intervalles de classe, peuvent être paramétrés à l'aide des options nclass= et breaks=.

La figure 4 montre une application de cette commande pour la variable age. Notons qu'une courbe de densité a été ajoutée à l'histogramme. Pour cela, on utilise la commande lines() qui permet de travailler sur la même fenêtre graphique et superposer plusieurs éléments graphiques (8) . Le nombre total d'observations a été également ajouté au graphique, à l'aide de paste() qui permet de juxtaposer du texte libre et, par exemple, des expressions R.

```
1. hist(smp$age, nclass=8, prob=TRUE, col="cornflowerblue", border="white",
2.      xlim=c(0,100), main="", xlab="Âge (années)", ylab="Densité")
3. lines(density(smp$age, na.rm=TRUE), lwd=2, col="orange")
4. text(65, 0.025, paste("N =", sum(complete.cases(smp$age))), cex=1.2)
```

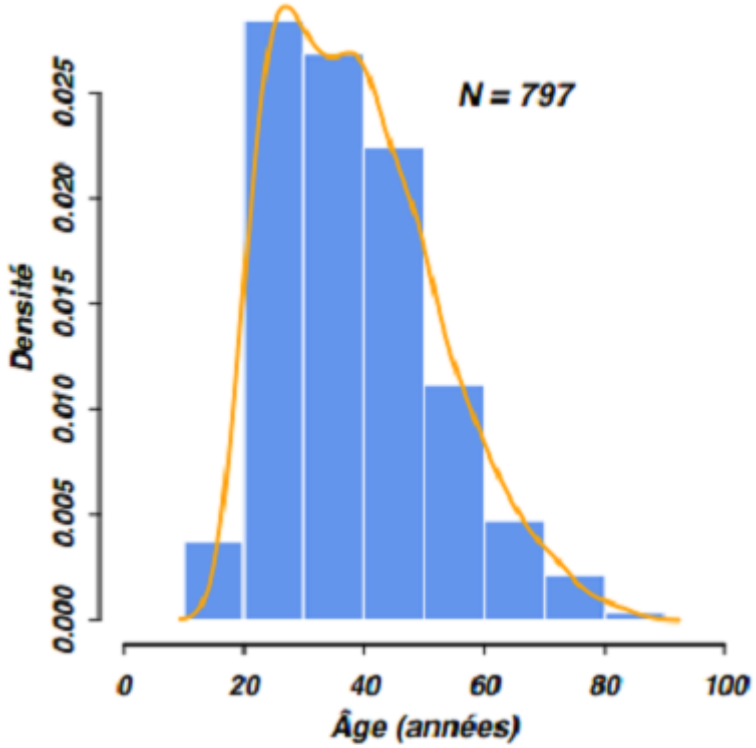


Figure 4 – Histogramme.

IV - Lab 3 : Mesures et tests d'association

IV-A - Tableau de contingence et test du chi-deux

La commande table() permet de construire un tableau d'effectifs pour une variable qualitative, ou pour le croisement des modalités de deux variables qualitatives. Un tableau de contingence pour les variables subst.cons et abus peut être construit à l'aide de l'instruction suivante, en se rappelant que la première variable mentionnée dans la commande sera celle dont les modalités apparaîtront sous forme de lignes dans le tableau.

```
1. > table(smp$subst.cons, smp$abus)
2.     0   1
3. 0 441 140
4. 1 131  80
```

Si l'on stocke ce tableau de contingence dans une variable auxiliaire, tab, il est ensuite possible d'utiliser d'autres commandes pour travailler avec ce tableau, par exemple prop.table() discutée **ici**. Dans le cas d'un tableau à deux entrées, les fréquences relatives affichées sont calculées par rapport à l'effectif total.

```
1. > tab <- table(smp$subst.cons, smp$abus)
2. > prop.table(tab)
3.        0      1
4. 0 0.5568 0.1768
```

```
5. 1 0.1654 0.1010
```

Il est possible de spécifier une option margin= pour indiquer à prop.table() comment calculer les fréquences relatives : avec margin=1, R calcule les fréquences relatives conditionnellement aux totaux lignes (somme des effectifs par modalités de la première variable), tandis qu'avec margin=2 les effectifs sont rapportés aux totaux colonnes.

```
1. > prop.table(tab, margin=1)
2.        0      1
3. 0 0.7590 0.2410
4. 1 0.6209 0.3791
5.
6. > prop.table(tab, margin=2)
7.        0      1
8. 0 0.7710 0.6364
9. 1 0.2290 0.3636
```

Il est également possible d'utiliser la commande xtabs() à la place de table().

L'avantage de cette commande est qu'elle repose sur l'usage de formules R qui permettent, comme on le verra plus loin, de décrire les relations entre différentes variables. Dans le cas où les deux variables jouent un rôle symétrique (c'est-à-dire que l'on ne fait pas de distinction entre une variable réponse et une variable explicative), la formule à utiliser est de la forme ~ x + y, où x et y sont les deux variables d'intérêt. Il est nécessaire d'indiquer le nom du data frame dans lequel trouver les variables. On remarquera également que R affiche désormais le nom des variables.

```
1. > xtabs(~ subst.cons + abus, data=smp)
2.            abus
3. subst.cons   0   1
4.          0 441 140
5.          1 131  80
```

La commande barplot() permet également de représenter la distribution d'effectifs (ou de fréquences) lorsque l'on s'intéresse à la relation entre deux variables qualitatives. Pour juxtaposer les barres, on prendra garde au fait qu'il faut rajouter l'option **beside=TRUE**. Enfin, dans le cas où l'on travaille avec plusieurs variables, il est nécessaire d'utiliser une légende pour indiquer les modalités de la variable illustrative, dans le cas présent celle dont les modalités sont affichées avec une couleur différente par R.

```
1. barplot(tab, beside=TRUE, legend=c("abus = 0", "abus = 1"),
2.         xlab="Consommation substance", ylab="Effectifs",
3.         col=c("cornflowerblue", "deepskyblue4"))
```

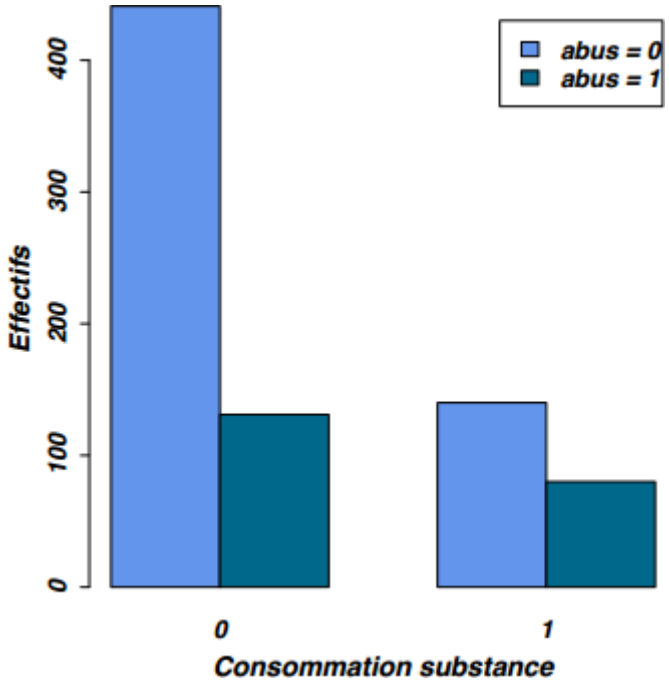


Figure 5 – Diagramme en barres pour deux variables qualitatives.

Le test du chi-deux est disponible via la commande chisq.test(), et on lui fournit simplement un tableau de contingence (9) .

```
> chisq.test(tab)

Pearson's Chi-squared test with Yates' continuity correction

data: tab
X-squared = 14.05, df = 1, p-value = 0.0001779
```

Si l'on stocke le résultat du test dans une variable auxiliaire, il est possible d'obtenir des informations supplémentaires comme, par exemple, les effectifs théoriques (effectifs attendus sous l'hypothèse d'indépendance entre les deux variables). On procédera de la même manière que l'on adresse une variable dans un data frame, en indiquant le nom de la variable suivi du symbole $ et du nom de l'objet d'intérêt, dans ce cas *expected*.

```
> res <- chisq.test(tab)
> res$observed
    0   1
0 441 140
1 131  80

> res$expected
        0      1
0 419.6 161.39
1 152.4    58.
```

Le test de Fisher est quant à lui disponible via la commande fisher.test().

```
> fisher.test(tab)

Fisher's Exact Test for Count Data

data: tab
p-value = 0.0002193
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:

 1.351 2.728
sample estimates:
```

```
12. odds ratio
13.       1.922
```

IV-B - Comparaison de deux moyennes

Considérons les variables age et subst.cons (variable binaire). Dans les deux cas, on obtient le résumé des distributions avec summary(), ou table() dans le cas de la variable qualitative.

```
1. > summary(smp$age)
2. Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
3. 19.0    28.0    37.0    38.9    48.0    83.0       2
4.
5. > table(smp$subst.cons)
6.   0   1
7. 587 212
```

Plutôt que d'utiliser des histogrammes pour représenter la distribution des âges selon le fait que les individus consomment des substances ou n'en consomment pas, il est tout à fait possible d'utiliser des courbes de densité, qui présentent d'ailleurs l'avantage de pouvoir être superposées dans le même graphique.

```
1. plot(density(smp$age[smp$subst.cons == 1], na.rm=TRUE), lwd=2,
2.      col="orange", xlim=c(19,83), main="")
3. lines(density(smp$age[smp$subst.cons == 0], na.rm=TRUE), lwd=2,
4.       col="deepskyblue4")
```

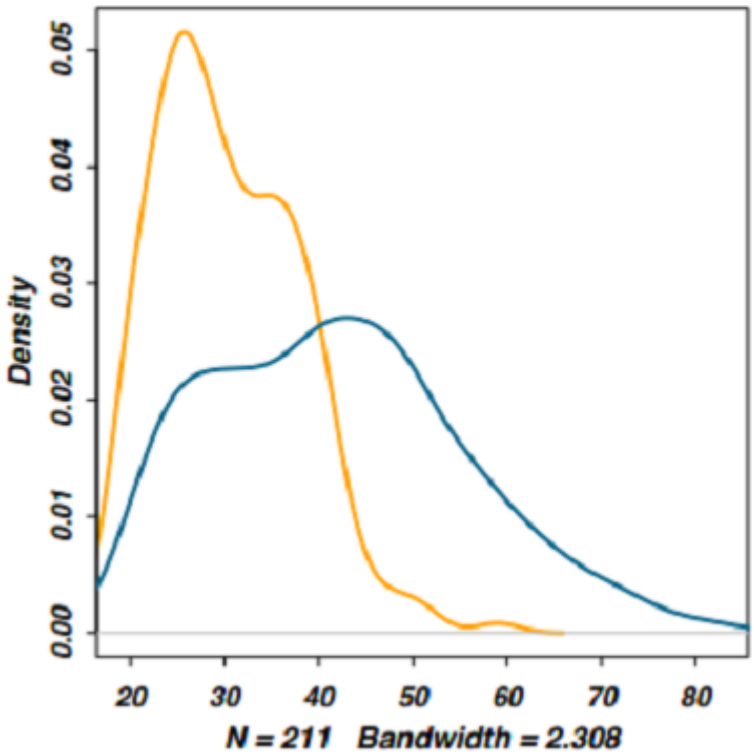


Figure 6 – Courbes de densité pour une variable numérique selon les niveaux d'une variable qualitative

Pour calculer l'âge moyen selon les niveaux de la variable subst.cons, il serait possible d'utiliser une expression de la forme mean(smp$age[smp$subst.cons == 0], na.rm=TRUE) et mean(smp$age[smp$subst.cons == 1], na.rm=TRUE), afin d'isoler les deux groupes d'individus comme dans le cas des commandes graphiques précédentes.

Toutefois, avec R il existe plusieurs commandes permettant d'appliquer une même commande à des sous-ensembles d'observations définis par une variable de classification (facteur). Outre la commande by(), il existe tapply() qui attend comme arguments : la variable numérique à laquelle on souhaite appliquer une commande (ici, age), le facteur

permettant de définir les groupes à partir de ses niveaux (subst.cons), et la commande (mean), suivie éventuellement de ses propres options (na.rm=**TRUE**). Voici un exemple d'application :

```
> tapply(smp$age, smp$subst.cons, mean, na.rm=TRUE)
     0      1
41.97 30.37
```

Une formulation alternative consiste à utiliser l'expression with(smp, ...) qui permet de désigner le data frame dans lequel opérer et évite d'avoir à préfixer systématiquement les variables par smp$.

```
with(smp, tapply(age, subst.cons, mean, na.rm=TRUE))
```

Le test de Student est accessible via la commande t.test(). On prendra garde au fait que, par défaut, R propose un test modifié de Welch (correction de Satterthwaite (10)).

Ci-dessous, on utilise t.test() en fournissant deux échantillons indépendants, définis selon la valeur prise par la variable subst.cons. On verra ensuite une formulation équivalente et plus économique à l'aide de formules R.

```
> t.test(smp$age[smp$subst.cons == 0], smp$age[smp$subst.cons == 1])

Welch Two Sample t-test

data: smp$age[smp$subst.cons == 0] and smp$age[smp$subst.cons == 1]
t = 15.24, df = 666.8, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 10.11 13.10
sample estimates:
mean of x mean of y
    41.97     30.37
```

Dans un premier temps, on peut tout à fait convertir la variable subst.cons en facteur et lui associer des étiquettes Oui/Non.

```
> smp$subst.cons <- factor(smp$subst.cons, levels=c(0, 1),
>                          labels=c("Non", "Oui"))
> table(smp$subst.cons)
Non Oui
587 212
```

La commande aggregate() fonctionne de la même manière que tapply() (appliquer une commande à une variable numérique pour chaque niveau d'une variable qualitative), à ceci près qu'elle accepte également une notation par formule, comme l'illustre l'exemple suivant.

```
> aggregate(age ~ subst.cons, data=smp, mean)
  subst.cons   age
1        Non 41.97
2        Oui 30.37
```

Notons que le résultat renvoyé par R est un data frame, contrairement à tapply(). On peut évidemment remplacer mean par n'importe quelle autre commande R, par exemple var pour calculer les variances dans chacun des deux groupes.

```
aggregate(age ~ subst.cons, data=smp, var)
```

Enfin, cette formule reste applicable dans le cas des représentations graphiques. Par exemple, pour tracer des diagrammes de type boîtes à moustaches pour résumer la distribution des âges selon les niveaux de subst.cons, on utilisera exactement la même formule avec la commande boxplot().

```
boxplot(age ~ subst.cons, data=smp,
```

```
        xlab="Consommation substance", ylab=" Âge (années)",
        col="cornflowerblue", border="cornflowerblue")
```

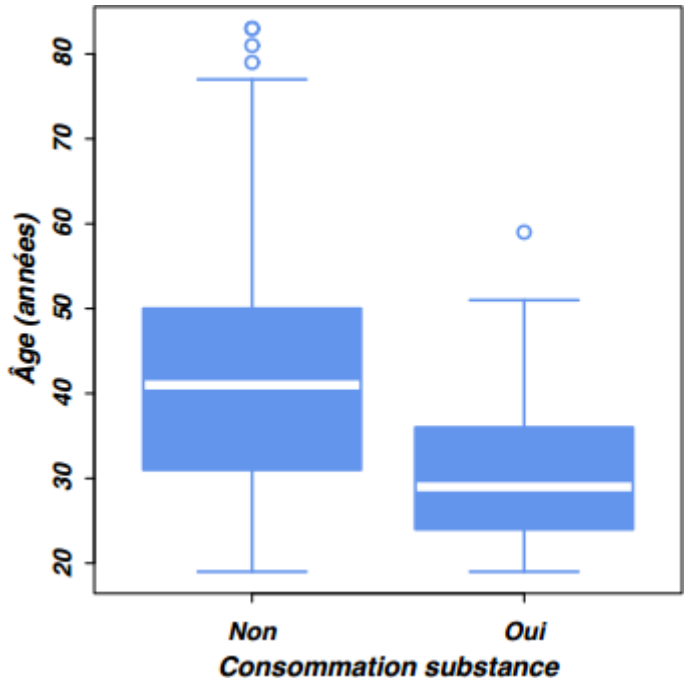


Figure 7 – Diagrammes de type boîtes à moustaches.

Les notations par formule présentent l'avantage de faire clairement apparaître le rôle joué par les variables, ou plus généralement leur relation : la variable réponse est placée à gauche du symbole ~ et la variable explicative à droite, de sorte qu'une formule telle que age ~ subst.cons se lit « la variable age expliquée par la variable subst.cons ».

Dans le cas du test de Student, cette notation par formule prend tout son sens. On ajoutera l'option var.equal=**TRUE** pour réaliser un test de Student classique, supposant l'homogénéité des variances parentes.

```
> ## t.test(age ~ subst.cons, data=smp)
> t.test(age ~ subst.cons, data=smp, var.equal=TRUE)

Two Sample t-test

data: age by subst.cons
t = 11.79, df = 795, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  9.669 13.534
 sample estimates:
mean in group Non mean in group Oui
            41.97             30.37
```

À l'aide de la commande plotmeans() du package **gplots**, il est possible de représenter graphiquement les moyennes de groupes et leurs intervalles de confiance associés, toujours en utilisant une formule décrivant la relation entre les deux variables d'intérêt.

```
library(gplots)
plotmeans(age ~ subst.cons, data=smp, col="cornflowerblue",
          barcol="cornflowerblue", pch=19,
          xlab="Consommation de substance", ylab=" Age (années)")
```

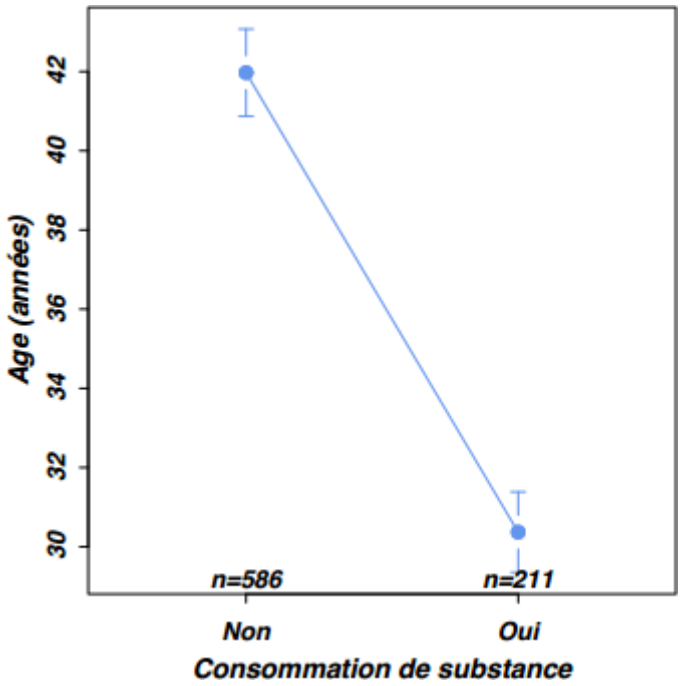


Figure 8 – Utilisation de commandes graphiques intégrées du package gplots.

Le test de Wilcoxon se réalise à l'aide de la commande wilcox.test().

```
> wilcox.test(age ~ subst.cons, data=smp)

Wilcoxon rank sum test with continuity correction

data: age by subst.cons
W = 93984, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

IV-C - Corrélation linéaire

Pour calculer le coefficient de corrélation linéaire de Pearson, on utilisera la commande cor(). Si les variables présentent des valeurs manquantes, il faudra explicitement demander à R d'effectuer le calcul sur l'ensemble des paires de valeurs complètes à l'aide de l'option use="pairwise". Voici un exemple d'utilisation avec les variables n.enfant et age.

```
> cor(smp$n.enfant, smp$age)
[1] NA

> cor(smp$n.enfant, smp$age, use="pairwise")
[1] 0.4326
```

Le coefficient de corrélation de rangs de Spearman est quant à lui obtenu en ajoutant l'option method="spearman" à la commande précédente.

```
> cor(smp$n.enfant, smp$age, use="pairwise", method="spearman")
[1] 0.4244
```

La commande cor.test() renvoie en plus du test de nullité du coefficient de corrélation un intervalle de confiance (par défaut à 95 %) et la valeur du coefficient de corrélation.

```
> cor.test(smp$n.enfant, smp$age)

Pearson's product-moment correlation

```

```
data: smp$n.enfant and smp$age
t = 13.32, df = 771, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3735 0.4882
sample estimates:
    cor
0.4326
```

Si l'on préfère utiliser une notation par formule, on prendra garde au fait que puisque les variables jouent un rôle symétrique (il n'y a pas vraiment de variable réponse dans une approche par corrélation), les deux variables sont placées à droite du symbole ~.

```
cor.test(~n.enfant + age, data=smp)
```

Il est tout à fait possible d'ajuster le niveau de confiance désiré ($1 - \alpha$) pour calculer l'intervalle de confiance via l'option conf.level.

```
> cor.test(~ n.enfant + age, data=smp, conf.level=.90)

Pearson's product-moment correlation

data: n.enfant and age
t = 13.32, df = 771, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
90 percent confidence interval:
 0.3832 0.4795
sample estimates:
    cor
0.4326
```

Il est recommandé d'inspecter graphiquement les données avant d'interpréter toute mesure d'association linéaire entre deux variables. En l'occurrence, un diagramme de dispersion peut être construit à l'aide de la commande plot() et d'une formule décrivant la relation à représenter : la variable apparaissant à gauche du symbole ~ sera représentée sur l'axe vertical (ordonnées), et la variable apparaissant à gauche de ~ figurera sur l'axe horizontal (abscisses).

Comme il y a beaucoup de données plus ou moins superposées, on utilisera le paramètre col= pour manipuler la transparence des points à l'aide de la commande rgb() dont la quatrième valeur représente le degré de transparence (0 = transparent, 1 = opaque).

Pour mieux apprécier les écarts locaux à la linéarité de la relation entre les deux variables, une courbe de type loess peut être superposée sur le graphique. La commande lowess() (11) utilisée ci-dessous suppose que le package **gplots** a été chargé au préalable (library(gplots)).

```
plot(n.enfant ~ age, smp, pch=19, col=rgb(0.39,0.58,0.93,0.35),
     xlab="Âge (années)", ylab="Nombre d'enfants")
lines(lowess(n.enfant ~ age, smp), col="orange", lwd=2) ## package gplots
grid()
```

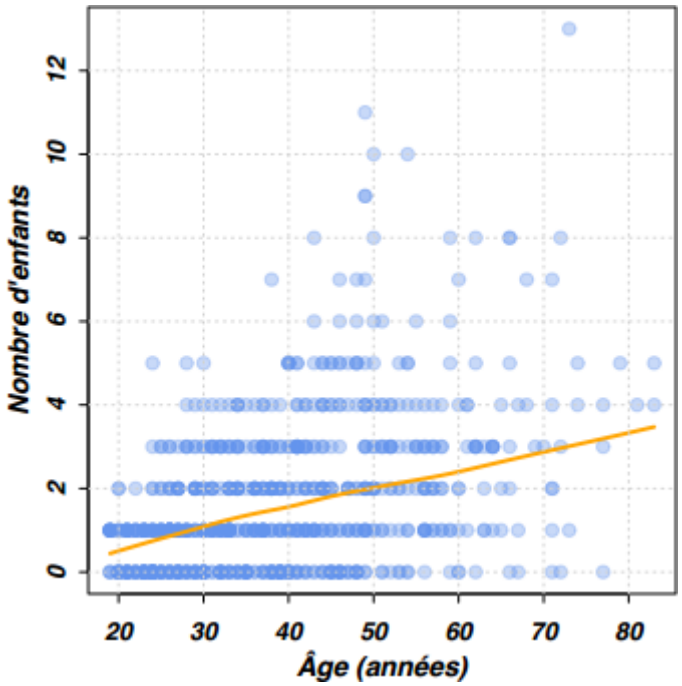


Figure 9 – Utilisation de commandes graphiques intégrées du package gplots.

V - Lab 4 : Modèles statistiques

V-A - Sélection critériée multiple

On a vu que la commande subset() permettait de sélectionner des observations remplissant une certaine condition, et éventuellement de ne renvoyer que certaines variables du data frame de travail (**Sélection et indexation d'observations**).

Supposons que l'on souhaite sélectionner tous les individus dont la profession figure parmi la liste suivante : « sans emploi », « profession intermédiaire », « cadre ». Il est possible de combiner ses différentes modalités à l'aide de l'opérateur | (« ou » logique), comme ci-dessous :

```
1. head(subset(smp, prof == "sans emploi" | prof == "prof.intermediaire" |
2.      prof == "cadre", c(age, n.enfant, prof)))
```

Notons que dans l'exemple précédent, seules trois variables sont sélectionnées, et le résultat est un data frame. Si la liste de modalités d'intérêt est grande, cela devient vite fastidieux et il est préférable d'utiliser la notation %in% qui permet d'indiquer les conditions à remplir sous forme d'une simple liste.

```
1. > head(subset(smp, prof %in% c("sans emploi", "prof.intermediaire", "cadre"),
2. >      c(age, n.enfant, prof)))
3.    age n.enfant              prof
4.  3  50        2 prof.intermediaire
5.  5  23        1        sans emploi
6. 11  31        0 prof.intermediaire
7. 17  60        2 prof.intermediaire
8. 23  32        0        sans emploi
9. 27  32        1        sans emploi
```

Il est d'ailleurs possible de sauvegarder ce nouveau data frame dans une variable appelée, par exemple, smpb pour faciliter les opérations suivantes.

```
1. > smpb <- subset(smp, prof %in% c("sans emploi", "prof.intermediaire", "cadre"),
2. >                c(age, n.enfant, prof))
3. > summary(smpb)
```

```
     age n.enfant prof
Min.   :19.0   Min.   : 0.00   sans emploi       :222
1st Qu.:27.0   1st Qu.: 0.00   prof.intermediaire: 58
Median :36.0   Median : 1.00   cadre             : 24
Mean   :38.4   Mean   : 1.65   agriculteur       :  0
3rd Qu.:47.2   3rd Qu.: 3.00   artisan           :  0
Max.   :83.0   Max.   :13.00   autre             :  0
               NA's   :11      (Other)           :  0
```

On remarque toutefois que la variable prof n'a pas uniquement trois niveaux, comme on aurait pu s'y attendre, mais que les anciens niveaux de la variable ont été conservés, même s'ils ne sont plus observés dans le jeu de données réduit.

Pour mettre à jour la liste des niveaux de cette variable, il est nécessaire de réutiliser la commande factor() afin que R identifie correctement les modalités uniques de prof.

```
> smpb$prof <- factor(smpb$prof, labels=c("cadre", "intermédiaire", "sans emploi"))
> table(smpb$prof)
     cadre intermédiaire sans emploi
        24            58         222
```

V-B - Analyse de variance à un facteur

Intéressons-nous au nombre d'enfants en fonction de la catégorie socioprofessionnelle, restreinte aux trois modalités discutées ci-dessus. Pour calculer le nombre moyen d'enfants dans chaque groupe, on peut utiliser aggregate() avec la commande mean() en troisième argument. De même, il serait possible de vérifier la valeur des variances, ou n'importe quelle autre quantité, en adaptant la commande à appliquer à la variable n.enfant selon les niveaux de prof.

```
> aggregate(n.enfant ~ prof, data=smpb, mean)
            prof n.enfant
1         cadre    2.167
2 intermédiaire    2.107
3   sans emploi    1.469
```

En termes de représentation graphique, un diagramme de type boîte à moustaches permettra d'apprécier la distribution des données dans chaque groupe. La commande boxplot() repose sur l'usage de la même formule que aggregate() permettant de décrire la relation entre les deux variables.

```
boxplot(n.enfant ~ prof, data=smpb,
        xlab="Profession", ylab="Nombre d'enfants",
        col="cornflowerblue", border="cornflowerblue")
```

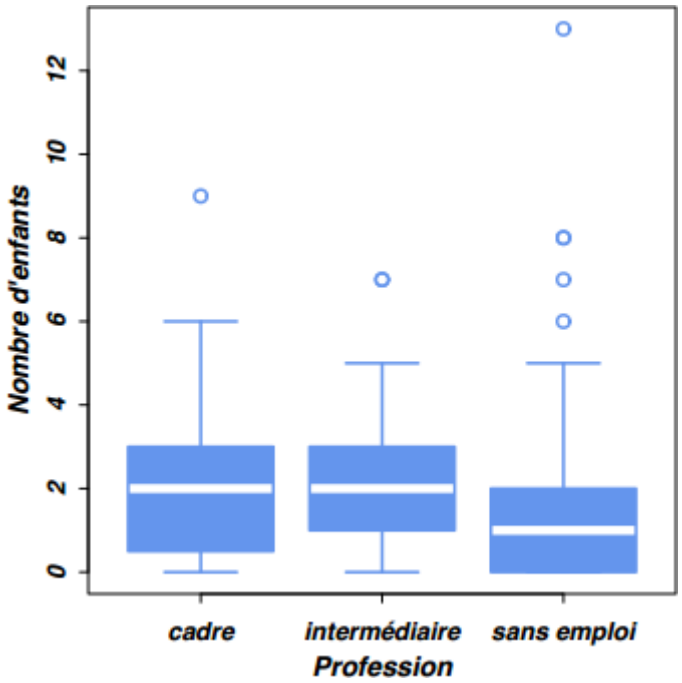


Figure 10 – Diagrammes de type boîte à moustaches

Pour réaliser le modèle d'analyse de variance (ANOVA) à un facteur, on utilisera directement la commande lm(), qui est également la commande à utiliser dans le cadre de la régression linéaire (**Régression linéaire**). La formule R utilisée permet de décrire les variations de la variable réponse (ici, n.enfant) en fonction de la variable explicative, de type facteur (ici, prof). Par contre, lm() dispose d'une option subset=, ce qui permet de travailler directement avec le data frame d'origine, smp, et d'indiquer les professions sur lesquelles on souhaite se concentrer via cette option (plutôt que d'utiliser le data frame intermédiaire smpb).

```
> m <- lm(n.enfant ~ prof, data=smp,
>         subset=prof %in% c("sans emploi", "prof.intermediaire", "cadre"))
> m

Call:
lm(formula = n.enfant ~ prof, data = smp, subset = prof %in%
    c("sans emploi", "prof.intermediaire", "cadre"))

Coefficients:
         (Intercept)  profprof.intermediaire
              2.1667                 -0.0595
    profsans emploi
             -0.6972
```

Par défaut, le résultat retourné par R inclut uniquement les coefficients de régression calculés selon la méthode de codage par contrastes de traitement pour les modalités de la variable qualitative : chaque coefficient représente la déviation moyenne par rapport à la catégorie de référence, dont la moyenne est représentée par le terme d'ordonnée à l'origine (« Intercept »), qui est toujours sous R le premier niveau du facteur (ici, « cadre »). Pour obtenir le tableau d'ANOVA, on utilisera drop1() en précisant le type de test à réaliser, test="F" signifiant un test de Fisher-Snedecor.

```
> drop1 (m, test="F")

Single term deletions

Model:
n.enfant ~ prof
       Df Sum of Sq RSS AIC F value Pr(>F)
<none>              948 350
prof    2      25.1 973 354    3.83  0.023 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Il est également possible d'utiliser la commande anova(), mais le calcul des sommes de carré n'est pas tout à fait le même (12) .

```
> anova(m)

Analysis of Variance Table

Response: n.enfant
           Df Sum Sq Mean Sq F value Pr(>F)
prof        2     25   12.52    3.83 0.023 *
Residuals 290    948    3.27
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

V-C - Régression linéaire

La commande lm() permet également de réaliser des modèles de régression linéaire simple ou multiple.

Si l'on s'intéresse à la relation entre le nombre d'enfants (variable réponse, n.enfant) et l'âge des répondants (variable explicative, age), la commande suivante permettra de fournir les coefficients de la droite de régression. Notons que comme dans le cas de l'ANOVA, il est plus pratique de stocker le résultat de lm() dans une variable auxiliaire afin de pouvoir appliquer d'autres commandes sur le résultat du modèle de régression. En utilisant summary(), on obtient directement le tableau des coefficients de régression avec leur test de significativité associé.

```
> m <- lm(n.enfant ~ age, data=smp)
> summary(m)

Call:
lm(formula = n.enfant ~ age, data = smp)

Residuals:
   Min     1Q Median    3Q   Max
-4.03  -1.03  -0.23  0.69  9.21

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.5898     0.1858   -3.17   0.0016 **
age           0.0600     0.0045   13.32   <2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.65 on 771 degrees of freedom
   (26 observations deleted due to missingness)
Multiple R-squared: 0.187,Adjusted R-squared: 0.186
F-statistic: 178 on 1 and 771 DF, p-value: <2e-16
```

Il est possible d'accéder aux coefficients individuellement à l'aide de la commande coef(). Par exemple, coef(m)[2] ou coef(m)["age"] renverra la pente de la droite de régression.

```
> coef(m)
(Intercept)         age
    -0.5898      0.0600

> coef(m)[2]
  age
0.06

## coef(m)["age"]
```

Si l'on souhaite obtenir les intervalles de confiance (par défaut, à 95 % (13)) pour les coefficients de régression, on utilisera confint() en indiquant le nom de la variable dans lequel on a stocké les résultats du modèle.

```
> confint(m)
                2.5%     97.5%
```

```
3. (Intercept) -0.95457 -0.22503
4.         age  0.05116  0.06884
```

Pour obtenir le tableau d'ANOVA associé au modèle de régression, on utilisera la commande anova(), comme ci-dessous.

```
1. > anova(m)
2.
3. Analysis of Variance Table
4.
5. Response: n.enfant
6.            Df Sum Sq Mean Sq F value Pr(>F)
7. age         1    486     486     178 <2e-16 ***
8. Residuals 771   2111       3
9. ---
10. Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Les valeurs ajustées (nombres d'enfants prédits par le modèle pour l'ensemble des âges observés) sont obtenues à l'aide de fitted(), tandis que les résidus de la régression sont obtenus via la commande resid().

```
1. > head( fitted(m))
2.      1      2      3      4      5      6
3. 1.2701 2.3500 2.4100 2.2300 0.7901 1.4501
4.
5. > head( resid(m))
6.      1      2       3       4      5      6
7. 0.7299 4.6500 -0.4100 -2.2300 0.2099 1.5499
```

On pourra d'ailleurs vérifier que les résidus correspondent bien à l'écart entre les valeurs observées et les valeurs prédites par le modèle. Par exemple, pour les deux premières observations, on obtient :

```
1. > smp$n.enfant[1:2] - fitted(m)[1:2]
2.      1      2
3. 0.7299 4.6500
```

Enfin, si l'on souhaite prédire le nombre d'enfants pour des valeurs de age non nécessairement observés dans le jeu de données, on utilisera la commande plus générique predict(), en spécifiant dans un data frame les valeurs d'intérêt. Notons qu'avec l'option interval="confidence", un intervalle de confiance à 95 % pour la prédiction est fourni automatiquement par R.

```
1. > predict(m, data.frame(age=c(20, 30, 40)), interval="confidence")
2.      fit    lwr    upr
3. 1 0.6101 0.4049 0.8154
4. 2 1.2101 1.0683 1.3519
5. 3 1.8101 1.6930 1.9272
```

V-D - Régression logistique

Considérons toujours le nombre d'enfants comme variable réponse, mais cette fois-ci sous la forme d'une variable binaire.

Les instructions suivantes permettent, grâce à l'instruction ifelse(), de créer une variable n.enfant.bin valant 1 si le nombre d'enfants est strictement supérieur à 2, et 0 autrement.

```
1. > smp$n.enfant.bin <- factor(ifelse(smp$n.enfant > 2, 1, 0))
2. > table(smp$n.enfant.bin)
3.   0   1
4. 559 214
```

Pour réaliser une régression logistique en considérant n.enfant.bin comme variable réponse et age comme variable explicative, on utilisera la commande glm() qui fonctionne de la même manière que lm(), mais à laquelle il faudra préciser l'option family=binomial("logit"), ou family=binomial puisque logit est l'option par défaut.

Le tableau affiché par l'instruction summary(m) ci-après fournit les valeurs des coefficients de régression exprimés sur l'échelle du log-odds.

```
> m <- glm(n.enfant.bin ~ age, data=smp, family=binomial("logit"))
> summary(m)

Call:
glm(formula = n.enfant.bin ~ age, family = binomial("logit"),
data = smp)

Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.855  -0.752  -0.533   0.876   2.130

Coefficients:
             Estimate Std. Error  z value Pr(>|z|)
(Intercept)  -3.82709    0.31280    -12.2   <2e-16 ***
age           0.06949    0.00702      9.9   <2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 912.06 on 772 degrees of freedom
Residual deviance: 794.16 on 771 degrees of freedom
  (26 observations deleted due to missingness)
AIC: 798.2

Number of Fisher Scoring iterations: 4
```

On peut toujours accéder aux coefficients individuellement à l'aide de coef(), et si l'on souhaite afficher la valeur de l'odds-ratio, on transformera le coefficient correspondant à la pente à l'aide de la commande exp().

```
> coef(m)
(Intercept)         age
   -3.82709     0.06949

> coef(m)[2]
     age
0.06949

> exp(coef(m)[2])
   age
1.072
```

Pour estimer l'odds ratio associé à une variation de 5 ans, plutôt qu'un an, on multipliera directement la valeur du coefficient avant d'en prendre l'exponentielle :

```
exp(5* coef(m)[2])
```

Enfin, comme dans le cas de la régression linéaire, il est possible d'obtenir les valeurs prédites par le modèle à partir des données observées en utilisant la commande fitted() ou predict().

Dans le cas de la régression logistique, on prendra toutefois garde au fait que, puisque dans le modèle on travaille sur l'échelle du log-odds, pour obtenir des valeurs individuelles de probabilité (ici, la probabilité d'avoir plus de deux enfants selon l'âge), il faudra ajouter l'option type="response".

```
> head(predict(m))
      1       2       3       4       5       6
-1.6730 -0.4222 -0.3527 -0.5612 -2.2289 -1.4645
```

```
> head(predict(m, type="response"))
      1       2       3       4       5       6
0.15803 0.39599 0.41272 0.36327 0.09719 0.18778
```

VI - Remerciements

Nous remercions Christophe Lalanne et Bruno Falissard qui nous ont autorisés à publier ce tutoriel.

Nous tenons également à remercier **Winjerome** pour la mise au gabarit et **f-leb** pour la relecture orthographique.

1 : **http://www.r-project.org/**
2 : **http://www.rstudio.com/**
3 : Le fait que l'on puisse appliquer une commande arithmétique telle que sum() découle du fait que R représente les valeurs FALSE comme valant 0 et les valeurs TRUE comme valant 1.
4 : Il est possible d'être plus précis et d'utiliser la commande quantile() qui permet de réaliser un quartilage de la variable, par exemple à l'aide d'une instruction telle que : cut(smp$age, breaks=quantile(smp$age), include.lowest=TRUE).
5 : **http://cran.r-project.org/**
6 : **http://yihui.name/knitr/**
7 : Attention, dans ce cas les résultats portent sur l'ensemble des données observées sans données manquantes puisque la commande table() a été utilisée sans l'option useNA="always".
8 : Une instruction telle que plot(density(…)) aurait eu pour conséquence de supprimer l'histogramme et de représenter simplement la courbe de densité dans une nouvelle fenêtre graphique.
9 : Il est également possible d'indiquer séparément les deux variables, par exemple : chisq.test(smp$subst.cons,smp$abus).
10 : **http://en.wikipedia.org/wiki/Welch-Satterthwaite_equation**
11 : Voir l'aide en ligne pour plus de détails, et les options spécifiques qui permettent de contrôler le degré de lissage de la courbe loess.
12 : La commande aov() inclut des options additionnelles pour l'analyse des plans d'expérience un peu plus complexes.
13 : Voir l'option conf.level= pour modifier le niveau de confiance.