Université de Sherbrooke
Faculté des lettres et sciences humaines
École de politique appliquée

Guide pour l'utilisation de RStudio

Par Alexandre Michel

Sherbrooke, Québec, Canada
Mis à jour le 19 août 2022à 19:10

TABLE DES MATIÈRES

1 **L'installation de RStudio** 1

1.1 *L'interface et la programmation orientée objet* 2

2 **Les opérateurs arithmétiques et logiques** 4

3 **Les fonctions** 5

3.1 *Comment définir une fonction* 6

3.2 *L'installation et l'activation des packages* 7

4 **L'importation d'une base de données** 10

4.1 *Le package WDI de la Banque Mondiale* 10

4.2 *Un fichier CSV* 11

4.3 *Un fichier Excel* 11

4.4 *Un fichier SPSS* 11

4.5 *Une base de données hébergée sur un site web* 11

4.6 *Base de données à importer pour les exemples du guide* 12

5 **Les manipulations à effectuer sur une base de données** 13

5.1 *Comment visualiser et explorer une base de données* 13

5.2 *Les informations générales sur la base* 13

5.3 *Les informations générales sur les variables* 14

5.4 *Comment renommer des variables (colonnes)* 14

5.5 *Comment recoder les valeurs d'une variable* 15

5.6 *Comment créer une nouvelle variable* 17

5.7 *Comment filtrer une base de données* 18

6 **L'inférence statistique** 20

6.1 *Historique et fondements de l'inférence* 20

6.2 *Conclusions de Pascal* 23

6.3 *Les niveaux de confiance, l'intervalle de confiance et la marge d'erreur* . . 23

7 **L'analyse univariée : les caractéristiques importantes d'une distribution** 26

7.1 *Une variable qualitative* 26

7.2 *Une variable quantitative* 28

8 **L'analyse bivariée : mesurer la relation statistique entre deux variables** 30

8.1 *Croisement entre deux variables qualitatives* 31

8.2 *Croisement entre une variable qualitative et une quantitative* 35
8.3 *Croisement entre deux variables quantitatives* 39
8.4 *Réflexions générales sur les croisements* 42

LISTE DES FIGURES

1 Première ouverture de RStudio 1
2 Codage des opérateurs arithmétiques et logiques dans R 4
3 Version originale du triangle de Pascal dans le traité de Blaise Pascal 21
4 Version épurée des six premières lignes du triangle de Pascal 21
5 Reproduction de la 10^{e} ligne du triangle de Pascal dans RStudio 22
6 Intervalle de confiance et marge d'erreur d'une distribution normale 25

1 L'installation de RStudio

Le logiciel RStudio est ce qu'on appelle un Environnement de développement intégré, ou Integrated Development Environment (IDE) en anglais. C'est un IDE qui utilise le langage de programmation R et qui l'affiche sur une interface, une fenêtre, qui offre beaucoup plus d'outils que la version standard de R. Pour installer convenablement le logiciel RStudio, il convient d'abord d'installer la version standard de R. Pour ce faire, il faut se diriger sur cette page pour Windows ou celle-ci pour Mac. Une fois les fichiers de base de R installés, on peut procéder à l'installation de RStudio ici. Vous pouvez ensuite choisir l'option de *Download* qui correspond à votre système d'exploitation (OS), donc le fichier se terminant par .exe pour Windows ou celui se terminant par .dmg pour MacOS.

Une fois l'installation terminée, vous pouvez ouvrir RStudio et vous devriez voir une fenêtre ressemblant à celle-ci :[1]

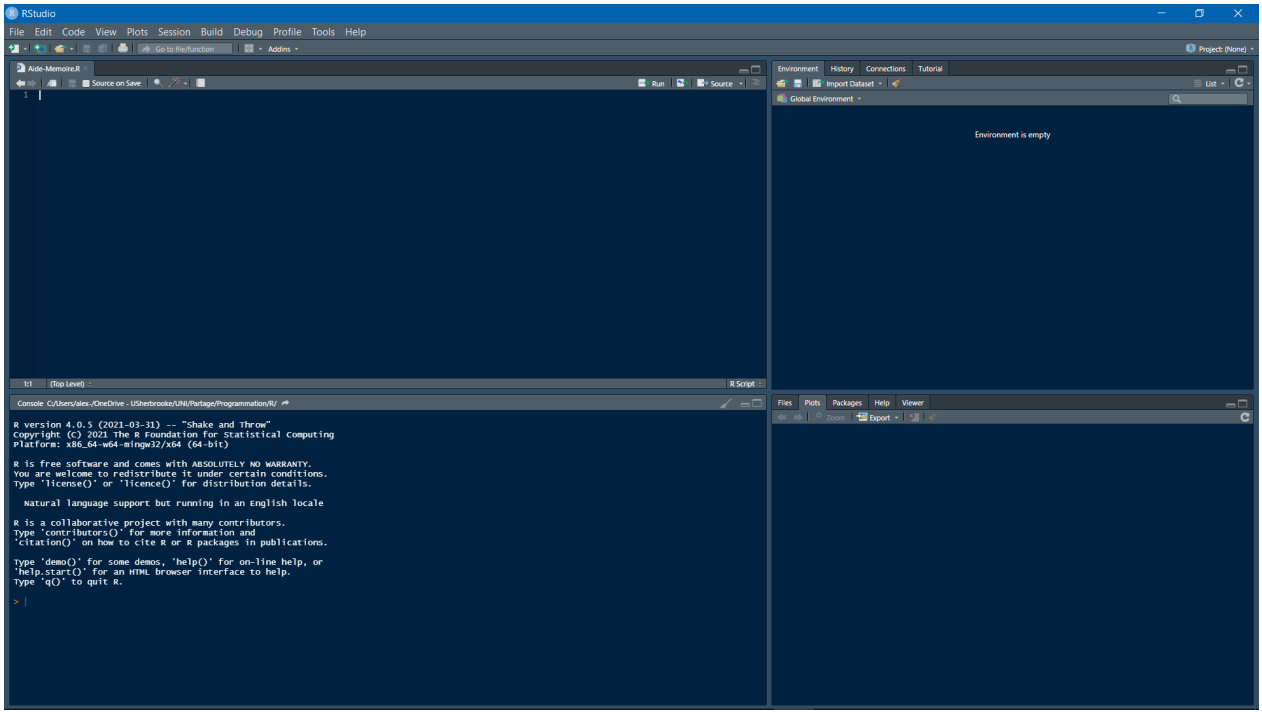


FIGURE 1 : Première ouverture de RStudio

[1] Il est possible de mettre le fond en noir comme sur la capture d'écran en procédant ainsi : *Tools* > *Global Options* > *Appearance* > *Editor Theme*

1.1 *L'interface et la programmation orientée objet*

Comme vous pouvez l'observer en ouvrant RStudio, l'interface est divisée en quatre parties distinctes. Premièrement, le **script** se trouve dans la fenêtre en **haut à gauche**. C'est dans cette fenêtre que vous devez écrire TOUTES vos lignes de codes. Pourquoi ? Puisque c'est ce script qui offre une pérennité à ce que vous écrivez, au même titre qu'un document Word que vous rédigez et sauvegardez. Ces lignes de code sont écrites sur le script dans le langage R.

Cela nous amène alors, deuxièmement, à la fenêtre en **bas à gauche** qui se nomme la **console**. C'est cette console qui utilise le langage R pour effectuer les calculs, les commandes, les fonctions, etc. qui sont passées via le script que vous rédigez. Il est techniquement possible d'écrire directement les commandes dans la console, mais celle-ci ne conserve aucune trace de ces dernières, ne permettant donc pas de les exécuter à nouveau si nécessaire (et ce sera nécessaire). Il est donc totalement DÉCONSEILLÉ d'écrire directement dans la console. Pour exécuter une commande sur RStudio, il faut : l'écrire ou la copier-coller dans le script s'assurer que son curseur soit bien dans la ligne que l'on souhaite exécuter, et cliquer sur le bouton *Run* qui se situe en haut à droite de la fenêtre de script. Vous verrez quelque chose apparaître dans la console. Il ne s'agit que d'une confirmation que la commande a bien été exécutée. Essayez, par exemple, d'écrire *x=5* dans le script et appuyez sur le bouton *Run*. Vous devriez voir apparaître, dans la console en bas à gauche, exactement le même texte précédé de ce symbole : « > ». Si rien d'autre n'apparait, c'est que cela a fonctionné. Un code d'erreur apparaîtra si une commande ne fonctionne pas comme elle le devrait.

Cette manipulation nous amène, troisièmement, à la fenêtre qui se situe en **haut à droite**, soit votre **environnement de travail**. Le langage de programmation R fonctionne avec une logique dite « orientée objet ». Il s'agit d'un mode de fonctionnement qui permet de créer des *objets* abstraits dont on en spécifie la nature et le nom, les liants ainsi à une information. En exécutant la formule *x=5*, par exemple, nous avons créé un objet nommé *x* auquel nous avons attaché la valeur numérique *5*. À noter qu'il est possible de nommer votre objet comme vous le désirez (Ex : chien="pomme"). Il sera très important de conserver cette logique en tête pour la suite des choses. C'est donc dans votre environnement de travail qu'apparaîtront tous les objets que vous allez créer. Avec des objets plus complexes comme des bases de données, cette fenêtre de RStudio vous permettra aussi d'accéder à certaines informations importantes. Vous avez aussi d'autres onglets associés à cette fenêtre que

celui *Environment*, comme *History*, un historique de vos manipulations, ou *Tutorial*, un tutoriel pour vous assister dans l'utilisation de RStudio.

Quatrièmement, la fenêtre en **bas à droite** dispose d'abord de l'onglet *Plots*. Cet onglet permet de visualiser tous les graphiques qui seront créés à partir de RStudio dans cette fenêtre qui peut, comme toutes les autres fenêtres, être élargie ou rétrécie. Ensuite, l'onglet *Files* permet d'afficher les dossiers et fichiers reliés au répertoire de travail *working directory*, c'est-à-dire le dossier source dans lequel le script sur lequel vous écrivez se trouve. Vous n'aurez probablement pas à utiliser fréquemment cet onglet. Vous pouvez aussi, en cas de besoin, accéder à l'onglet *Packages* qui vous permet de visualiser, télécharger et activer manuellement les packages offerts par RStudio. Il est toutefois déconseillé de naviguer dans les packages via cet onglet si vous n'êtes pas suffisamment à l'aise pour bien le comprendre. Cet aspect sera couvert dans la section 3.2 du présent guide.

2 Les opérateurs arithmétiques et logiques

Dans sa syntaxe, R utilise les opérateurs arithmétiques et logiques afin d'effectuer des calculs ou vérifier des conditions. Les opérateurs arithmétiques sont les opérateurs de calcul comme l'addition, la soustraction ou la division, alors que les opérateurs logiques reposent sur l'algèbre booléenne (des conditions qui sont vraies ou fausses). Les opérateurs arithmétiques serviront à faire des calculs sur des variables, des bases de données, ou tout autre objet simple ou complexe, alors que les opérateurs logiques serviront à vérifier des conditions. Les conditions permettront, entre autres, de filtrer une base de donnée en fonction d'une caractéristique précise (Ex : nous pourrions filtrer une base afin de conserver seulement les personnes âgées de moins de 25 ans). Il est normal de ne pas saisir encore toutes les particularités et utilités des opérateurs logiques à cette étape, le tout sera couvert plus loin dans ce guide. Il est toutefois pertinent de retenir que les opérateurs arithmétiques et logiques sont codés de la manière suivante dans R :

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

(a) Opérateurs arithmétiques

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

(b) Opérateurs logiques

FIGURE 2 : Codage des opérateurs arithmétiques et logiques dans R

3 Les fonctions

En plus des opérateurs mathématiques, le langage de programmation R utilise des *fonctions* afin d'effectuer des manipulations simples ou complexes sur les objets créés. Les fonctions ont toutes une structure similaire : elles débutent par leur **nom**, suivi de **parenthèses**, dans lesquelles s'insère une série d'**arguments séparés par des virgules**. Il sera très important de retenir cette structure. Commençons d'abord par créer un objet. Nous utiliserons ensuite une fonction très simple qui permet d'arrondir un chiffre à virgule à la longueur désirée. Vous pouvez copier-coller[2] les lignes suivantes afin d'en faire l'exercice :

```
x = 10.5678
x_arrondi = round(x, 2)
```
(1)

Par les lignes précédentes, nous avons d'abord créé un objet nommé « x » qui vaut « 10.5678 ». La seconde ligne de code permet ensuite, via la fonction ***round()***, d'arrondir **x** à **2 chiffres après la virgule**. Dans cet exemple simple, la fonction permet de passer le chiffre à arrondir comme premier argument, ici « x », ainsi que le nombre de chiffres à conserver après la virgule comme second argument, ici « 2 ». La structure de la fonction *round()* sera toujours la même. D'ailleurs, comme mentionné précédemment, TOUTES les fonctions utiliseront exactement cette syntaxe : le nom de la fonction, les parenthèses et les arguments séparés par des virgules à l'intérieur de celles-ci. À noter que l'exemple précédent permet de **créer un second objet dénommé *x_arrondi*** afin de sauvegarder la nouvelle valeur calculée.

Une autre fonction de base de R qui sera importante à conserver en tête est la fonction *c()*. Cette fonction extrêmement simple permet de combiner plusieurs objets afin d'en faire une liste, une énumération. Elle peut ainsi remplacer un seul objet unique par une énumération d'un plus grand nombre. Elle fonctionne d'ailleurs dans plusieurs autres fonctions utilisées par R, de manière combinée. Reprenons l'exemple précédent, cette fois en remplaçant l'argument « x » dans la

[2] À noter que vous devez OBLIGATOIREMENT disposer chaque ligne de code les unes par-dessus les autres comme elles le sont dans le document PDF. Cela vaut aussi pour toutes les autres lignes de codes que vous aurez à copier-coller dans ce document.

fonction *round()* par une énumération de trois nombres distincts en utilisant la fonction *c()* pour les combiner en tant que premier argument, comme suit :

```
x = 10.5678
y = 9.8543
z = 25.9468
x_y_z_arrondis = round(c(x,y,z), 2)
```
(2)

3.1 *Comment définir une fonction*

Bien que l'ensemble des fonctions que vous allez utiliser dans R se retrouvent dans l'installation de base ou dans les packages, il peut être pertinent de savoir qu'il est possible pour vous de définir manuellement des fonctions directement dans votre script. Cette manipulation requiert toutefois une connaissance un peu plus approfondie des différents éléments qui constituent une fonction. Comme nous l'avons vu précédemment, une fonction peut être identifiable par l'utilisation des parenthèses. Le nom de la fonction se situe directement devant l'ouverture de la parenthèse (la première), alors que les différents arguments de la fonction se retrouvent entre les parenthèses, séparés individuellement par des virgules. Essentiellement, une fonction exécute un calcul ou une manipulation qui transforme le ou les éléments spécifiés comme arguments, et retourne un résultat qui peut être associé à un nouvel objet.

Définissons ici une fonction qui vous permettra de calculer automatiquement la marge d'erreur d'un sondage ou de tout autre échantillon. Cette fonction nécessitera trois arguments : le « Z » associé au seuil de signification désiré (généralement 1.96 pour un seuil de 95%), la proportion observée « pr » (il est de mise d'utiliser systématiquement 0.5 afin de calculer la plus grande marge d'erreur sur un échantillon), et puis finalement la taille de l'échantillon « n ». Cette fonction retourne la valeur de la marge d'erreur (sur une base 1) lors de son utilisation, qui peut d'ailleurs être stockée dans un nouvel objet que vous pouvez nommer comme vous le désirez. Il est normal que vous ne connaissiez pas encore la signification de toutes ces mesures, nous allons y revenir ultérieurement dans ce guide, mais sachez que cette fonction vous sera très utile pour faciliter certaines manipulations. Voici les lignes de code qui permettent de définir la fonction en question.

À noter que vous DEVEZ copier-coller les lignes de code exactement de la manière dont elles sont disposées sur le document, et NON SUR UNE SEULE LIGNE. Cette consigne vaudra aussi pour toutes les lignes de code présentes dans ce document.

```
MargeErreur = function(Z,pr,n)
{
  me = round((sqrt(pr*(1-pr)/n))*Z, 2)
  print(c("Taille de l'échantillon:",n), quote=F)
  print(c("Niveau de confiance:",Z), quote=F)
  print(c("Marge d'erreur:",me), quote=F)
  print(c("Intervalle de confiance (borne inférieure):",pr-me), quote=F)
  print(c("Intervalle de confiance (borne supérieure):",pr+me), quote=F)
  return(me)
}
```
(3)

Une fois les lignes de code exécutées et la fonction définie, il est possible de l'utiliser à n'importe quel moment, de la même manière que toute autre fonction déjà présente dans R. Il suffit donc d'écrire le nom de la fonction, suivi des parenthèses, à l'intérieur desquelles se trouvent les trois arguments séparés par des virgules. Par exemple, pour calculer la plus grande marge d'erreur reliée à un sondage de 1000 personnes à un niveau de confiance habituel de 95% (1.96) et sauvegarder le résultat dans un nouvel objet, on exécuterait la ligne de code suivante :

```
ME = MargeErreur(1.96,0.5,1000)
```
(4)

3.2 *L'installation et l'activation des packages*

Lors du téléchargement initial, R contient déjà plusieurs fonctions par défaut comme *table()* pour faire un tableau ou *plot()* pour faire un graphique. Il est cependant possible d'utiliser des centaines et des centaines de fonctions additionnelles qui sont définies dans des installations supplémentaires

nommées *Packages*. Les packages doivent d'abord être **installés** sur votre ordinateur en utilisant la fonction *install.packages("Nom du package",depend=TRUE)* en remplaçant ici la partie entre guillemets par le nom du package désiré. Cette manipulation sert seulement à télécharger les fichiers associés au package sur votre ordinateur, il est donc nécessaire de l'exécuter qu'à une seule reprise par ordinateur et par package. Vous pouvez accéder à un script préenregistré qui vous permettra d'installer quelques packages supplémentaires via ce lien. Il ne vous suffit qu'à copier-coller les lignes de code présentes dans le fichier au début de votre propre script, ou bien de télécharger le fichier .R entier, l'ouvrir et l'exécuter.

Une fois les packages installés sur votre ordinateur, vous devez ensuite les **activer** à CHAQUE fois que vous ouvrez une nouvelle fenêtre de R après que la vôtre ait été fermée. Les packages peuvent être activés avec la fonction *library("Nom du package")*. Cette manipulation est très importante et doit être exécutée systématiquement au tout début de votre script pour chaque package, en plus d'avoir à installer les packages avant de les activer. Il est donc nécessaire d'installer un package avant de l'activer, et ce, pour chaque package à activer. Vous pouvez accéder à un script préenregistré qui vous permettra d'activer les packages précédemment installés via ce lien. Il ne vous suffit qu'à copier-coller les lignes de code présentes dans le fichier au début de votre propre script, ou bien de télécharger le fichier .R entier, l'ouvrir et l'exécuter. Je vous conseille de conserver ce fichier dans vos dossiers personnels. Il vous sera fortement utile afin d'activer ces packages à CHAQUE fois que vous ouvrez une nouvelle fois RStudio après l'avoir fermé.

Petite mise en garde : certains packages différents définissent deux fonctions différentes avec le même nom. Cela fait donc en sorte de créer un **conflit** et R peut avoir de la difficulté à comprendre quelle fonction vous appelez exactement. Il retournera alors un message d'erreur, même si vous croyez avoir bien utilisé la fonction. Il faut donc rester vigilent lorsque l'on active une grande quantité de packages à la fois. Pour remédier à ce problème, il est possible d'ajouter le nom du package auquel la fonction désirée se rapporte, suivi d'un double deux-points « : : » avant le nom d'une fonction afin de bien spécifier de quel package exactement provient ladite fonction. Cela permet donc d'éviter tout conflit. Un conflit fréquent survient, par exemple, avec la fonction *rename()* puisque les packages *dplyr* et *plyr* définissent tous deux une fonction portant le nom

rename(), mais avec des arguments différents. On peut alors préciser à R d'utiliser la fonction *rename()* définie par le package *plyr* de cette manière : « plyr : :rename(base,replace=c(...)) ».

4 L'importation d'une base de données

Pour importer une base de données dans R, il existe quelques fonctions légèrement différentes, dépendamment de la forme qu'a la base à importer. Pour un fichier téléchargé sur l'ordinateur, l'extension du fichier de la base (.csv, .xlsx, .sav, etc.) déterminera la fonction à utiliser. Lors de l'importation de la base, nous nommons l'objet créé *base* par convention, mais libre à vous de nommer votre base comme vous le souhaitez. Il ne suffit qu'à remplacer le tout premier mot *base* qui vient avant le « = » par ce que vous désirez.

4.1 *Le package WDI de la Banque Mondiale*

La Banque Mondiale offre un package qui permet d'importer une base à partir de la fonction *WDI()*. Cette fonction contient quatre arguments centraux : *country*, *indicator*, *start* et *end*. L'argument country permet d'abord de cibler un ou plusieurs pays dont les données seront importées, l'argument indicator permet ensuite de spécifier quel(s) indicateur(s) de la Banque Mondiale seront importés, alors que les arguments start et end permettent de délimiter la période souhaitée en indiquant les années de début et de fin. Les pays et les indicateurs de la Banque Mondiale fonctionnent par codes. Évidemment, vous ne devez pas connaître par cœur ce genre de codes. Vous pouvez donc avoir accès à une liste des codes des pays et des indicateurs de la Banque Mondiale sur le site Dimension de l'Université de Sherbrooke. Il ne vous suffit qu'à rechercher par mot clé dans cette page avec le raccourci *Ctrl* + *F* sur votre clavier afin de trouver les pays et les indicateurs que vous désirez. Vous pouvez copier-coller la ligne de code suivante et observer la base qui a été importée :

```
base=WDI(country=c("US","CA","CN"),
         indicator=c("SP.POP.TOTL","EN.ATM.CO2E.PC","EG.USE.PCAP.KG.OE"),
         start=1960,end=2021)
```

(5)

4.2 *Un fichier CSV*

Pour importer une base qui se trouve dans un fichier avec l'extension « .csv » (comma separated values), on doit utiliser l'une des deux fonctions suivantes. La première pour un document en français et la deuxième pour un document en anglais. Essayez les deux et voyez laquelle permet de créer une base conventionnelle (regardez le nombre de variables pour le savoir. Si la base n'en a qu'une seule, essayez l'autre fonction) :

```
base=read.csv2(file.choose(),header=TRUE,encoding="UTF-8")
base=read.csv(file.choose(),header=TRUE,encoding="UTF-8")
```
(6)

4.3 *Un fichier Excel*

Pour importer une base qui se trouve dans un fichier avec l'extension « .xlsx » (utilisé par Excel), on doit utiliser la fonction suivante :

```
base=read.xlsx(file.choose(),sheet=1,colNames=TRUE)
```
(7)

4.4 *Un fichier SPSS*

Pour importer une base qui se trouve dans un fichier avec l'extension « .sav » (utilisé par SPSS), on doit utiliser la fonction suivante :

```
base=spss.get(file.choose(),use.value.labels=TRUE,encoding="UTF-8")
```
(8)

4.5 *Une base de données hébergée sur un site web*

Si jamais la base est hébergée sur un site web, on peut utiliser la fonction correspondante à l'extension du fichier (.csv, .xlsx, .sav ou autre) et remplacer l'argument *file.choose()* par un URL entre guillemets. Ce sera le cas, par exemple, des bases de données hébergées sur le site Dimension de l'Université de Sherbrooke :

```
base=read.csv2("URL de la base",header=TRUE,encoding="UTF-8")
base=read.csv("URL de la base",header=TRUE,encoding="UTF-8")
base=read.xlsx("URL de la base",sheetIndex=1,header=TRUE,encoding="UTF-8")
base=spss.get("URL de la base",use.value.labels=TRUE,encoding="UTF-8")
```
(9)

4.6 *Base de données à importer pour les exemples du guide*

Pour effectuer les exemples des différentes manipulations que nous allons explorer dans ce guide, vous devez importer la base de données de l'enquête électorale canadienne de 2015. Celle-ci est disponible en ligne sur le site Dimension à la page suivante. Pour l'importer dans RStudio, il vous suffit de copier-coller la ligne de code suivante et de l'exécuter :

```
base=read.csv2("https://dimension.usherbrooke.ca/donnees/base2015CES.csv",
               header=TRUE, encoding="UTF-8")
```
(10)

5 Les manipulations à effectuer sur une base de données

5.1 *Comment visualiser et explorer une base de données*

Après avoir importé votre base dans R avec l'une des méthodes présentées à la section 4, vous devriez pouvoir observer un nouvel objet qui s'est créé dans la fenêtre en **haut à droite**. L'objet sera nommé « base » par défaut si vous avez repris les lignes présentées en exemple, sinon il porte le nom que vous avez décidé de lui donner. Si vous cliquez sur le mot « base » dans votre environnement de travail, votre base s'ouvrira dans un nouvel onglet de la fenêtre en **haut à gauche**. Cela vous permettra donc de visualiser votre base dans sa totalité afin de bien identifier ce qui la constitue. Vous pouvez aussi cliquer sur la petite flèche bleue située juste à la gauche de votre base. Cela fera en sorte de faire défiler une liste d'informations sur la base. En fait, chaque ligne d'information est une variable présente dans la base. Pour chaque variable, on peut observer son nom, son type et les premières valeurs de la variable en question.

Dès l'ouverture de votre base de données, il convient d'identifier une série d'informations importantes concernant celle-ci. Il faut d'abord **identifier et ouvrir la base** nouvellement créée de la manière présentée au paragraphe précédent. Ensuite, vous devez vous demander **quelle est votre unité d'analyse**. Analysez-vous des pays ? des humains ? des partis politiques ? etc. Une fois l'unité d'analyse bien identifiée, vous devez prendre conscience des **variables** qui sont contenues dans la base de données, et comment elles sont construites et codées. Vous devez aussi, par le fait même, observer quelles sont les **valeurs possibles** que peut prendre chaque variable de la base. Pour finir, vous devez aussi porter attention au **nombre de rangées (observations)** et au **nombre de colonnes (variables)** afin de savoir combien de variables la base contient et quelle est la taille de l'échantillon avec lequel vous travaillez (combien d'unités d'analyse, essentiellement).

5.2 *Les informations générales sur la base*

Premièrement, la fonction *str()* offre des informations plus générales sur la base, telles que le nombre d'observations (de rangées), notées obs., et le nombre de variables (colonnes). Elle donne essentiellement des informations très similaires à celles présentes dans votre environnement de

travail (dans la fenêtre en **haut à droite**). Les différentes variables contenues dans la base importée sont ensuite énumérées de haut en bas. Chaque variable est précédée d'un signe de dollar ($), suivi de son nom exact dans la base. Vient ensuite le type de la variable (« chr » pour les variables en mots/lettres, « int » pour les variables en nombres entiers et « num » pour les variables en nombres à virgule). On peut finalement voir un aperçu des quelques premières valeurs pour chaque variable après les « : » qui suivent le type de la variable.

Le symbole **« $ » est très important** et il est à retenir. Vous l'utiliserez systématiquement pour faire appel à une variable contenue dans une base. **La structure d'appel à une variable sera toujours la suivante : base$variable**. Vous devez donc spécifier à R le nom de la base dans laquelle vous allez chercher ladite variable, apposer le symbole « $ », puis indiquer le nom de la variable que vous allez chercher dans la base indiquée. Vous pouvez maintenant exécuter la fonction suivante et observer le résultat :

```
str(base)
```
(11)

5.3 *Les informations générales sur les variables*

Deuxièmement, la fonction *summary()* donne accès à une multitude d'informations plus spécifiques sur chaque variable de la base. Pour les variables qualitatives, les informations sont limitées, montrant seulement le nombre d'observations (length) et le type de la variable (voir section 5.2). Pour les variables quantitatives, la fonction donne des informations cruciales pour une présentation univariée comme la moyenne, la médiane, le minimum, le maximum et le nombre de données manquantes « NA ».

```
summary(base)
```
(12)

5.4 *Comment renommer des variables (colonnes)*

Troisièmement, la fonction *rename()* permet de renommer certaines variables (colonnes) qui ne sont pas codées de manière très intuitive. C'est le cas, par exemple, des codes des indicateurs de la Banque Mondiale que nous avons explorés à la section 4.1. La fonction *rename()* contient toujours

deux arguments, mais la longueur du deuxième peut varier en fonction du nombre de variables à renommer. Le premier argument de la fonction est toujours le nom de la base comme elle est codée dans votre environnement de travail (fenêtre en **haut à droite**). Le second argument, nommé *replace*, a toujours la même structure : replace = c("Code actuel de la variable 1" = "Nouveau nom de la variable 1", "Code actuel de la variable 2" = "Nouveau nom de la variable 2"). Comme vous pouvez l'observer, chaque changement de nom est séparé par une virgule à l'intérieur de la fonction *c()* qui englobe le tout (voir section 3). En reprenant l'exemple de la base de l'enquête électorale canadienne de 2015 importée à la section 4.6, nous allons renommer les variables « age » par « an_naissance » et « p.aborig » par « autochtone » pour être plus représentatif[3] :

```
base=rename(base,replace=c("age"="an_naissance","p.aborig"="autochtone"))
```
(13)

5.5 *Comment recoder les valeurs d'une variable*

Quatrièmement, la fonction *recode()* permet de recoder les valeurs possibles d'une variable. On peut notamment utiliser cette fonction pour remplacer des variables numériques en lettres, ou l'inverse. Par exemple, on pourrait vouloir modifier les valeurs d'une variable dichotomique qui est codée « OUI » ou « NON » pour les valeurs numériques « 1 » et « 0 », où « 1 » correspond à « OUI » et « 0 » à « NON ». Nous effectuerons fréquemment cette manipulation puisqu'il est généralement préférable de travailler avec des valeurs numériques plutôt qu'avec des valeurs en lettres pour les variables dichotomiques. La fonction *recode()* peut toutefois servir à modifier les valeurs possibles d'une variable pour les raisons que vous désirez. La syntaxe de cette fonction différera légèrement de ce que nous avons vu précédemment. D'abord, la fonction possède deux arguments : le premier étant l'appel de la variable à modifier et le second, la modification des codes des valeurs entre guillemets et séparés par des virgules. Ensuite, comme il a été présenté dans le deuxième paragraphe de la section 5.2, la structure syntaxique d'une variable diverge légèrement de celle des autres objets. Comme le nouvel objet créé par cette fonction est une variable et non une base, celui-ci doit nécessairement suivre la structure d'appel d'une variable,

[3] **MISE EN GARDE :** Il est possible que R vous renvoie un code d'erreur relié à un **conflit** en utilisant la fonction *rename()*.

soit « base$variable ». Reprenons la base importée à la section 4.6, et modifions la variable nouvellement renommée « autochtone » en une variable dichotomique. Pour ce faire, nous allons changer la valeur du « Non (1) » pour « 0 » et regrouper les valeurs « Première nation (2) », « Métis (3) », « Inuit (4) » et « Autre (6) » en leur donnant la valeur « 1 ». Les autres valeurs seront considérées comme des données aberrantes, elles auront donc une valeur inexistante nommée « NA » :

```
base$autochtone=recode(base$autochtone,"c(1)=0 ; c(2,3,4,6)=1 ; else=NA")
```

(14)

Prenez le temps de bien observer la manière dont cette fonction est construite et permet de transformer les valeurs initiales d'une variable en de nouvelles valeurs que vous choisissez vous-même. Vous pourrez copier-coller cette fonction dans le futur et changer quelques éléments seulement, tels le nom de la variable ou les valeurs que vous désirez modifier. Il est TRÈS IMPORTANT de toujours enchâsser la valeur initiale et la nouvelle valeur entre des apostrophes lorsque celle-ci est en caractères (lettres). Pour des valeurs chiffrées, cette manipulation n'est pas nécessaire. Elle sert essentiellement à indiquer à R que tout ce qui se trouve entre les deux apostrophes est à prendre comme une seule valeur.

Il est aussi possible de recoder les valeurs d'une variable de manière individuelle en utilisant une condition, à la manière de la création d'une nouvelle variable qui sera présentée à la section suivante. Pour un rappel concernant les conditions et les opérateurs logiques, vous pouvez consulter la section 2. On pourrait, toujours en reprenant la base de la section 4.6, retirer les observations de la variable « an_naissance » lorsque celles-ci correspondent à « 1000 ». Comme nous pouvons être certains qu'aucun individu est né en l'an 1000, on peut recoder seulement cette valeur et lui donner la valeur « NA ». On pourrait, si on le désire, effectuer cette manipulation de manière individuelle pour chaque valeur de la variable, mais il est recommandé d'utiliser la

fonction *recode()* présentée précédemment lorsqu'il y a plusieurs valeurs à modifier en même temps. Faites un copier-coller de la fonction suivante[4] et observez le résultat :

```
base$an_naissance[base$an_naissance == 1000] = NA
```
(15)

5.6 *Comment créer une nouvelle variable*

Cinquièmement, bien que la création d'une nouvelle variable ne passe pas par une fonction directement, elle est primordiale lorsque vient le temps d'analyser un phénomène social. La création de variables peut servir, par exemple, à séparer une variable quantitative en catégories et sauvegarder le résultat dans une nouvelle variable. C'est une manipulation que nous ferons fréquemment, notamment avec des variables comme l'âge ou le revenu. Comme mentionné précédemment, la création d'une variable ne nécessite pas l'utilisation d'une fonction, bien qu'il soit possible de passer par la fonction *recode()*. Ce qui est crucial, c'est que nous devons **créer un nouvel objet qui a la forme d'une variable** (voir le deuxième paragraphe de la section 5.2). Les manipulations possibles sont exactement les mêmes qu'à la section précédente, mais le résultat sera sauvegardé dans un nouvel objet (une nouvelle variable) plutôt que d'écraser la même variable. Reprenons encore la base de la section 4.6 et commençons par créer une variable pour l'âge en soustrayant l'année de naissance à l'année de réponse du questionnaire (2015), et créons ensuite une autre variable afin de séparer les âges en trois catégories (18-39, 40-59 et 60+). Faites un copier-coller des lignes de code suivantes (une par une) et observez le résultat sur le nombre de variables de votre base.

Création de la variable âge :

```
base$age = 2015-base$an_naissance
```
(16)

Création de la variable pour les trois catégories d'âge :

[4] Cette ligne de code se lit comme suit en français : La variable « an_naissance » sera égale à « NA » lorsque la valeur de celle-ci correspond à 1000.

```
base$agecat[base$age >= 18 & base$age <= 39] = 1
base$agecat[base$age >= 40 & base$age <= 59] = 2
base$agecat[base$age >= 60] = 3
```
(17)

5.7 *Comment filtrer une base de données*

Sixièmement, la fonction *subset()* permet de filtrer une base de deux manières différentes. Il est possible de ne conserver que certaines observations (rangées) selon une condition, ou de ne conserver que certaines variables (colonnes) spécifiques.

5.7.1 Conserver seulement certaines observations (rangées)

En premier lieu, la condition qui permet de retirer certaines observations utilise les opérateurs logiques présentés précédemment (voir section 2) afin de vérifier si l'énoncé est VRAI ou FAUX, en ne retenant que les observations pour lesquelles la condition est VRAIE. Par exemple, dans une base constituée de personnes répondantes à un sondage, on pourrait décider de ne conserver que les personnes s'identifiant comme des femmes. Toujours en reprenant la base de la section 4.6, nous pourrions décider de créer une nouvelle base en ne conservant que les observations (rangées) pour lesquelles la variable « sex.r » correspond à 2 (femmes). On créerait ainsi une nouvelle base de données constituée uniquement de sujets féminins. Petite mise en garde, si vous décidez de créer un nouvel objet avec la base filtrée, il est important de bien choisir le nom du nouvel objet créé. Si vous le renommez aussi « base », vous allez écraser la base initiale en la remplaçant par la nouvelle (c'est en fait ce que nous avons effectué au paragraphe précédent pour renommer les variables de la base). Vous pouvez toutefois éviter cette situation en renommant le nouvel objet « nouvellebase », « base2 » ou un autre nom qui décrit mieux la base. Vous pouvez copier-coller la ligne de code suivante et observer le résultat sur la nouvelle base créée :

```
base_femmes = subset(base, base$sex.r == 2)
```
(18)

La fonction se lirait donc comme suit en français : Ne conserver que les observations (rangées) de la base « base » pour lesquelles la variable « sex.r » de la base « base » correspond à 2.

5.7.2 Conserver seulement certaines variables (colonnes)

En second lieu, le deuxième argument de la fonction *subset()* peut prendre la forme d'un *select = c("variable1", "variable2", "variable3")* afin de ne conserver que les variables désirées. Cette manipulation est extrêmement utile lorsque l'on travaille avec des bases de données qui peuvent avoir quelques centaines de variables, voire des milliers. Par soucis de précaution, nous allons généralement sauvegarder le résultat dans un nouvel objet afin de ne pas écraser la base de données initiale. En reprenant toujours le même exemple[5] avec les modifications qui ont été apportées précédemment, nous pourrions décider de ne conserver que certaines variables. Nous conserverons ici les variables « sex.r », « age », « p.intpol », « p.voted » et « income ». Vous pouvez copier-coller la ligne de code suivante et l'exécuter pour effectuer cette manipulation :

```
base2=subset(base,select=c("sex.r","age","p.intpol","p.voted","income"))
```

(19)

[5] Pour reproduire la manipulation sur une autre base de données, il ne suffit qu'à remplacer le nom de la base et le nom des variables dans le *select*.

6 L'inférence statistique

L'inférence statistique est un concept très important lorsque nous travaillons avec des échantillons (c'est-à-dire, à peu près tout le temps). Chaque fois que l'on traite un échantillon, il faut avoir la conviction qu'on peut en tirer profit, mais il faut être prudent puisque celui-ci contient nécessairement une marge d'erreur. De manière plus précise, l'inférence statistique consiste en l'ensemble des méthodes qui permettent d'obtenir de l'information sur une population à partir de données provenant d'un échantillon tiré aléatoirement de cette population.

Une population est constituée d'un nombre immense d'individus. Interroger une population est très coûteux et presque impossible. Le Canada, par exemple, produit des recensements à tous les cinq ans et une bonne part des données qui sont récoltées proviennent elles-mêmes d'échantillons. Cela démontre bien à quel point les enquêtes sur des populations sont limitées.

Dans cette optique, nous allons donc travailler avec des échantillons dans presque la totalité des recherches en sciences sociales puisque ceux-ci sont moins coûteux en ressources.

6.1 *Historique et fondements de l'inférence*

Plusieurs personnages historiques importants ont permis de définir les fondements sur lesquels se basent nos méthodes actuelles d'inférence statistique, notamment Ronald Aylmer Fisher, Carl Gauss, Pierre-Simon de Laplace et Blaise Pascal.

Blaise Pascal, au 17e siècle, a posé les fondements à l'intuition du hasard et des probabilités. Il s'est mis à réfléchir aux jeux de hasard, particulièrement les jeux impliquant des pièces de monnaie. Commençons d'abord par établir que « Pile » sera représenté par la lettre « P » et « Face » par la lettre « F ». Si l'on tire une seule pièce de monnaie au pile-ou-face, nous serons tous d'accord pour dire que les probabilités de tomber sur pile ou face sont de 50%-50%. Les deux seules **permutations** possibles sont donc « P,F ». Si l'on tire deux pièces de monnaie maintenant, nous arriverons au résultat qu'il existe quatre permutations possibles : « PP, PF, FP, FF ». En observant plus en détail les deux permutations du milieu, on peut voir qu'elles contiennent en fait toutes deux pile et face. Il s'agit donc de la même **combinaison**, mais dans un ordre différent.

En poursuivant cette logique, si l'on tire maintenant trois pièces de monnaie, les permutations possibles seront les suivantes : « PPP, PPF, PFP, FPP, FFP, FPF. FFP, FFF ». Malgré la pluralité de permutation, on peut faire ressortir quatre combinaisons distinctes, puisque les trois permutations en rouge sont les mêmes combinaisons, mais dans un ordre différent, et cela vaut aussi pour les trois permutations en jaune. Si l'on poursuit toujours cet exercice en ajoutant une pièce de monnaie tirée, on recrée le triangle de Pascal :

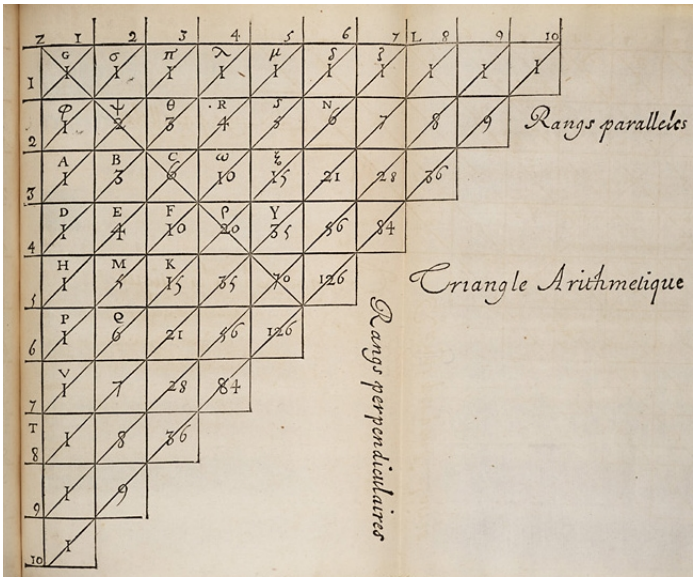


FIGURE 3 : Version originale du triangle de Pascal dans le traité de Blaise Pascal

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

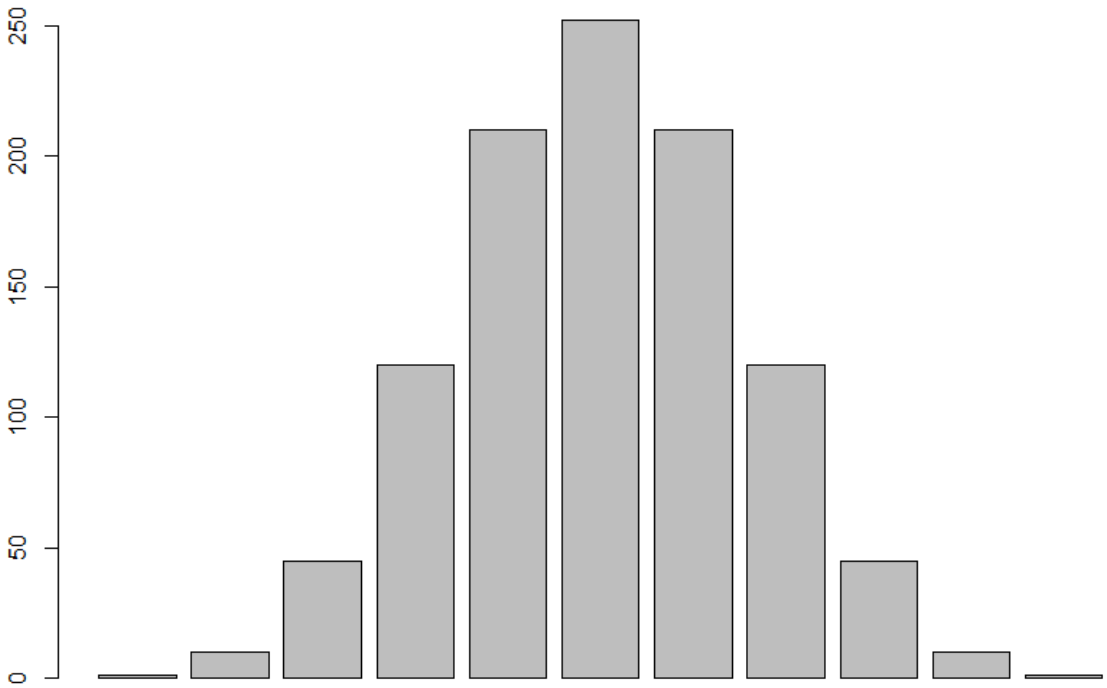
FIGURE 4 : Version épurée des six premières lignes du triangle de Pascal

10^e rangée du triangle de Pascal											
10	1	10	45	120	210	252	210	120	45	10	1

TABLEAU 1 : Permutations et combinaisons possibles si 10 pièces sont tirées à pile ou face

Lorsque l'on tire 10 pièces de monnaies maintenant, on arrive à un résultat particulièrement intéressant. La distribution que l'on aperçoit ressemble à un concept que vous avez fort probablement déjà rencontré : la courbe normale. Voici le résultat lorsque nous reproduisons la distribution du tableau 1 dans RStudio :

FIGURE 5 : Reproduction de la 10^e ligne du triangle de Pascal dans RStudio




```
> pascal = c(1,10,45,120,210,252,210,120,45,10,1)
> pascal
 [1]   1  10  45 120 210 252 210 120  45  10   1
> barplot(as.vector(pascal))
```

On retrouve ici 11 combinaisons possibles. Certaines sont plus rares, et d'autres plus fréquentes. La plus fréquente compte 252 permutations possibles, alors qu'il ne s'agit en fait que d'une seule combinaison si l'on ne tient pas compte de l'ordre des piles et des faces. Les deux combinaisons les plus rares surviennent lorsque les 10 pièces tirées sont des piles ou les 10 sont des faces. La logique sera la même plus on grossit l'échantillon de pièces de monnaie tirées. C'est ce principe qui jeta les bases de la loi normale.

6.2 *Conclusions de Pascal*

Avec cet exercice, Pascal découvre qu'il y a une logique au hasard. Ce dernier, qui semblait si illogique à première vue, répond finalement à un ordre, à des lois. On peut donc utiliser la connaissance que nous avons de cet ordre afin d'établir ce qu'il y a de plus probable et de moins probable, en unissant hasard et savoir. Les fondements statistiques théorisés par Pascal étaient d'ailleurs déjà connus par des scientifiques du Moyen-Orient, du Maghreb et de l'Orient des siècles avant même sa naissance.

D'autres figures ont continué sur les bases jetées par Pascal, notamment Adolphe Quetelet qui fut un précurseur de l'étude démographique. Quetelet va être un des premiers à mettre de l'avant l'échantillonnage afin de connaitre les mesures d'une caractéristique humaine, sans passer par l'analyse d'une population entière. Il prendra, par exemple, un échantillon de personnes belges et il extrapolera ses conclusions sur l'ensemble de la population belge.

6.3 *Les niveaux de confiance, l'intervalle de confiance et la marge d'erreur*

Les niveaux de confiance, d'abord, déterminent la probabilité de contenir la valeur à estimer à l'intérieur de l'intervalle de confiance. Il s'agit donc du % de chances que l'on a de ne pas se tromper lorsque l'on extrapole les résultats d'un échantillonnage contenus dans l'intervalle de confiance à l'ensemble de la population. Par convention, nous utiliserons presque systématiquement un niveau de confiance de 95% et son « Z » associé, soit 1.96. Un niveau de confiance de 95% signifie, dans son interprétation, que le résultat issu d'un échantillon est valable 19 fois sur 20. Voici un tableau qui présente les niveaux de confiance et leur « Z » associés :

Niveaux de confiance	Z associé au niveau de confiance
50%	0.674
75%	1.150
90%	1.645
95%	1.960
97%	2.170
99%	2.576
99.9%	3.291

TABLEAU 2 : Niveaux de confiance et leur « Z » associé

La marge d'erreur,[6] ensuite, est calculée en multipliant l'écart-type d'une distribution par le « Z » associé au niveau de confiance choisi, soit 1.96 pour un niveau de confiance de 95% par convention. Il n'est pas nécessaire de comprendre les tenants et aboutissants de ce chiffre, mais il est à retenir. La marge d'erreur représente ce qui sépare la moyenne des bornes supérieure et inférieure de l'intervalle de confiance.

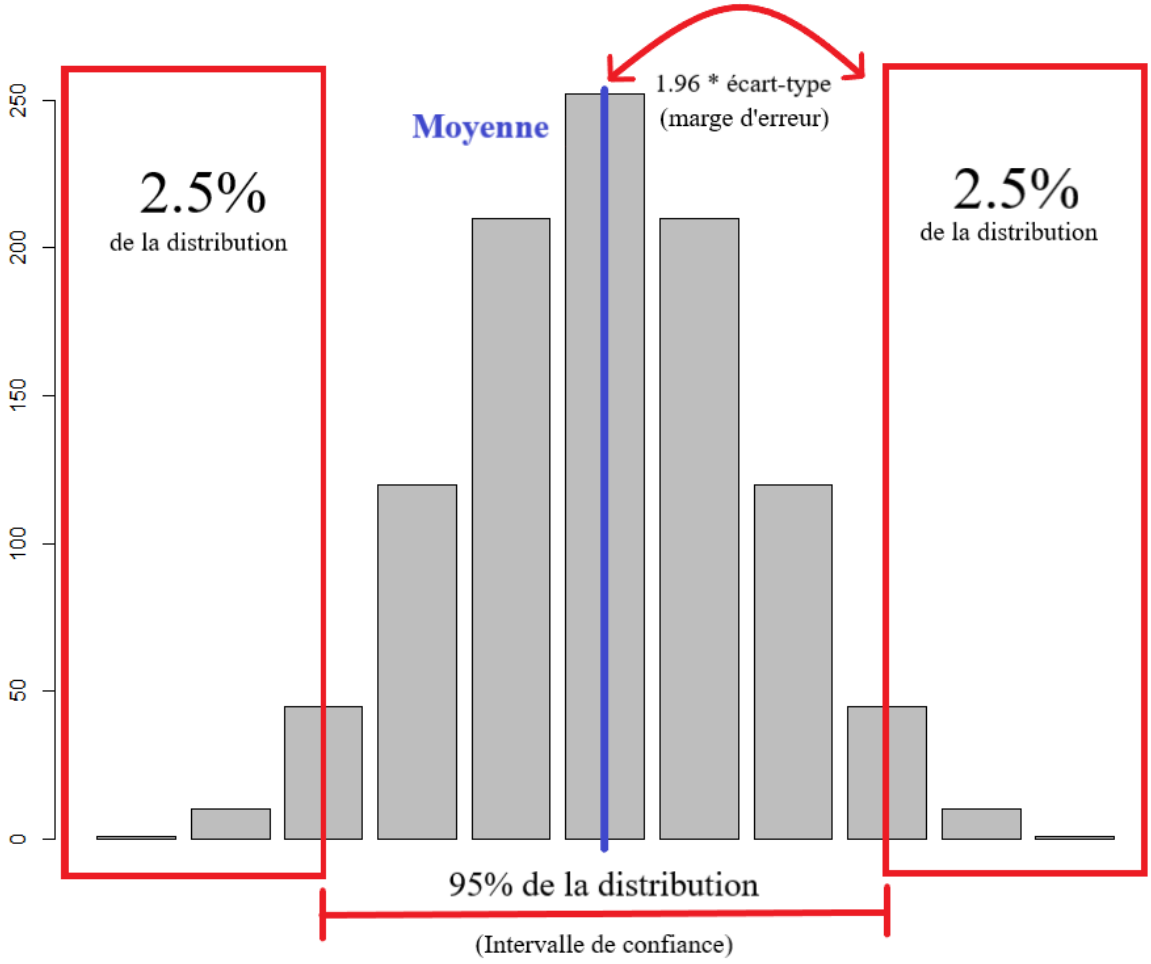
L'intervalle de confiance, pour finir, représente la zone dans laquelle se retrouve 95% de la distribution.[7] L'intervalle de confiance se calcule en soustrayant et en additionnant la marge d'erreur à la moyenne. Elle se présente comme suit : [moyenne - marge d'erreur , moyenne + marge d'erreur]. Même si nous ne pouvons pas être confiants à 100% comme nous le pourrions en analysant une population entière, on accepte qu'il y a un 5% des cas dans lesquels nous avons un échantillon trompeur, voire très trompeur.

[6] Une fonction a été définie dans la section 3.1 afin de calculer automatiquement la marge d'erreur. Une fois que la définition a été exécutée, la fonction peut être utilisée en remplaçant les arguments par les valeurs désirées.
Ex : ME = MargeErreur(1.96, 0.5, 1000)

[7] Ce % dépend du niveau de confiance choisi et de son « Z » associé qui a été utilisé dans le calcul de la marge d'erreur.

Voici un graphique qui permet de bien comprendre ce que représentent visuellement ces mesures sur une distribution normale :

FIGURE 6 : Intervalle de confiance et marge d'erreur d'une distribution normale



7 L'analyse univariée : les caractéristiques importantes d'une distribution

Avant de croiser plusieurs variables entre elles, il est de mise d'effectuer une courte analyse univariée afin de faire ressortir certaines informations pertinentes des variables à croiser. Ces informations vous permettent aussi de prendre connaissance de votre distribution et de sa répartition, et ainsi de mieux comprendre vos variables, leur nature et ce qu'elles représentent.

Avant même d'effectuer cette présentation univariée des variables comprises dans nos hypothèses, il est **NÉCESSAIRE** d'effectuer un **traitement** des variables afin de ne conserver que les valeurs désirées. En effet, par défaut, presque la totalité des variables codées dans R ont des valeurs qui signifient « refus de répondre » ou « ne sait pas » qui doivent absolument être considérées comme des « NA ». Vous pouvez consulter les sections 2, 5.5 et 5.6 du présent guide pour plus d'informations à ce sujet. Recodons alors les deux variables que nous allons utiliser en exemple dans les sous-sections suivantes (qui proviennent de la base importée à la section 4.6), soit le « revenu » et « avoir voté », afin d'en retirer les valeurs indésirables :

```
base$income[base$income < 1000 | base$income > 1000000] = NA
base$a_vote = recode(base$p.voted, "c(1)=1 ; c(2)=0 ; else=NA")
```
(20)

Pour le revenu, nous avons retiré les données en dessous de 1000$ et au-dessus de 1000000$ et pour l'acte de voter, nous avons renommé la variable pour « a_vote » en plus d'avoir recodé et conservé seulement les valeurs OUI (1) et NON (0), en accordant la valeur « NA » au reste dans les deux situations.

7.1 *Une variable qualitative*

Pour les variables qualitatives, le mode et la répartition de la distribution selon les différentes valeurs possibles peuvent être calculés de manière relativement simple. Le mode, d'abord, représente la valeur qui revient le plus souvent dans la distribution. Il ne se calcule pas avec une fonction déjà présente dans R. Nous devons donc définir manuellement la fonction qui le

calculera, de la manière présentée à la section 3.1 du présent guide. Faites un copier-coller des lignes de code suivantes **AU TOUT DÉBUT** de votre script et exécutez-les :

```
Modes = function(variable, na.rm = FALSE)
{
  if(na.rm){
    x = variable[!is.na(variable)]}
  y = as.factor(x)
  freq = summary(y)
  mode = names(freq)[freq[names(freq)] == max(freq)]
  return(as.numeric(mode))
}
```
(21)

Après avoir exécuté les lignes de code précédentes, nous pouvons ensuite utiliser la fonction *Modes()* avec notre variable comme premier argument et « na.rm = TRUE » en deuxième argument afin de ne pas considérer les « NA » dans le calcul. Vous pouvez copier-coller le ligne de code suivante :

```
mode = Modes(base$a_vote, na.rm = TRUE)
mode
```
(22)

Ensuite, après avoir déterminé la valeur de notre variable qui revient le plus souvent dans notre distribution, on peut aller vérifier comment notre distribution est répartie entre les différentes valeurs possibles que la variable peut prendre. Ce test se fait en utilisant la fonction *freq()* avec notre variable comme seul argument comme suit :

```
freq(base$a_vote)
```
(23)

Il importe d'abord de bien comprendre ce que les trois colonnes représentent. Le « n » représente le nombre d'observations en valeur absolue, le « % » représente la proportion de la distribution

qui la valeur assignée à la rangée du tableau et le « val% » a la même signification, mais en ne considérant pas les « NA » dans l'équation. La colonne « val% » représente donc le poids réel de la valeur observée dans la distribution. Cette fonction est extrêmement importante afin de bien comprendre les données avec lesquelles nous travaillons. Elle devrait être systématiquement exécutée sur chaque variable qualitative que l'on désire analyser.

Si vous observez le résultat de l'exemple, vous pourrez voir que cette distribution est particulière. Sur les 7572 personnes interrogées dans l'enquête (observations), 5355 ont la valeur « NA ». Cette situation est relativement fréquente lorsque nous traitons des questions comme « avez-vous voté aux dernières élections ? ». Pour plusieurs personnes, ce genre d'information est confidentielle et elles ne désirent pas la divulguer. On peut toutefois observer que, sans considérer les NA, 87.8% des personnes répondantes ont voté, ce qui représente 1946 personnes.

7.2 *Une variable quantitative*

Pour les variables quantitatives, les fonctions *Modes()* et *freq()* ne sont pas d'une grande utilité. Nous allons plutôt calculer certaines mesures de tendance centrale et de dispersion afin d'avoir un aperçu de la structure de notre distribution.

Premièrement, la moyenne se calcule de manière relativement simple dans R avec la fonction *mean()*, nul besoin de passer par le calcul classique d'addition de toutes les valeurs divisé par le nombre de valeurs. Cette fonction contient deux arguments principaux, soit la variable en premier argument et « na.rm = TRUE » en deuxième afin de ne pas considérer les « NA » dans le calcul. Vous pouvez copier-coller la ligne de code suivante :

```
moyenne = mean(base$income, na.rm = TRUE)
moyenne
```
(24)

Deuxièmement, la médiane représente la donnée qui sépare la distribution en deux parties qui contiennent un nombre d'observations égal. Elle s'obtient en exécutant la fonction *median()* avec les mêmes arguments que pour la moyenne, comme suit :

```
mediane = median(base$income, na.rm = TRUE)
mediane
```
(25)

Troisièmement, l'écart-type est une mesure de dispersion qui donne un aperçu du type de distribution avec laquelle on travaille. Les observations sont-elles concentrées autour de la moyenne, ou davantage évasées vers les extrêmes ? C'est d'ailleurs la mesure qui est multipliée par 1.96 afin de déterminer la marge d'erreur d'un échantillon, tel que présenté dans la figure 6. L'écart-type se calcule par la fonction *sd()* avec les mêmes arguments que la moyenne et la médiane, comme suit :

```
ecart_type = sd(base$income, na.rm = TRUE)
ecart_type
```
(26)

Quatrièmement, pour les valeurs minimale et maximale de la distribution, celles-ci s'obtiennent assez simplement avec les fonctions *min()* et *max()* :

```
min = min(base$income, na.rm = TRUE)
min
max = max(base$income, na.rm = TRUE)
max
```
(27)

Si vous avez exécuté les fonctions précédentes avec la variable de revenu « income » non recodée et non traitée, vous obtiendrez des résultats un peu farfelus comme un minimum de -99 et un maximum de 100000000. **Il s'agit d'un très bon exemple de la raison pour laquelle il est primordial de bien vérifier la manière dont notre variable est codée dans R et les valeurs possibles qu'elle peut prendre.** Vous pouvez vous référer aux sections 5.5 et 5.6 du présent guide pour plus d'informations à ce sujet.

8 L'analyse bivariée : mesurer la relation statistique entre deux variables

Lorsque nous développons une hypothèse concernant un phénomène, celle-ci cherchera à tester une relation entre un minimum de deux variables. Elle émane d'une réponse provisoire à une question de recherche. À partir des données tirées de la base importée à la section 4.6, commençons par formuler quatre hypothèses de recherche qui serviront aux trois types de croisement distincts présentés dans cette section :

H1 : Une personne canadienne a plus de chances de voter si elle est une femme.

H2 : Au Canada, une femme a un revenu moyen inférieur à celui d'un homme.

H3 : Au Canada, les personnes qui ont un plus grand intérêt pour la politique ont un revenu moyen plus élevé que celles qui ont un faible intérêt pour la politique.

H4 : Plus l'âge d'une personne canadienne est élevé, moins elle a tendance à apprécier les personnes musulmanes qui vivent sur le territoire.

Une fois les hypothèses de recherche posées, on doit tester la relation entre les variables concernées. Pour ce faire, nous allons toujours suivre les quatre mêmes étapes de manière systématiques, en dans cet ordre :

1. Le recodage des variables, si nécessaire (voir section 5.5) ;
2. La présentation visuelle des données ;
3. Le test de significativité de la relation ($p < 0.05$) ;
   (a) Le seuil de 0.05 pour le p fait référence au niveau de confiance de 95% expliqué dans la figure 6. Nous cherchons donc à savoir si ce qui se trouve à l'extérieur de l'intervalle de confiance représente moins de 5% de la distribution. De cette manière, nous pouvons être sûr d'avoir confiance en nos résultats et ce, 19 fois sur 20.
4. La mesure de la taille de l'effet (selon le type des variables croisées).

Il existe trois croisements possibles en fonction des types de nos variables à analyser et la présente section sera divisée de manière analogue à ces croisements :

1. Un croisement entre deux variables qualitatives ;
2. Un croisement entre une variable qualitative et une variable quantitative ;
   (a) Lorsque la variable qualitative est dichotomique (codée 1, 0) ;
   (b) Lorsque la variable qualitative est polytomique et ordinale (codée 1, 2, 3, etc.) ;

3. Un croisement entre deux variables quantitatives.

Voici un tableau sommaire qui présente les mesures de la taille de l'effet à utiliser en fonction du type de croisement et les bornes inférieures qui déterminent la taille de l'effet (si ledit effet est significatif [$p < 0.05$]). La mesure de la taille de l'effet doit **au moins** atteindre le chiffre indiqué dans la colonne « Petite » pour considérer que la force de la relation est petite. Si elle est en dessous de cette borne, on la considérera **triviale** (qui signifie banale). La force de la relation entre les deux variables devient moyenne si la mesure obtenue est plus grande que la borne de la colonne « Moyenne », et le même principe s'applique pour la colonne « Grande ».

TABLEAU 3 : Force de la relation entre deux variables selon le type de croisement

Type de croisement	Mesure de taille de l'effet	Petite	Moyenne	Grande
Quali + Quali	V de Cramer	0.1	0.3	0.5
Quali (dichotomique) + Quanti	D de Cohen	0.2	0.5	0.8
Quali (polytomique) + Quanti	tau de Kendall	(-) 0.1	(-) 0.3	(-) 0.5
Quanti + Quanti	r de Pearson	(-) 0.1	(-) 0.3	(-) 0.5

8.1 *Croisement entre deux variables qualitatives*

Les données qualitatives se croisent assez simplement puisqu'elles contiennent un très faible nombre de valeurs possibles par rapport aux données quantitatives. Une variable dichotomique comme « l'action de voter », par exemple, contient seulement deux valeurs possibles : 1 et 0. Bien qu'elles soient codées avec des valeurs chiffrées, la nature de ces données reste textuelles, qualitatives. Les chiffres sont seulement utilisés pour simplifier l'analyse informatique des données. Il est donc primordial de ne pas s'arrêter à l'allure des données dans une base, mais de plutôt réfléchir à ce que ces données représentent concrètement, réellement. Tentons, dans cette sous-section, de vérifier la véracité de l'hypothèse **H1**, soit : « Une personne canadienne a plus de chances de voter si elle est une femme ».

8.1.1 Recodage des variables

Avant de croiser deux variables qualitatives, il est de mise d'effectuer un **traitement** des données sélectionnées afin de retirer les valeurs indésirables et les recoder en « NA ». Reprenons toujours les

données de la base importée à la section 4.6, et croisons les variables impliquées dans l'hypothèse **H1**. Nous allons donc procéder au recodage des variables « sex.r » et « p.voted ». Faites un copier-coller des lignes de code suivantes et exécutez-les :

```
base$a_vote = recode(base$p.voted, "c(1)=1 ; c(2)=0 ; else=NA")
base$femme = recode(base$sex.r, "c(1)=0 ; c(2)=1 ; else=NA")
```
(28)

8.1.2 Présentation visuelle des données

Une fois les valeurs des deux variables recodées, nous pouvons maintenant procéder à la présentation visuelle du croisement. Puisque les données qualitatives ne contiennent qu'un nombre limité de valeurs possibles, nous allons les présenter dans des tableaux ou des diagrammes à bandes.

Premièrement, pour faire un tableau simple des valeurs absolues de notre distribution, nous pouvons utiliser la fonction *table()* en plaçant notre variable « Y » en premier argument pour qu'elle soit placée en rangées et notre variable « X » en second argument pour qu'elle soit placée en colonnes. Bien que les rangées et les colonnes ne soient pas clairement identifiées, dans cet exemple, la variable « femme » recodée est en colonnes (verticales) et la variable « a_vote » est en rangées (horizontales). Faites un copier-coller des lignes de code suivantes et observez le résultat :

```
t1 = table(base$a_vote, base$femme)
t1
```
(29)

Pour éviter d'avoir des rangées et des colonnes qui ne sont pas identifiées, il est possible de définir soi-même une fonction (voir section 3.1) qui permet de créer un tableau avec les noms des variables directement affichés pour remplacer la fonction *table()*. Vous pouvez copier-coller les lignes de code suivantes **AU TOUT DÉBUT** de votre script et les exécuter :

```
TableLabels = function(variableY, variableX)
{
  t1=table(variableY,variableX)
  names(dimnames(t1)) = c(as.character(deparse(substitute(variableY))),
                          as.character(deparse(substitute(variableX))))
  return(t1)
}
```
(30)

Après avoir exécuté ces lignes de code, vous pouvez maintenant utiliser la fonction *TableLabels()* de la même manière que la fonction *table()*. Copiez-collez les lignes suivantes et observez le résultat :

```
t1 = TableLabels(base$a_vote, base$femme)
t1
```
(31)

Deuxièmement, pour faire un tableau croisé avec des pourcentages de colonnes et de rangées, nous pouvons utiliser la fonction *CrossTable()* du package « gmodels » (celui-ci est inclus dans les packages activés à la section 3.2 du présent guide). La lecture d'un tableau croisé peut être assez ardue si l'on ne sait pas exactement où regarder. D'abord, chaque cellule du tableau contient trois valeurs. En ordre, celles-ci représentent : le nombre d'unités d'analyse en valeur absolue, le pourcentage de la colonne et le pourcentage de la rangée. Ensuite, pour lire adéquatement un tableau croisé, on doit systématiquement se poser les questions suivantes et suivre ces étapes :

1. Il faut d'abord différencier les deux variables que nous traitons et ce qu'elles représentent.
   (a) Quelle variable représente l'ensemble à analyser, la variable de groupement ? (généralement la variable indépendante)
   (b) Quelle variable représente la caractéristique à analyser en fonction de la variable de groupement ; ce qui mesure la différence que l'on souhaite observer entre les groupes ? (généralement la variable dépendante)

2. Il faut ensuite observer le tableau croisé et déterminer si la variable de groupement (variable indépendante) est placée en colonnes ou en rangées.
   (a) Si la variable de groupement (variable indépendante) est placée en colonnes, on regardera le pourcentage de colonnes (3^{e} chiffre de la cellule) ;
   (b) Si la variable de groupement (variable indépendante) est placée en rangées, on regardera le pourcentage de rangées (2^{e} chiffre de la cellule).

En gardant ces étapes en tête et en se basant toujours sur l'hypothèse émise précédemment (**H1**), on souhaite comparer le pourcentage de femmes qui ont voté par rapport au total des femmes de la distribution, et le comparer au pourcentage d'hommes qui ont voté par rapport au total des hommes de la distribution. Notre variable de groupement, qui divise la distribution en deux ensembles distincts, se trouve alors à être la variable « femme ». Si la variable femme est en colonnes, nous regarderons le pourcentage de colonnes et si elle est en rangées, nous regarderons le pourcentage de rangées. Faites un copier-coller de la ligne suivante et prenez le temps d'effectuer ces étapes en observant attentivement le tableau qui apparaît :

```
CrossTable(base$a_vote, base$femme, prop.t=FALSE, prop.chisq=FALSE)
```
(32)

Dans le présent exemple, puisque la variable « femme » se trouve en colonnes, nous allons regarder le pourcentage de colonnes (3^{e} chiffre de la cellule). Ensuite, puisque nous voulons savoir la proportion qui a voté, nous regarderons les cellules de la rangée codée « 1 ». Sur ce tableau croisé, on observe que 87.3% des femmes de la distribution ont voté (cellule femme=1 et a_vote=1), contrairement à 88.4% des hommes de la distribution (cellule femme=0 et a_vote=1), ce qui semble infirmer **H1**.

8.1.3 Test de significativité ($p < 0.05$)

Bien que ce premier croisement nous donne un aperçu que notre hypothèse de départ puisse être fausse, il est nécessaire d'effectuer un test de significativité afin de déterminer si la relation entre les deux variables est significative ou non. Pour un croisement entre deux variables qualitatives, le « p » s'obtient en utilisant la fonction *chisq.test()* avec notre tableau précédemment créé en

seul argument. Faites un copier-coller de la ligne de code suivante et tentez de trouver l'indicateur « p-value » dans le résultat :

```
chisq.test(t1)
```
(33)

Dans cet exemple, le « p-value » est de 0.4452. Étant beaucoup plus élevé que le seuil de 0.05, la relation entre le sexe et l'acte de voter n'est donc pas significative. On peut alors dire qu'il n'y a pas de lien statistique entre les deux variables).

8.1.4 Mesure de la taille de l'effet

Une fois le « p » obtenu et la significativité de la relation déterminée, la prochaine étape permet de mesurer la taille de la relation entre les deux variables. En se fiant au tableau 3, nous pouvons calculer le « V de Cramer » et obtenir la taille de l'effet en utilisant la fonction *assocstats()* avec le tableau créé précédemment comme seul argument :

```
assocstats(t1)
```
(34)

Ici, le « V de Cramer » est de 0.018. Même si la relation était significative, la force de celle-ci serait seulement triviale. Les mesures du « p » et du « V de Cramer » que nous avons obtenues nous permettent donc **d'infirmer l'hypothèse H1**, ce qui nous indique qu'il n'y a pas de lien statistique entre le sexe et l'acte de voter.

8.2 *Croisement entre une variable qualitative et une quantitative*

Lorsque nous croisons une variable qualitative et une variable quantitative, maintenant, le processus est légèrement différent du précédent. Puisqu'une des deux variables est quantitative, la présentation des données sous forme d'un tableau ou d'un tableau croisé devient relativement impossible. On pourrait tenter de le faire, mais on réalisera rapidement qu'il y a un nombre beaucoup trop grand de rangées ou de colonnes puisque la variable quantitative peut prendre presque une infinité de valeurs uniques.

Pour cette raison, le croisement entre une variable qualitative et une variable quantitative implique que la variable qualitative devient une variable de groupement, et que nous observerons la différence entre les moyennes de la variable quantitative selon les différents groupes. Le nombre de groupes à comparer dépendra du nombre de valeurs que peut prendre la variable qualitative. Si elle est dichotomique (1, 0), nous comparerons la moyenne entre les deux groupes à l'aide du « D de Cohen » et si elle est polytomique et ordinale (1, 2, 3, etc.), nous comparerons la moyenne entre les différents groupes à l'aide du « tau de Kendall ».

Dans cette sous-section, nous tenterons de confirmer ou d'infirmer les hypothèses **H2** et **H3** afin de tester les deux croisements possibles entre une variable qualitative et une variable quantitative. Les variables à traiter seront le sexe « femme » et le revenu « income » pour **H2**, et l'intérêt pour la politique « polint » et le revenu « income » pour **H3**.

8.2.1 Recodage des variables

Comme il a été question pour un croisement entre deux variables qualitatives, il est de mise d'effectuer un **traitement** des données sélectionnées afin de retirer les valeurs indésirables et les recoder en « NA ». Reprenons toujours les données de la base importée à la section 4.6 et recodons les variables présentées précédemment. La variable de l'intérêt pour la politique sera catégorisée en quatre classes distinctes, « 1 » étant l'intérêt le plus faible et « 4 » l'intérêt le plus fort. La variable « sex.r » est recodée de la même manière qu'à la sous-section précédente et la variable « income » est recodée afin de retirer les gens qui gagnent moins de 1000$ et ceux qui gagnent plus de 1000000$. Les lignes de code suivantes effectuent ces modifications :

```
base$femme = recode(base$sex.r, "c(1)=0 ; c(2)=1 ; else=NA")
base$income[base$income < 1000 | base$income > 1000000] = NA))
base$polint = recode(base$p.intpol, "c(1,2,3)=1 ; c(4,5)=2 ;
                     c(6,7)=3 ; c(8,9,10)=4 ; else=NA")
```
(35)

8.2.2 Présentation visuelle des données

Puisque ce type de croisement sert à comparer les moyennes de différents groupes, il convient d'abord de calculer la moyenne de « Y » selon les valeurs possibles de « X ». Cette manipulation se fait en utilisant la fonction *tapply()*, en spécifiant en troisième argument que la fonction utilisée est la moyenne et en ajoutant « na.rm = TRUE » en quatrième argument afin de ne pas considérer les données manquantes dans le calcul. Voici les lignes de code qui permettent d'obtenir la moyenne du revenu en fonction du genre, puis la moyenne du revenu en fonction de l'intérêt pour la politique :

```
t2 = tapply(base$income, base$femme, FUN = mean, na.rm = TRUE)
t2
t3 = tapply(base$income, base$polint, FUN = mean, na.rm = TRUE)
t3
```
(36)

Les données obtenues nous montrent d'abord, pour **H2**, que le revenu moyen des femmes répondantes est d'environ 63000$ alors que celui des hommes répondants est de 69000$. Pour **H3**, ensuite, elles nous montrent que le revenu moyen des personnes les moins intéressées par la politique est de 56814$ alors que celui des personnes les plus intéressées par la politique est de 73053$. Ces données se présentent visuellement sur un diagramme à bandes avec les groupes en « X », soit le genre ou l'intérêt pour la politique, puis la moyenne des groupes en « Y », soit le revenu moyen. Les lignes de code suivantes permettent de créer deux diagrammes à bandes distincts et d'inscrire le revenu moyen de chaque groupe au-dessus des bandes :

```
graph1 = barplot(t2, xlab = "Groupes", ylab = "Moyennes",
                 main = "Titre du graphique", ylim = c(0, max(t2+t2/10)))
text(graph1, t2+(t2/20), labels = as.character(round(t2,2)))
graph2 = barplot(t3, xlab = "Groupes", ylab = "Moyennes",
                 main = "Titre du graphique", ylim = c(0, max(t3+t3/10)))
text(graph2, t3+(t3/20), labels = as.character(round(t3,2)))
```

(37)

Une fois les données présentées visuellement, il reste à déterminer la significativité de la relation entre les deux variables et la taille de l'effet. Nous débuterons d'abord par le croisement entre le sexe et le revenu, viendra ensuite le croisement entre l'intérêt pour la politique et le revenu.

8.2.3 Pour une variable qualitative dichotomique (1, 0)

Puisque la variable « femme » impliquée dans l'hypothèse **H2** est dichotomique, nous utiliserons la fonction *t.test()*, ou un « test de Student », pour obtenir le « p » et ainsi déterminer si la différence de revenu entre les deux groupes est significative. La mesure peut-être relativement difficile à cibler dans le résultat affiché dans la console, mais il suffit de regarder la valeur du « p-value » qui se situe en haut à droite. Ensuite, la fonction *cohensD()* calcule le « D de Cohen », ce qui nous permet de déterminer la taille de l'effet entre les deux variables (selon les bornes présentées au tableau 3). Faites un copier-coller des lignes de code suivantes et observez le résultat :

```
t.test(base$income ~ base$femme)
cohensD(base$income ~ base$femme)
```
(38)

Dans cet exemple, la fonction *t.test()* nous montre que le « p » est égal à 0.00282, donc plus petit que 0.05, ce qui signifie que la relation entre le sexe et le revenu est significative. Lorsque l'on observe la valeur du « D de Cohen » avec la fonction *cohensD()*, toutefois, on s'aperçoit que celui-ci n'est que de 0.087. En comparant cette valeur aux bornes présentes dans le tableau 3, on peut arriver à la conclusion que malgré le fait que la relation entre les deux variables est significative, celle-ci n'a qu'une force triviale ou banale. **On peut alors infirmer l'hypothèse H2**.

8.2.4 Pour une variable qualitative polytomique ordinale (1, 2, 3, etc.)

Pour ce qui est de l'hypothèse **H3**, puisque la variable d'intérêt pour la politique « polint » est divisée est plus de deux catégories ordinales (polytomique), nous utiliserons la fonction *cor.test()* à la manière d'un croisement entre deux variables quantitatives, mais nous spécifierons en troisième argument que la méthode utilisée est celle de « Kendall ». Cette fonction nous permettra d'obtenir

à la fois le « p » et le « tau de Kendall », qui mesure la taille de l'effet selon les bornes du tableau 3. Vous pouvez copier-coller la ligne de code suivante et observer le résultat :

```
cor.test(base$polint, base$income, method = "kendall")
```
(39)

Ici, la fonction *cor.test()* nous indique que le « p » est égal à 5.81e-05, ou 0.0000581, ce qui est plus petit que 0.05. La relation entre l'intérêt pour la politique et le revenu est donc significative. Cependant, si l'on observe la valeur du « tau », qui est égal à 0.056, et qu'on la compare aux bornes du tableau 3, on peut conclure que la relation entre ces deux variables est triviale. **L'hypothèse H3 est donc infirmée**.

8.3 *Croisement entre deux variables quantitatives*

Puisque les données quantitatives possèdent généralement un nombre presque infini de valeurs possibles, leur traitement diverge légèrement de celui des variables qualitatives. Il est, par exemple, impossible d'effectuer un tableau ou un tableau croisé avec des données quantitatives, puisque ceux-ci contiendraient un nombre ridiculement grand de colonnes et de rangées. Nous allons plutôt effectuer les étapes suivantes pour les données quantitatives.

8.3.1 Recodage des variables

Avant de croiser deux variables quantitatives entre elles, il est de mise d'effectuer un **traitement** des données sélectionnées afin de ne conserver que les valeurs désirées. Reprenons toujours les données de la base importée à la section 4.6, et croisons les variables impliquées dans la quatrième hypothèse (**H4**). Nous allons donc procéder au recodage des variables « age » et « feel.musl » en recodant la variable « age » en âge réel plutôt qu'en année de naissance,[8] et en retirant les personnes qui ont répondu « 1000 » à la question sur l'âge et sur l'échelle de 0 à 100 pour l'appréciation des musulmans. Faites un copier-coller des lignes de code suivantes et exécutez-les :

[8] ATTENTION : Utilisez plutôt la variable « an_naissance » si vous aviez déjà renommé la variable « age » aux sections 5.4 et 5.6. S'il y a des erreurs dans vos données en raison de multiples traitements, réimportez la base de données et réeffectuez le recodage présent à la section actuelle.

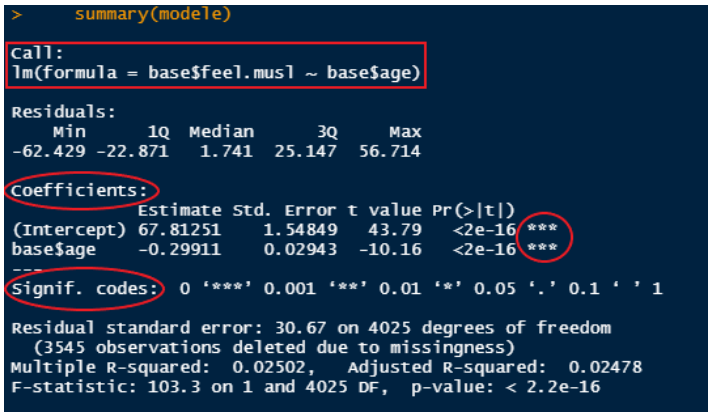
```
base$age[base$age == 1000] = NA
base$age = 2015 - base$age
base$feel.musl = recode(base$feel.musl, "c(1000) = NA")
```
(40)

8.3.2 Présentation visuelle des données

Pour un croisement entre deux variables quantitatives, les données doivent être présentées sur un graphique par un nuage de points des valeurs réelles et une droite de régression qui offre un aperçu des valeurs estimées par le modèle. Ici, le graphique de base est créé par la fonction *plot()*, le modèle formant la droite de régression par la fonction *lm()*, sa formule est affichée par la fonction *summary()* puis la droite de régression est affichée sur le graphique avec la fonction *abline()*. Faites un copier-coller des lignes de code suivantes et prenez le temps d'observer le résultat de chacune d'entre elles, tant sur le graphique que dans la console :

```
plot(base$age, base$feel.musl)
modele = lm(base$feel.musl ~ base$age)
summary(modele)
abline(modele)
```
(41)

Ici, on observe que la droite de régression a une pente descendante. La fonction *summary()* nous donne une série d'informations pertinentes sur le modèle de régression, entre autres dans le petit tableau situé juste en dessous de « Coefficients : » dans la console. Prenez le temps d'observer toutes les mesures qui suivront sur le résultat qui devrait ressembler à ceci :



On peut ici connaitre la mesure de cette pente en observant la valeur de la colonne « Estimate » associée à la rangée de notre variable indépendante, ici « base$age ». On observe alors que la pente de cette droite est de « -0.299 », ce qui signifie que pour chaque augmentation de 1 an, l'appréciation des musulmans diminue de 0.3 sur une échelle de 0 à 100. On peut aussi connaître l'ordonnée à l'origine de la droite de régression en regardant la colonne « Estimate » du tableau associée à la rangée « (Intercept) ». Dans cet exemple, la valeur de « Y » lorsque « X = 0 » est de « 67.81 ». La pente et l'ordonnée à l'origine de la droite de régression nous permettent d'obtenir la formule de ladite droite, où « a » représente la pente et « b » l'ordonnée à l'origine :

```
y = a*x + b
```
(42)

Sinon, les étoiles présentes à la droite du tableau sont une indication visuelle de la significativité de la relation entre les deux variables, ou la valeur du « p ». Pour les interpréter, on peut se fier aux bornes situées juste en dessous du tableau, à la rangée intitulée « Signif. codes : ». Les trois étoiles présentes à la rangée de notre variable signifient donc que la relation entre celle-ci et la variable dépendante « feel.musl » est très significative, avec un « p » qui se situe en dessous de « 0.001 ».

8.3.3 Test de significativité ($p < 0.05$) et mesure de la taille de l'effet

Pour un croisement entre deux variables quantitatives, bien que le modèle de régression nous donne plusieurs informations importantes, le « p », le coeffictient de corrélation « r de Pearson » et le coefficient de détermination « R^2 » peuvent s'obtenir tous les trois à partir de la fonction « cor.test() », en utilisant la méthode de Pearson. Il ne suffit qu'à inclure la variable indépendante en premier argument et la variable dépendante en second, comme suit :

```
cor = cor.test(base$age, base$feel.musl, method = "pearson")
cor$p.value
cor$estimate
(cor$estimate^2)*100
```
(43)

La fonction précédente nous offre le « p-value » et le coefficient de corrélation (r de Pearson) affiché « cor » dans la console lorsque nous exécutons le nouvel objet qui a été nommé « cor ». Les trois lignes qui suivent permettent d'afficher dans la console les mesures pertinentes en ordre, soit le « p », le « r de Pearson » et le « R^2 ».

Dans cet exemple, on peut d'abord observer que le « p » est de « 5.63e-24 », ou 0.000000000000000000000005633943 sans la notation scientifique. Celui-ci étant beaucoup plus petit que 0.05, on peut affirmer que la relation entre les deux variables est significative.

Après avoir déterminé qu'il y a un lien statistique entre l'âge et l'appréciation des personnes musulmanes à l'aide du « p », le coefficient de corrélation « r de Pearson » et le coefficient de détermination « R^2 » nous permettent de mesure l'intensité de la relation entre les deux variables, ou la taille de l'effet. En se fiant aux bornes présentes dans le tableau 3, le « r de Pearson », d'abord, nous indique si la relation est positive ou négative (proportionnelle ou inversement proportionnelle) par la présence ou l'absence du négatif devant le chiffre. Ensuite, les bornes du tableau nous indiquent la force de la relation.

Ici, le coefficient de corrélation obtenu est de « -0.158 », ce qui indique une corrélation petite et inversement proportionnelle. Le coefficient de détermination, quant à lui, est de « 2.50% », ce qui signifie que l'âge expliquerait 2.5% de la variation de l'appréciation pour les personnes musulmanes. **Ces données nous permettent donc de confirmer l'hypothèse H4**, et d'affirmer que plus l'âge d'une personne canadienne est élevée, moins elle apprécie les personnes musulmanes.

8.4 *Réflexions générales sur les croisements*

Il est normal de ne pas tomber sur des résultats extrêmement significatifs lors d'analyses bivariées puisqu'en sciences sociales, les phénomènes sont généralement expliqués par une plus grande quantité de variables. Bien que l'âge ait un léger impact inversement proportionnel sur l'appréciation des personnes musulmanes, nous savons pertinemment qu'il ne s'agit pas du seul facteur explicatif, et qu'une personne plus âgée n'a pas nécessairement une opinion défavorable des personnes musulmanes. Les relations ne sont jamais absolues et parfaites.

Il est aussi normal de tomber sur des résultats qui ne sont pas significatifs du tout. Parfois, cette situation est due à l'échantillonnage, parfois à d'autres facteurs explicatifs. Ces exemples démontrent bien les limites de l'analyse bivariée, alors que **H1** est infirmée puisqu'elle est non significative, **H2** et **H3** sont infirmées puisque la force de la relation n'est que triviale, et **H4** est la seule hypothèse qui est confirmée, mais avec une corrélation faible.