



Université Laval

## ***RStudio Cloud*** pour les nuls

POL-2000, Méthodologie quantitative

---

# Table des matières

<b>1 C'est quoi R?</b>	<b>2</b>
<b>2 R vs. RStudio vs. RStudio Cloud</b>	<b>2</b>
<b>3 RStudio Cloud – La base</b>	<b>3</b>
3.1 Connexion . . . . .	3
3.2 Interface . . . . .	5
<b>4 HelloWorld</b>	<b>7</b>
4.1 Workflow . . . . .	7
4.1.1 Arborescence . . . . .	8
4.1.2 Commentaires . . . . .	8
4.1.3 Raccourcis clavier . . . . .	8
<b>5 Intro à la programmation en R</b>	<b>9</b>
5.1 Ça va bien aller . . . . .	9
5.2 Opérateurs . . . . .	10
5.2.1 Opérateurs de calcul . . . . .	10
5.2.2 Opérateurs d'assignement . . . . .	11
5.2.3 Opérateurs logiques . . . . .	11
5.2.4 Instructions de contrôle . . . . .	11
5.3 Structure de données . . . . .	13
5.3.1 Types de données et constantes . . . . .	13
5.3.2 Vecteurs . . . . .	14
5.3.3 Data frames . . . . .	16
<b>6 Importation de données</b>	<b>17</b>
6.1 Importation de données . . . . .	17
<b>7 Analyse – Description</b>	<b>18</b>
7.1 Univariée . . . . .	18
7.2 Bivariée . . . . .	18
<b>8 Analyse – Régression linéaire simple et multiple</b>	<b>21</b>
<b>9 Visualisation</b>	<b>21</b>

---

# 1 C'est quoi R?

Bonjour et bienvenue dans le cours POL-2000! Ce tutoriel sera votre guide de démarrage ainsi qu'un document de référence tout au long du cours<sup>1</sup>. Dans l'objectif d'introduire les étudiants de sciences politiques aux méthodes quantitatives et à l'analyse causale en sciences sociales, nous avons cru bon de vous initier au langage de programmation R. N'ayez crainte, c'est plus simple qu'il n'y paraît et vous en tirerez beaucoup d'avantages.

Pour la petite histoire, la première version de **R** a été publiée en 1995 par Ross Ihaka et Robert Gentleman, mais le langage s'inspire des travaux de John Chambers aux laboratoires Bell dans les années 1970. Aujourd'hui, **R** est un outil d'analyse statistique populaire, tant dans le secteur privé que dans le monde universitaire. **R** est ce que l'on appelle un *logiciel libre*, ce qui signifie que son code source est ouvert. Ceci permet à des utilisateurs bénévoles de développer des *packages* (micrologiciel ou librairie de fonctions) qui sont ensuite rendus disponibles à la communauté (pour la plupart gratuitement). Ceci fait de **R** un outil puissant, flexible, public, et donc particulièrement adapté à la méthode scientifique.

Le reste du document vous permettra de vous familiariser avec **R** et avec son environnement de travail. Nous encourageons donc sa lecture attentive.

## 2 R vs. RStudio vs. RStudio Cloud

Une distinction importante à effectuer est la différence entre le langage **R** et l'*IDE*<sup>2</sup> **RStudio**. L'*IDE* a pour fonction principale de recevoir le code et de le compiler. En d'autres mots, **R** c'est la langue que l'on écrit et le papier c'est l'*IDE*. Plusieurs *IDE* existent et il est facile de se perdre dans leurs différents paramètres et fonctionnalités. C'est pourquoi nous imposons l'utilisation de **RStudio** ou, à proprement parler, de **RStudio Cloud**. **RStudio Cloud** est une reproduction de l'environnement **RStudio** en ligne. De cette façon, tous les étudiants ont accès au même environnement de travail, peu importe l'ordinateur utilisé.

Ainsi, dans le cadre du cours, les étudiants utiliseront le langage de programmation **R** à partir de l'environnement **RStudio Cloud**. Cette distinction s'affinera au cours de la session, n'ayez crainte. Pour les plus curieux d'entre vous, de nombreuses ressources, francophones et anglophones, existent sur internet :

- [Stackoverflow](#)
  - Google est le meilleur ami des programmeurs. Si vous rencontrez un problème, Google vous fournira sans doute la solution sous la forme d'un *post* sur Stackoverflow. Il s'agit d'un site répertoriant les questions d'utilisateurs concernant la plupart des langages de programmation, incluant **R**. À la manière du logiciel libre, ce sont les autres utilisateurs du site qui se chargent de répondre avec grande précision. C'est vraiment un outil important.
- [r-bloggers](#)
  - Pour être informé sur les nouveaux développements de **R** et de RStudio. Encore une fois, il s'agit d'un point de rencontre de la communauté.
- [Coursera](#)
  - Site de formation en ligne. Les cours ont le format de cours universitaires, mais avec la version gratuite: pas besoin de suivre l'entièreté des plans de cours (ni de remettre les travaux).
- [DataCamp](#)
  - Similaire à Coursera, DataCamp se concentre sur des exercices pratiques en **Python**, **R** et **SQL**. Avec une approche très pratique, c'est un bon moyen d'accélérer l'intégration de connaissances techniques.
- [Datanovia](#)
  - Sous forme de tutoriel et de blogue, Datanovia est une excellente source d'information bilingue, spécialement lorsqu'il s'agit de visualisation de données.

---

<sup>1</sup> Tutoriel basé sur le travail de Vincent Arel-Bundock, Yannick Dufresne, et Florence Vallée-Dubois

<sup>2</sup> *Integrated development environment* ou environnement de développement intégré.

## 3 RStudio Cloud – La base

### 3.1 Connexion

Dans le cadre du cours, nous utiliserons un espace de travail commun. Pour y accéder, veuillez d'abord vous créer un compte **RStudio Cloud** en cliquant [ici](#). **N'oubliez pas de vous servir de votre courriel institutionnel (@ulaval.ca) pour ce faire.** Ce lien vous mènera à une page semblable à la figure 1a. Une fois votre compte créé, cliquez Le travail est à remettre en format PDF sur le portail au plus tard le 2022-22-04, à 23h59. Veuillez prendre note que vous devrez vous servir de l'interface R Studio Cloud pour exécuter ce travail. La présentation et la citation des sources seront prises en compte dans votre note. Vous pourrez prendre connaissance des consignes ici : <https://www5.bibl.ulaval.ca/services/redaction-et-citation/citation-de-sourcespour> accéder à l'environnement du cours. Si vous cliquez sur l'hyperlien dans le Portail du cours, vous accéderez à une page semblable à la figure 1b, mais les étapes sont les mêmes. Ce lien vous permettra d'accéder à tous les exercices et devoirs du cours qui se servent du logiciel R.

The image contains two side-by-side screenshots of the RStudio Cloud interface.

**Screenshot (a): Création du compte (Left)**

This screenshot shows the "Sign Up" page. It features a "Log In" link in the top left, a "Sign Up" button in the center, and four input fields for "Email", "Password", "First name", and "Last name". Below these is a large blue "Sign Up" button. At the bottom, there are "Sign Up with Google" and "Sign Up with GitHub" buttons.

**Screenshot (b): Inscription à l'environnement (Right)**

This screenshot shows the "Please log in or sign up to continue." screen. It includes a "Log In" button, a "Sign Up" button, an "Email" input field, a "Continue" button, and a "Forgot your password?" link. At the bottom, there are "Log In with Google" and "Log In with GitHub" buttons.

Figure 1: Connexion à **RStudio Cloud**

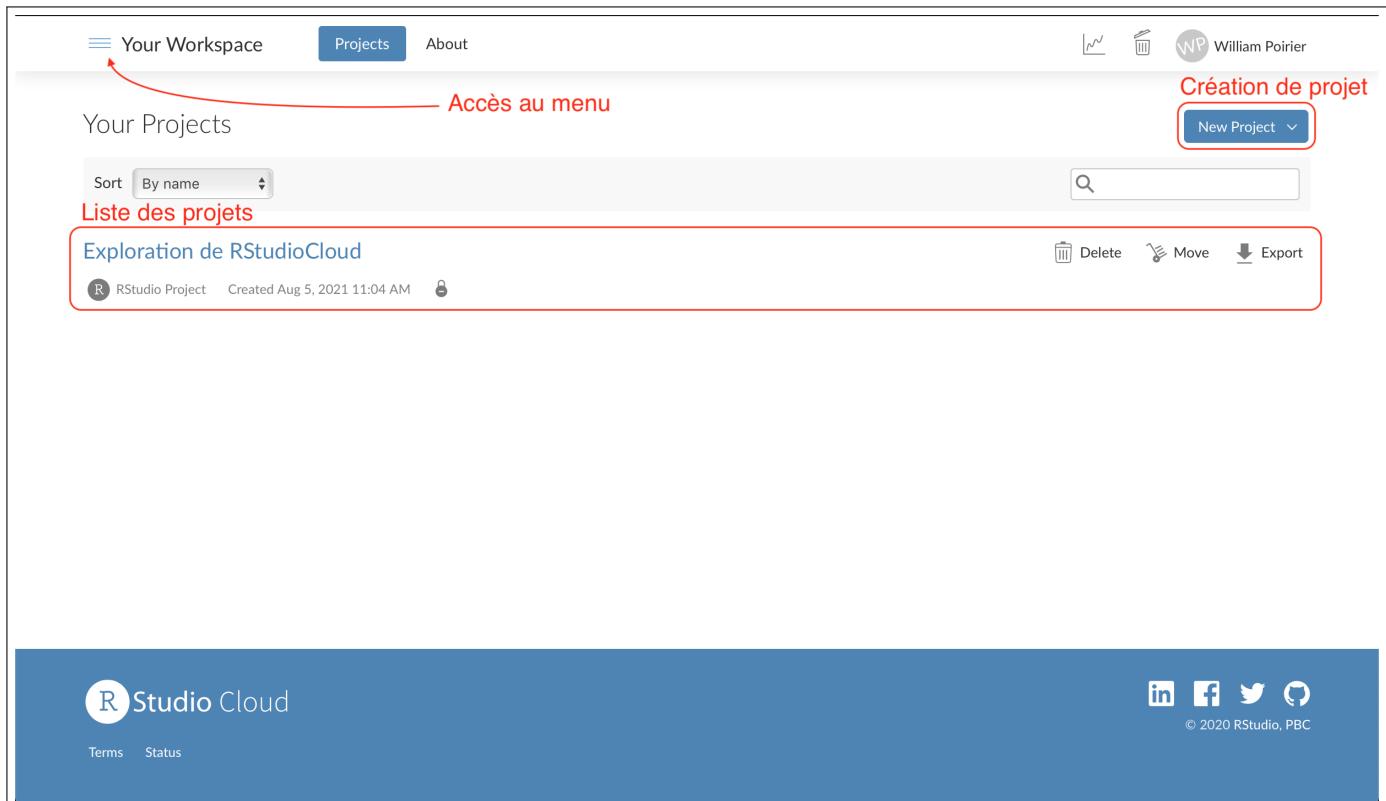


Figure 2: Page d'accueil

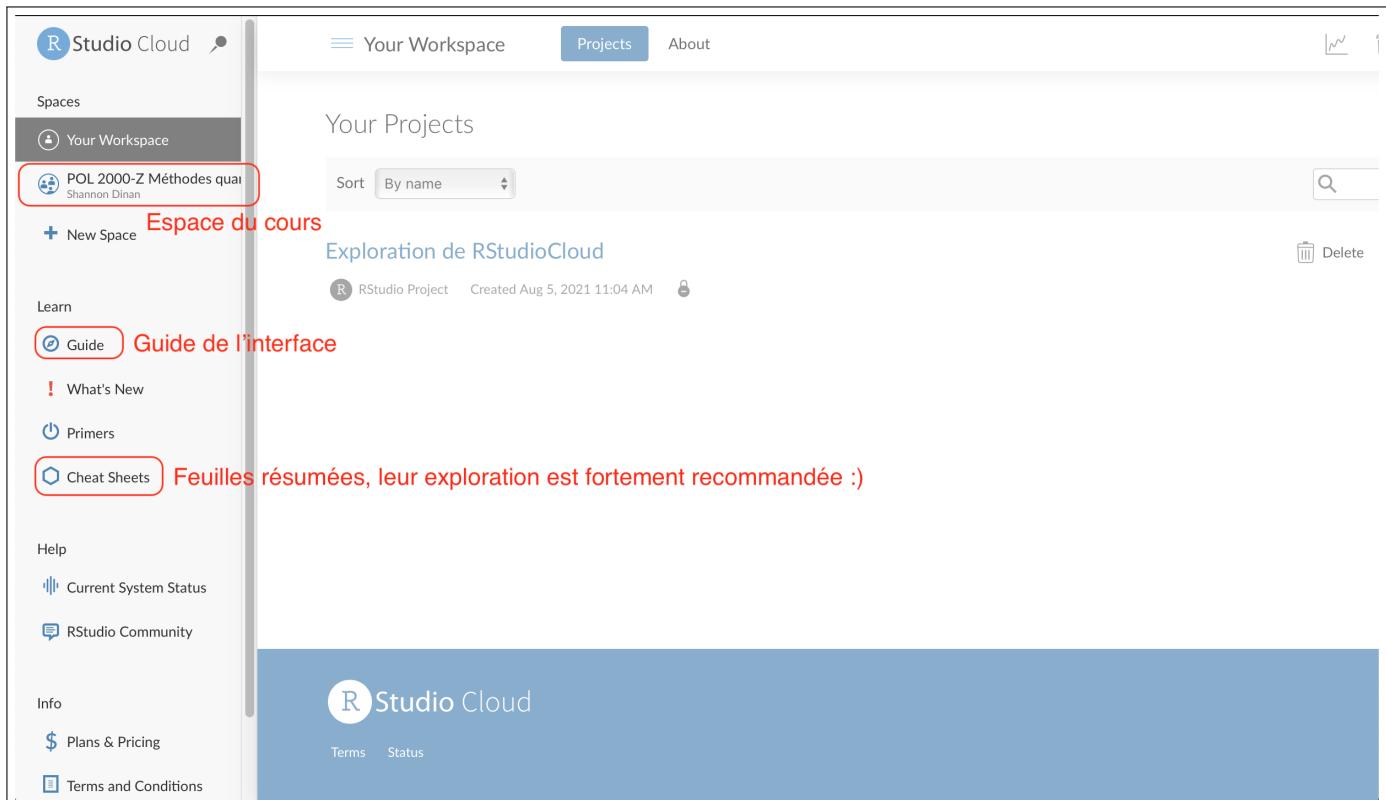


Figure 3: Menu d'accueil

## 3.2 Interface

**RStudio Cloud** a deux interfaces, l'interface des espaces de travail et l'interface des projets. L'interface des espaces de travail est la première chose que vous rencontrez en vous connectant à **RStudio Cloud**. C'est ce qui vous permet de vous déplacer d'un projet à l'autre et d'un espace de travail à l'autre. La section *menu* de cette interface offre aussi plusieurs liens menant à des ressources externes sur 1) l'utilisation de **RStudio Cloud**, 2) l'utilisation de **R** et 3) l'utilisation de ses *packages* les plus populaires. Pour faciliter votre compréhension, les figures 2 et 3 présentent un aperçu de l'interface des espaces de travail de **RStudio Cloud** et ses principaux points d'intérêt.

La seconde interface est celle qui nous intéresse le plus, c'est l'endroit où vous écrirez et exécuterez votre code. La figure 4 présente les points d'intérêt les plus importants. L'*environnement* contiendra les bases de données que vous importerez ainsi que tous les objets créés lors de la session<sup>3</sup>. Nous aborderons les bases de données dans une section ultérieure. En dessous de l'*environnement*, vous avez accès à vos dossiers, les librairies utilisées, un aperçu des graphiques produits ainsi qu'à de l'aide dans la fenêtre en bas à droite. La plus grande fenêtre par défaut est celle contenant la console. La **console** est un environnement d'exécution directe. En d'autres mots, vous pouvez y écrire des commandes qui seront exécutées immédiatement. C'est utile pour faire des tests et exécuter de petites manipulations. Or, la **console** n'enregistre pas la suite de commandes que vous lui demandez de faire, ce n'est pas son rôle. Pour ce faire, il faut ouvrir l'**éditeur** tel que spécifié par la figure 4. L'**éditeur**, c'est l'endroit où l'on écrit un code (une suite de commandes permettant d'atteindre un but quelconque, comme calculer une moyenne, faire une régression simple ou produire un graphique) afin qu'il puisse être enregistré et réutilisé. Dans le cadre de ce cours, vous allez principalement travailler dans l'**éditeur**.

Avant d'entrer dans la syntaxe d'utilisation de **R**, je recommande fortement la personnalisation de votre environnement. Non seulement cela vous permettra d'avoir une expérience esthétique plus agréable, cela vous aidera à long terme à reconnaître les structures du langage. Les figures 5 et 6 montrent comment accéder aux options de l'environnement global. Vous y trouverez ma mise en place préférée. Je tends à préférer coder sur fond plus foncé, mais je vous encourage à explorer les options et à trouver ce qui vous correspond le mieux. C'est pour vos yeux, pas ceux de l'équipe d'enseignement.

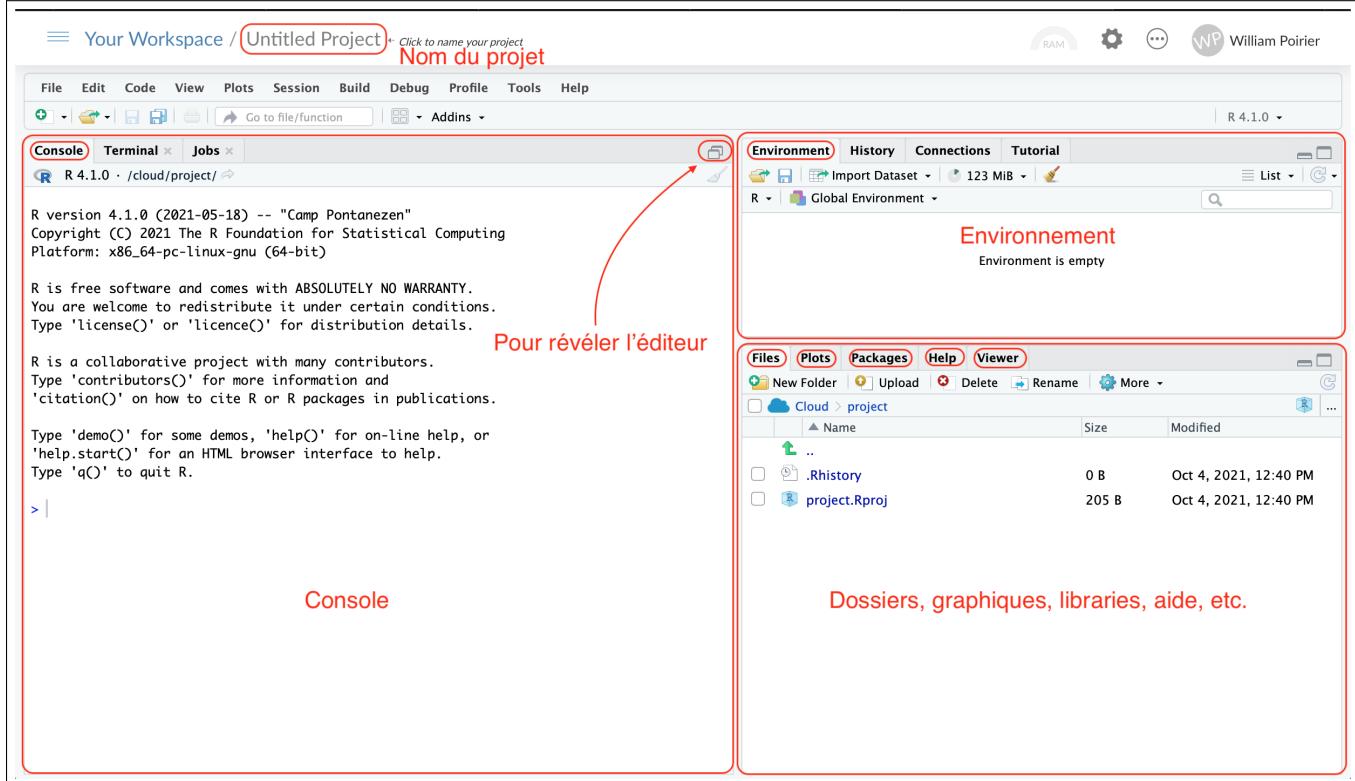


Figure 4: État par défaut

<sup>3</sup> Ici, session fait référence à la période de travail sur **RStudio Cloud**.

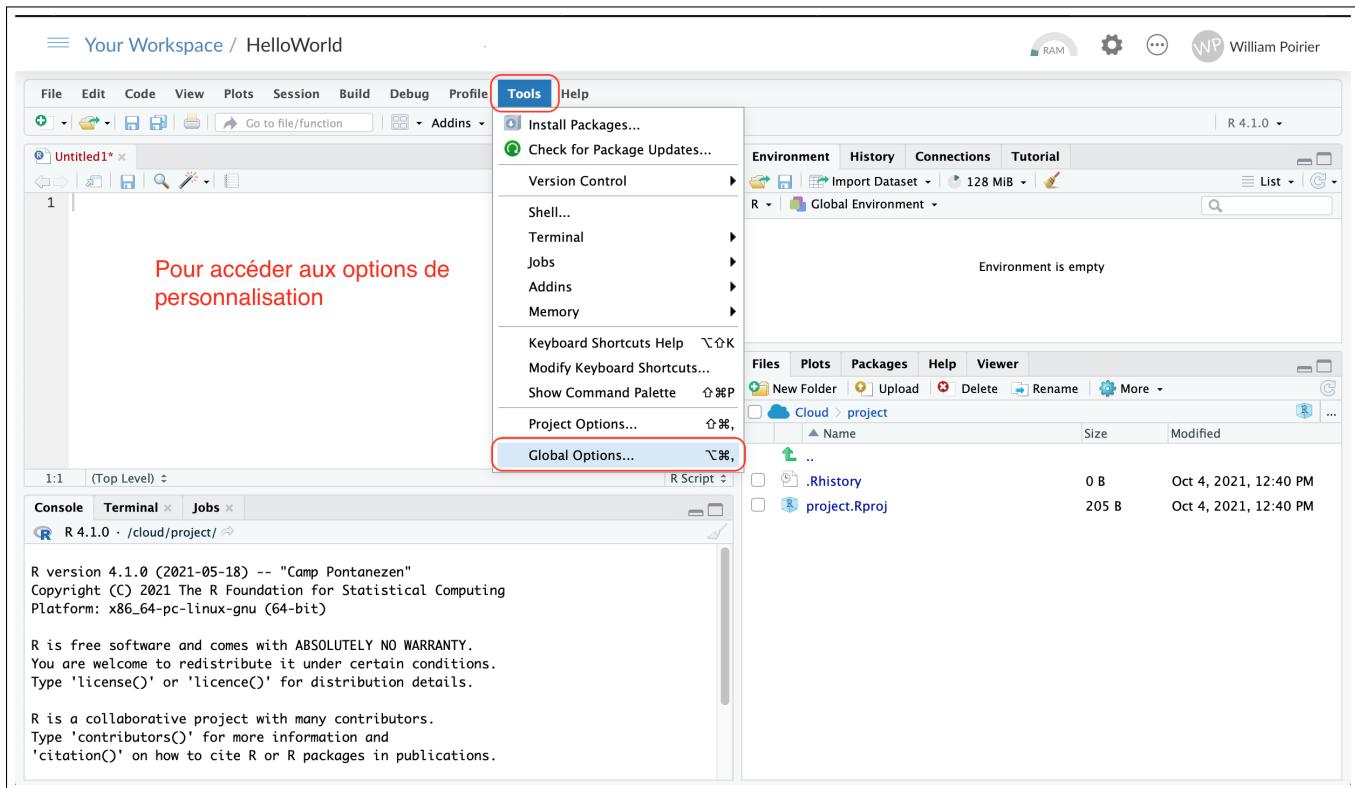


Figure 5: Début de la personnalisation

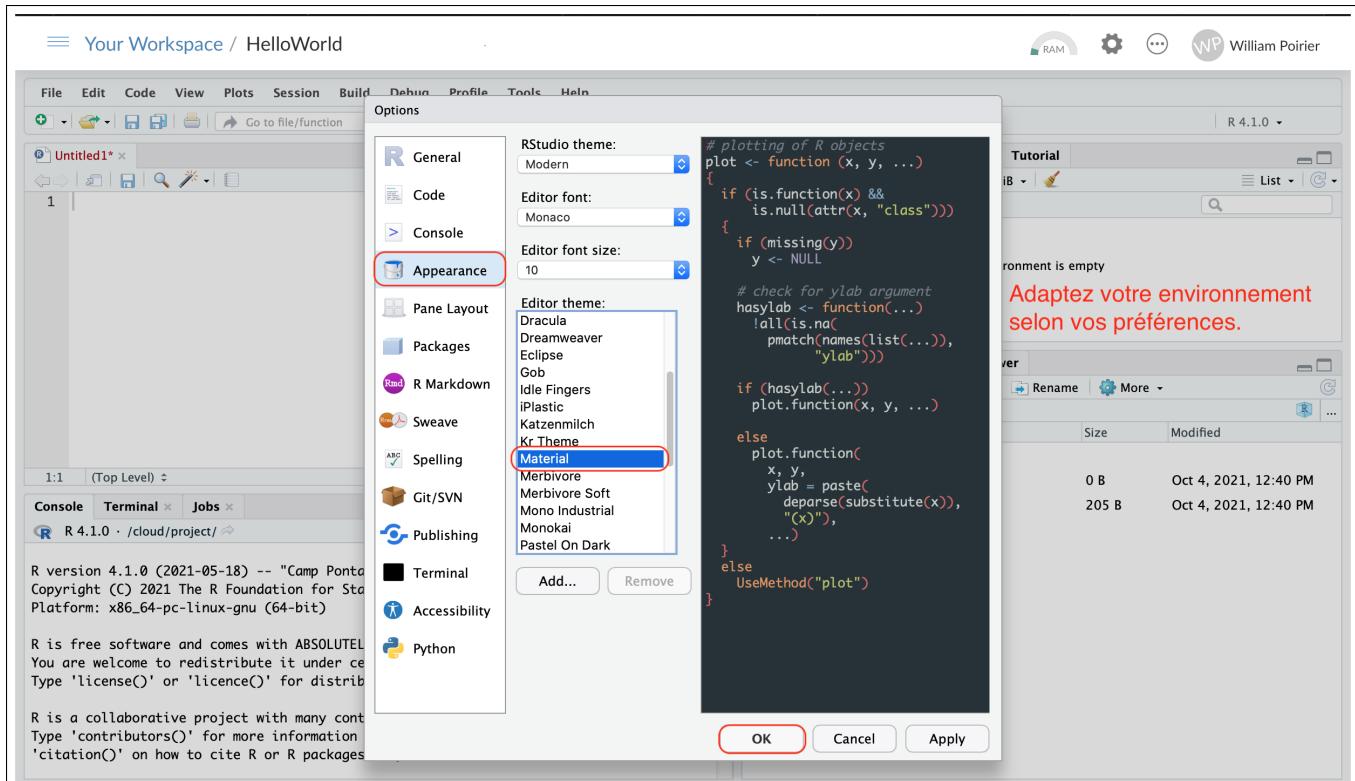


Figure 6: Fin de la personnalisation

## 4 HelloWorld

Vous en connaissez maintenant assez à propos de R pour rédiger votre première ligne de code! Allez dans l'éditeur, et tapez la commande suivante :

```
> print("HelloWorld!")
```

Comme vous pouvez voir, R est un gentil programme qui fait une partie du travail pour vous. Dès que vous ouvrez une parenthèse dans l'éditeur, le logiciel ajoute déjà la seconde parenthèse pour s'assurer qu'il n'y ait pas de message d'erreur. R fait la même chose pour les guillemets.

À présent, il faut exécuter votre ligne de code. Pour ce faire, sélectionnez la ligne, puis cliquez sur le bouton **RUN** en haut à droite. La figure 7 vous montre un exemple. Une fois exécuté, ce simple code imprimera n'importe quelle suite de caractères que vous aurez choisi. Amusez-vous!

Remarquez que le résultat s'affiche dans la console en bas. C'est sa deuxième utilité. La console vous indique les résultats des commandes que vous exécutez à partir de l'éditeur. Félicitations, vous êtes désormais des programmeurs, le fun peut commencer!

ATTENTION! Pour la suite du tutoriel, je vous invite tout de suite à repérer ces symboles sur votre clavier d'ordinateur : <, >, =, ", ', \$, |, &, %, [ , ], { , }.

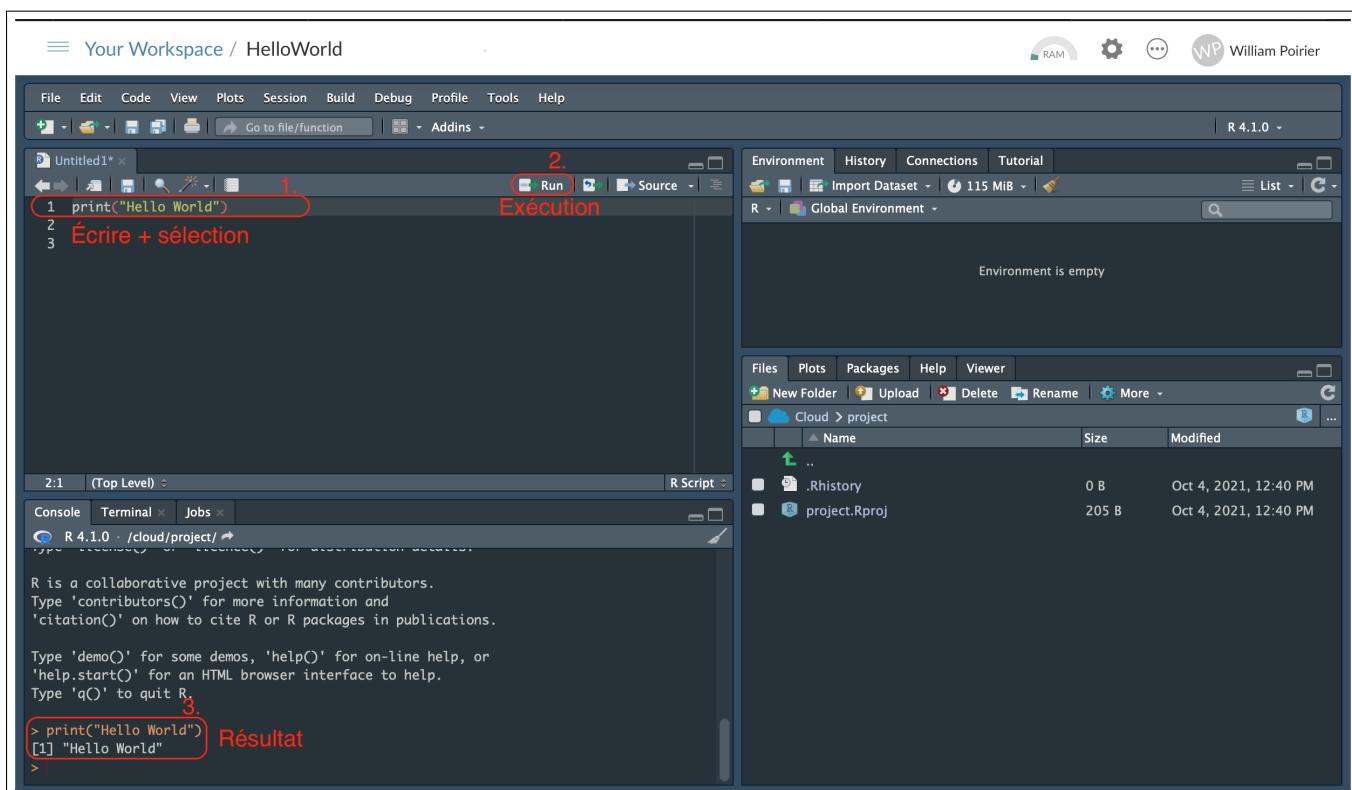


Figure 7: Votre première ligne de code!

### 4.1 Workflow

Un facteur important à prendre en considération lorsque l'on programme ou que l'on gère des bases de données, c'est l'organisation de notre « *flux de travail* ». Plusieurs éléments sont à considérer lorsque l'on veut optimiser son travail. Nous aborderons les principaux dans cette section. Ces habitudes peuvent vous aider à éviter des situations fâcheuses.

### 4.1.1 Arborescence

Si vous lisez le titre de cette section sans savoir ce à quoi nous faisons référence, ne vous inquiétez pas. La plupart des gens utilisent l’arborescence de leur ordinateur sans même s’en rendre compte. C’est tout simplement le chemin par lequel vous (ou votre ordinateur) devez passer pour accéder à un fichier. Comme quand vous voulez retrouver un vieux travail, votre ordinateur doit savoir où se trouvent les fichiers que vous mobilisez dans votre code. Pour ce faire, il faut établir le « répertoire de travail » (*working directory*) de la session **R**. On utilise alors la fonction `setwd()` et on y inscrit l’arborescence (le chemin) :

```
# Pour les Mac  
> setwd("/Users/nomDelaSession/nomDuDossier/nomDuSousDossier")  
# Pour les PC  
> setwd("C:/Users/nomDelaSession/nomDuSousDossier")
```

Remarquez la façon dont l’arborescence est écrite. Chaque dossier est suivi d’une barre oblique (/) et d’un sous-dossier. C’est donc important de bien organiser les dossiers sur son ordinateur de sorte à éviter de devoir changer de répertoire de travail trop souvent. L’idéal c’est de référer à un dossier général contenant les dossiers spécifiques. Par exemple, si vous travaillez sur un projet de recherche à propos des élections fédérales en 2021, vous pourriez avoir un dossier `ElectionFed21` qui contient un dossier `codeR`, un dossier `graphs` et un dossier `Data`. De cette façon, vous pouvez indiquer un répertoire de travail général au début de votre code pour ensuite utiliser des arborescences relatives. Ne vous inquiétez pas, tout ceci deviendra très simple au cours de la session.

### 4.1.2 Commentaires

Commenter un code c’est l’habitude la plus importante à prendre, surtout lorsqu’on est en apprentissage. Que voulons-nous dire par là? Imaginez que vous êtes en train d’explorer une forêt. Vous pourrez sans doute retrouver votre chemin de mémoire, mais l’idéal c’est de laisser des repères tout au long du chemin. Surtout si vous n’y retournerez pas avant un moment. Coder, c’est la même chose. Vous allez développer une suite d’instructions logiques qui permettra de produire un résultat spécifique. Il faut donc vous assurer de pouvoir bien comprendre les différentes étapes de votre code, et ce, même après plusieurs mois ou années. La mémoire étant une faculté qui oublie, ce n’est peut-être pas l’outil le plus fiable pour cette tâche. Donc, pour commenter, il vous suffit de précéder ce que vous écrivez par un `#`. Voici un exemple dans lequel nous avons divisé les réponses à une variable (qui variait entre 0 et 1) en trois catégories distinctes :

```
# Séparation en trois niveaux de la variable ses_health  
> Data$ses_health[Data$ses_health==0.75 | Data$ses_health==1] <- 1  
> Data$ses_health[Data$ses_health==0.5] <- 0.5  
> Data$ses_health[Data$ses_health==0.25 | Data$ses_health==0] <- 0
```

### 4.1.3 Raccourcis clavier

Écrire des lignes de code peut être frustrant, surtout lorsque vous n’êtes pas habitué à repérer certains symboles comme [ ] et { }. Comme dans toute chose, avec de l’expérience, vous allez acquérir de la rapidité. Par ailleurs, certains raccourcis clavier de **RStudio** et **RStudio Cloud** pourront vous éviter de perdre de précieuses secondes. Puisque ces raccourcis vont varier en fonction du système d’opération de votre ordinateur, les deux prochaines sections s’adressent aux utilisateurs Mac et PC respectivement.

#### 4.1.3.1 Raccourcis clavier – Utilisateurs Mac

1. `cmd + enter`, permet de rouler la partie du code sélectionné.
2. `option + sélection`, permet de sélectionner une portion de plusieurs lignes en même temps. Voir la figure 8.
3. `shift + flèches`, permet de sélectionner à l’aide des flèches (gauche, droite, bas, haut) du clavier.
4. `cmd + flèches`, permet d’accéder au début (avec la flèche du haut) et à la fin (avec la flèche du bas) d’un document.
5. `option + flèche`, permet d’inverser une ligne avec celle du haut (avec la flèche du haut) ou avec celle du bas (avec la flèche du bas).

#### 4.1.3.2 Raccourcis clavier – Utilisateurs PC

1. `ctrl + enter`, permet de rouler la partie du code sélectionné.
2. `alt + sélection`, permet de sélectionner une portion de plusieurs lignes en même temps. Voir la figure 8.
3. `shift + flèches`, permet de sélectionner à l'aide des flèches (gauche, droite, bas, haut) du clavier.
4. `ctrl + flèches`, permet d'accéder au début (avec la flèche du haut) et à la fin (avec la flèche du bas) d'un élément du code.
5. `alt + flèche`, permet d'inverser une ligne avec celle du haut (avec la flèche du haut) ou avec celle du bas (avec la flèche du bas).

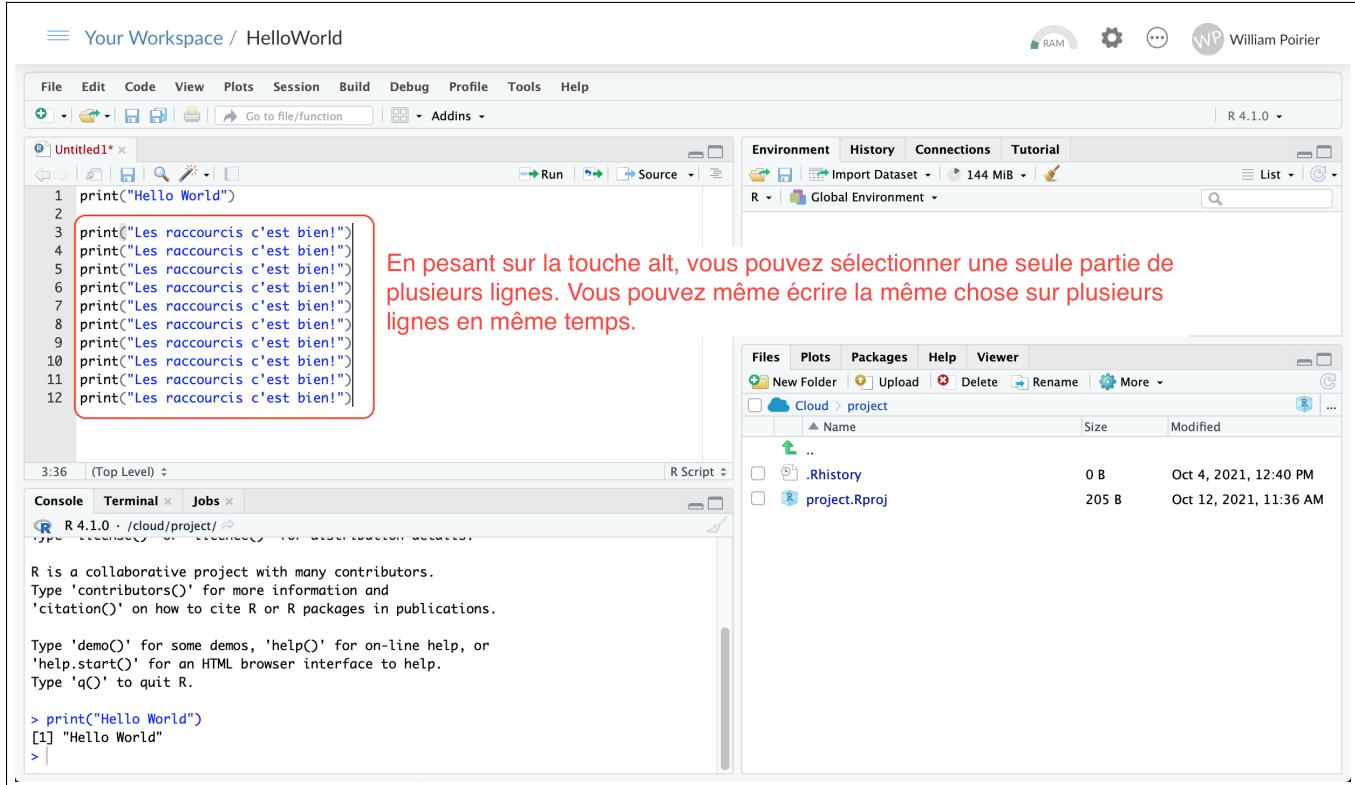


Figure 8: Sélection multiple de lignes

## 5 Intro à la programmation en R

Vous connaissez le logiciel **R** et vous avez de bonnes habitudes de codage. Nous pouvons désormais entrer dans le vif du sujet, la programmation en **R**. Que vous utilisez **RStudio** ou **RStudio Cloud**, les prochaines indications ne changent pas.

### 5.1 Ça va bien aller

Plusieurs étudiants sont anxieux face à la tâche d'apprendre un langage de programmation. C'est tout à fait normal. La programmation vient avec une aura cryptique, un jargon et des programmeurs qui ne sont pas toujours à même d'expliquer ce qu'ils font. Or, pour ce cours, nous considérerons **R** comme une calculatrice +, c'est-à-dire comme un outil permettant d'effectuer des opérations mathématiques plus ou moins complexes. Vous ne deviendrez pas des développeurs à la fin de ce cours et ce n'est pas l'objectif. Le but est plutôt de vous former pour que vous soyez en mesure d'effectuer des analyses de base, une compétence qui vous distinguera sur le marché du travail.

Ceci étant dit, vous allez avoir besoin d'aide à un moment, c'est certain. L'équipe d'enseignement est, bien entendu, disponible pour répondre à vos questions. Cependant, nous préférons que vous dévelopez votre autonomie. Pour ce faire, nous vous encourageons à consulter les ressources en ligne mentionnée à la section 2. D'autres ressources sont également offertes par R lui-même. En précédant toute fonction d'un ? dans la console 4:

```
# Dans la console, pour obtenir de l'aide pour la fonction sample()
> ?sample()
```

## 5.2 Opérateurs

Comme vous avez certainement déjà saisi, au lieu d'avoir une interface qui vous permet de cliquer sur les boutons et onglets, la plupart des actions en R se font à l'aide d'opérateurs. Ceux-ci sont simplement des symboles utilisés pour diriger le programme à faire des actions quelconques. C'est donc en manipulant ces opérateurs qu'il soit possible de calculer une moyenne, de faire un graphique, ou même de faire de l'apprentissage machine. Ainsi, les différents types d'opérateurs sont la base de la programmation en R. Nous pouvons sous-diviser les opérateurs et différents types. Les prochaines sections vous présenteront aux opérateurs de bases auxquels vous allez avoir affaire dans le cadre du cours.

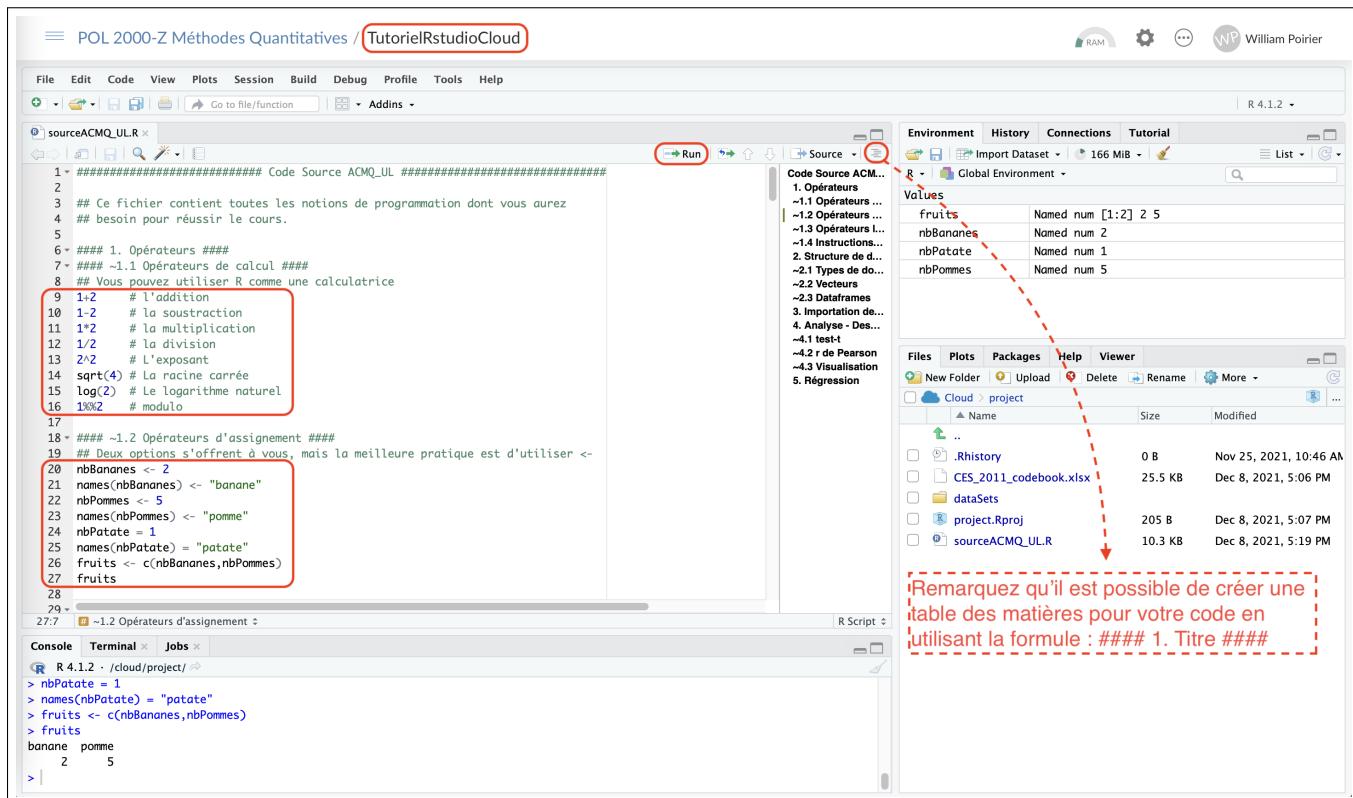


Figure 9: Opérateurs de calcul et d'assignement

### 5.2.1 Opérateurs de calcul

Les opérateurs de calcul sont les opérateurs mathématiques que vous connaissez (+, -, \*, /).<sup>5</sup> Ces opérateurs sont aussi visibles dans l'éditeur à la figure 9. Tout comme une calculatrice, vous pouvez utiliser ces opérateurs pour faire des calculs. Essayez :

<sup>4</sup> Lorsqu'une fonction n'est pas dans votre environnement de travail et que vous utilisez `?ggplot()`, R retournera une erreur du genre « `Error in .helpForCall(topicExpr, parent.frame()) : no methods for 'ggplot' and no documentation for it as a function` ». Lorsque c'est le cas, vous pouvez précéder la fonction d'un double ?. Ceci indiquera à R de chercher dans toute la documentation des serveurs de CRAN.

<sup>5</sup> Il existe aussi l'opérateur modulo (%%) qui permet de calculer le reste d'une division euclidienne. C'est pratique pour identifier si un nombre est un diviseur d'un autre nombre.

---

```
# Calcul pour obtenir la moyenne de cet échantillon fictif de 5 cas : 8, 12, 15, 19, 20
> (8 + 12 + 15 + 19 + 20) / 5
```

Mais cette façon de procéder est peu productive. Nous verrons les commandes pour les mesures de la centralité et de dispersion à la section 5.3.2.

### 5.2.2 Opérateurs d'assignement

Les opérateurs d'assignement, comme leur nom l'indique, permettent d'assigner des valeurs à des objets. Voici un exemple avec la moyenne que nous venons de calculer. Une fois exécuté, la console retournera la réponse « 14.8 » lorsque nous écrivons la moyenne dans l'éditeur et que l'on clique sur « Run ». Vous pouvez voir d'autres exemples de l'éditeur dans la figure 9.

```
# Assigner le calcul précédent à l'objet "moyenne"
> moyenne <- (8 + 12 + 15 + 19 + 20) / 5
```

Lorsque vous utilisez des opérateurs d'assignement, deux options s'offrent à vous. Vous pouvez utiliser le signe d'égalité (=) ou utiliser la flèche d'assignement (<-). Même si utiliser le signe d'égalité est plus intuitif pour la plupart d'entre vous, il est fortement recommandé d'utiliser la flèche d'assignement (<-). Pourquoi? Il s'agit d'un standard de bonne pratique. Souvent en programmation, il y a plusieurs façons d'arriver au même résultat. Les standards de bonne pratique nous permettent d'identifier les pratiques optimales sans avoir à comprendre pourquoi une pratique est meilleure qu'une autre dans toutes ses nuances. Pour ce cours, utilisez la flèche d'assignement (<-).

### 5.2.3 Opérateurs logiques

Les opérateurs logiques retournent l'une de deux valeurs, soit « vrai » ou « faux ». Ils sont donc utilisés pour comparer deux objets. Voici les principaux :

- A < B : A est plus petit que B
- A > B : A est plus grand que B
- A <= B : A est plus petit ou égal à B
- A >= B : A est plus grand ou égal à B
- A == B : A est égal à B
- A != B : A n'est pas égal à B
- & : Une condition **ET** une autre condition
- | : Une condition **OU** une autre condition
- A %in% B : A est contenu dans B

La figure 10 contient une série de tests logique (lignes 29 à 38) qui peuvent servir d'exemple. Cependant, nous vous suggérons fortement de tenter d'y répondre dans votre tête avant d'exécuter chaque ligne.

- `if (condition) {suite d'opération}`
- `else {suite d'opération}`
- `for (facteur d'itération) {suite d'opération}`

### 5.2.4 Instructions de contrôle

Jusqu'à présent, R ne se distingue pas beaucoup d'une calculatrice. Les instructions de contrôle sont ce qui rend R beaucoup plus puissant que votre calculatrice. En effet, elles permettent d'automatiser des procédures et des calculs. Trois instructions de contrôle principales sont à retenir :

Les instructions de contrôle sont souvent ce qu'il y a de plus difficile à apprendre lorsqu'on commence à programmer. *Leur maîtrise n'est donc pas essentielle pour la réussite de ce cours.* Il est cependant important que vous ayez connaissance de leur existence. Vous avez un exemple à la figure 10.

**POL 2000-Z Méthodes Quantitatives / TutorielRstudioCloud**

The screenshot shows the RStudio interface with the following elements:

- Code Editor:** Displays the script `sourceACMQ_UL.R` containing R code. A red box highlights the first few lines of code related to logical operators.
- Environment View:** Shows the global environment with objects like `fruits`, `fruitsVeggies`, `i`, `nbBananes`, `nbPatate`, `nbPommes`, `veggies`, and `visProduce`.
- Console:** Shows the output of the R code execution.
- Annotations:**
  - A red box surrounds the first few lines of code: `## Opérateurs logiques`, `1> 1 < 2`, `2> 2 < 3`, etc.
  - The text "Testez votre compréhension!" is placed above the code editor.
  - The text "De l'extra." is placed near the bottom right of the code editor area.

Figure 10: Opérateurs logiques

**POL 2000-Z Méthodes Quantitatives / TutorielRstudioCloud**

The screenshot shows the RStudio interface with the following elements:

- Code Editor:** Displays the script `sourceACMQ_UL.R` containing R code. A red box highlights the assignment of a factor variable.
- Environment View:** Shows the global environment with objects like `fruits`, `fruitsVeggies`, `i`, `myFruitsFactor`, `myFruitsNames`, `nbBananes`, `nbPatate`, and `nbPommes`.
- Console:** Shows the output of the R code execution.
- Annotations:**
  - A red box highlights the assignment line: `myFruitsName <- 1 #ERROR`.
  - The text "Remarquez l'importance de la classe des objets." is placed near the assignment line.
  - A red arrow points from the assignment line to the `myFruitsFactor` entry in the Environment view.
  - The text "Lorsque vous assignez un objet, il se retrouve dans votre environnement de travail." is placed at the bottom right of the screen.

Figure 11: Assigner un objet

```

81 ##### ~2.2 Vecteurs #####
82 myFruitsNames <- c("pomme", "banane", "orange", "tomate")
83
84 nbFruits <- c(5,2,7,1)
85 length(nbFruits)
86
87 names(nbFruits) <- myFruitsNames
88 nbFruits
89 class(nbFruits) #Est-ce numérique ou caractère?
90
91 nbFruits <- c(nbFruits, "concombre" = "trois")
92 nbFruits
93 class(nbFruits) #Et maintenant ?
94
95:1 ~2.2 Vecteurs +

```

Console Terminal Jobs

```

R 4.1.2 - /cloud/project/ ~
> myFruitsNames <- c("pomme", "banane", "orange", "tomate")
> myFruitsFactor <- factor(myFruitsNames)
> myFruitsFactor #Remarquez l'ordre alphabétique des "Levels"
[1] pomme banane orange tomate
Levels: banane orange pomme tomate
> myFruitsFactor <- factor(myFruitsNames, levels=c("orange", "pomme", "banane", "tomate"))
> myFruitsFactor #Remarquez comment le meilleur fruit est au début!
[1] pomme banane orange tomate
Levels: orange pomme banane tomate
> ##### ~2.2 Vecteurs #####
> myFruitsNames <- c("pomme", "banane", "orange", "tomate")
> nbFruits <- c(5,2,7,1)
> length(nbFruits)
[1] 4
> names(nbFruits) <- myFruitsNames
> nbFruits
pomme banane orange tomate
 5 2 7 1
> class(nbFruits) #Est-ce numérique ou caractère?
[1] "numeric"
> nbFruits <- c(nbFruits, "concombre" = "trois")
> nbFruits

```

Faire un `length()` sur un vecteur de longueur 4 est un peu inutile. Cependant, vous aurez bientôt à utiliser des bases de données contenant des milliers de lignes, ce qui vous empêchera de compter manuellement.

Figure 12: Le vecteur

## 5.3 Structure de données

Vous savez désormais comment effectuer des opérations de base sur des objets. Or, pour ce faire, il vous faut des objets à exploiter. Ces objets peuvent prendre plusieurs formes ayant chacune des caractéristiques et fonctionnalités différentes qui les rendent plus ou moins adaptés à différent type d'utilisation. Vous verrez ici les trois formats principaux qui seront utilisés dans le cours.

### 5.3.1 Types de données et constantes

Quatre types de données principales sont à maîtriser. Les données de classe *caractère* (ou «*character*») sont considérées comme du texte par **R**. *Elles ne peuvent donc pas être sujettes aux opérations mathématiques standards*. Par exemple, la figure 11 présente l'objet `veggiesName` qui contient la valeur textuelle `carotte`. Pour assigner une donnée textuelle, il faut l'encadrer de guillemets «`" "`». Si vous exécutez la ligne 62, vous obtiendrez une erreur puisque l'opération demandée ne peut pas s'appliquer à la classe de l'objet. Chose importante, même si l'il s'agit d'un chiffre, si vous l'encadrez de guillemets comme à la ligne 64, **R** considérera l'objet comme du texte et la même erreur se produira.

Pour faire des opérations mathématiques, il faut utiliser des valeurs numériques ou booléennes. Les valeurs numériques sont des chiffres. Ceci inclut les nombres entiers, rationnels ( $\pi$  ou  $e$ ), et décimaux. Les valeurs booléennes ont deux valeurs possibles qui correspondent aux réponses possibles des opérateurs logiques, soit `FALSE` ou `TRUE`. Bien qu'à première vue les valeurs booléennes ressemblent à des données textuelles, **R** les comprend comme 0 et 1 respectivement. C'est pourquoi si on additionne `tomatoFruit` et `potatoFruit`, on obtient 1. Pour ceux et celles d'entre vous un peu confus par ceci, rappelez-vous du processus de la codification des données et des guides de référence pour expliquer les bases de données.

Un format alternatif et particulier des données textuelles et numériques est le *facteur*. Conceptuellement, un facteur en **R** est un objet pouvant prendre un nombre limité de valeurs. En d'autres mots, il s'agit d'une variable catégorielle. À ces valeurs il est possible d'imposer un ordre (pensez ici aux variables ordinaires, intervalles et ratio). Par défaut, **R** ordonnera les valeurs numériques en ordre croissant et les valeurs textuelles en ordre alphabétique. En utilisant la fonction `factor()`, il est possible de créer un objet de classe facteur et de lui imposer un ordre avec

---

l'argument *levels*. Si les facteurs vous semblent compliqués pour rien, ils vous seront très utiles lorsque vous voudrez faire de la visualisation de données plus avancée. Il s'agit également d'une source d'erreur importante. Assurez-vous donc de bien connaître la classe des objets avec laquelle vous travaillez.

Il est possible de connaître la classe d'un objet en regardant ce qui se trouve à l'intérieur. Or, ce n'est pas toujours possible ou évident, soit à cause de la taille de l'objet (la quantité de choses qui s'y trouve), soit parce qu'on peut confondre un 1 caractère et un 1 numérique ou une "pomme" caractère et une "pomme" facteur. Pour être absolument certain, il est toujours préférable de demander à R en utilisant la fonction `class()`.

Les constantes sont des objets avec une seule donnée d'associée. À la figure 11, `veggiesName`, `nbVeggies`, et `tomatoFruit` sont des constantes, un objet avec un seul élément. Ces constantes n'ont cependant pas la même classe – soit, respectivement, caractère, numérique et booléen. Les constantes forment l'unité de base des structures de données possibles en R.

### 5.3.2 Vecteurs

Vous avez sans doute déjà entendu parler de vecteurs dans vos cours de mathématiques au secondaire. N'ayez crainte, il n'est pas question de la même chose ici. Un vecteur, pour R, c'est une série de données du même type. En d'autres mots, c'est une suite de constantes de la même classe mises bout à bout. Un vecteur R peut également être compris comme contenant les valeurs d'une variable – des notes d'examen, le positionnement idéologique, ou des noms de fruits. Sans vecteur, il n'est pas possible pour R d'effectuer des opérations aussi simples que le calcul d'une moyenne. Autant dire que R ne sert pas à grand chose sans vecteur. Heureusement, vous en avez déjà rencontré sans l'avoir remarqué. À la figure 12, `myFruitsNames` est un vecteur de classe « caractère » et de longueur 4. Pour déterminer combien d'éléments contient un vecteur (ou quelle est sa longueur, vous pouvez les compter vous-même ou demander à R de le faire pour vous avec la fonction `length()` comme c'est fait à la ligne 85 de la figure 12).

Pour créer un vecteur, il faut utiliser la fonction `c()` et l'assigner à un objet, comme à la ligne 82 de la figure 12. Le `c` de la fonction représente le verbe anglais «concatenate» qui signifie relier beaucoup de choses ensemble dans une chaîne ou une série. Vous pouvez également ajouter des noms aux éléments de votre vecteur. Ces noms n'auront aucune influence sur le traitement des valeurs associées. Par exemple, à la figure 12, j'ai un vecteur avec le nombre de chaque type de fruits dans le bon ordre (`nbFruits`). Si je désire leur associer un nom pour ne pas oublier ce que chaque chiffre représente, je peux utiliser la fonction `names()` comme à la ligne 87 de la figure 12. Cette ligne se lit comme suit : aux noms des éléments de `nbFruits`, assigne les valeurs qui se trouvent dans `myFruitsNames`. Vous pouvez également associer directement les noms de chaque élément à la création du vecteur comme à la ligne 91 de la figure 12.

À la ligne 91 de la figure 12, j'amende le vecteur `nbFruits` en y ajoutant un élément. Remarquez comment je réutilise l'objet `nbFruits` dans l'opération. La ligne 91 se lit : associe à l'objet `nbFruits` un vecteur des valeurs déjà contenues dans `nbFruits` et la valeur "trois" au nom "concombre". Or, n'ai-je pas dit plus haut qu'un vecteur se devait d'avoir le même type de valeurs? R est gentil, il vous laisse le faire sans retourner d'erreur. À la place, il prend une décision pour vous en assignant la classe caractère à **TOUS** les éléments du nouveau vecteur. C'est donc important d'être attentif à ce que fait R; il prend parfois de mauvaises décisions pour vous.

Lorsque le vecteur est de classe numérique, il est possible d'effectuer des opérations mathématiques plus ou moins complexes. Vous pouvez bien sûr additionner, soustraire, multiplier et diviser par la même valeur tous les éléments du vecteur. Or, R vous permet aussi d'effectuer les opérations statistiques de base sans appel à d'autres librairies. La moyenne, la médiane, l'écart-type, la variance et plusieurs autres opérations ont des fonctions associées. Vous pouvez les consulter à la figure 13. Je vous encourage à explorer le code et à expérimenter.

Le dernier élément important à aborder au sujet des vecteurs est la façon de sélectionner des éléments. Pour ce faire, on utilise les crochets "[ ]". Ceux-ci nous permettent d'indiquer la position dans le vecteur des éléments à sélectionner. La figure 13 montre, dans l'ordre à partir de la ligne 117, comment sélectionner un élément, une suite d'éléments et plusieurs éléments distincts. Il est également possible d'apposer un - devant le chiffre pour retirer l'élément.

## Opérations mathématiques sur des vecteurs

```

95 friendsAge <- c(18, 21, 23, 24, 23, 22, 23, 20, 31, 26, 28, 35, 23, 22, 21)
96
97 friendsAgePlusOne <- friendsAge + 1 #ajoute 1 à tous les éléments
98 friendsAgePlusOne - friendsAge #soustrait les éléments i de chaque vecteur
99
100 sum(friendsAge) #retourne la somme de tous les éléments du vecteur
101 mean(friendsAge) #retourne la moyenne de tous les éléments du vecteur
102 median(friendsAge) #retourne la médiane de tous les éléments du vecteur
103 range(friendsAge) #retourne le min et le max d'un vecteur
104 min(friendsAge) #retourne le min d'un vecteur
105 max(friendsAge) #retourne le max d'un vecteur
106 sd(friendsAge) #retourne l'écart type d'un vecteur
107 var(friendsAge) #retourne la variance d'un vecteur
108 quantile(friendsAge) #retourne les quartiles d'un vecteur
109
110 #retourne le min, le 1er quart., la médiane, la moyenne, le 3e quartile et le max
111 summary(friendsAge)
112
113 #retourne la fréquence de chaque élément unique du vecteur
114 table(friendsAge)
115
116
117 favoriteFruit <- myFruitsNames[3]
118 goodFruits <- myFruitsNames[1:3]
119 allButBananas <- myFruitsNames[c(1,3,4)]
120 noBananas <- myFruitsNames[-2] #est-ce qu'il y a une différence avec allButBananas?
121 #qu'est-ce qui arrive si on met un - devant le c ?
122 allButBananas <- myFruitsNames[-c(1,3,4)]
123
124 ~2.2 Vecteurs :
125
126
127 someMovies <- c(myFavMovie, "The Lord of the Rings: The Fellowship of the Ring",
128   "Star Wars: Episode V - The Empire Strikes Back",
129   "The Theory of Everything", "Arrival", "Les choristes",
130   "Amadeus", "Astérix & Obélix: Mission Cléopâtre",
131   "De père en flic")
132 DataMovies <- data.frame>Title=someMovies)
133 for (i in 1:length(someMovies)){
134   #install.packages("imdbapi") si nécessaire
135   Info <- imdbapi::find_by_title(someMovies[i], include_tomatoes = FALSE,
136     api_key = "333f9bc4")[1,]
137   if (i == 1) {
138     DataMovies <- Info
139   } else{
140     DataMovies <- rbind(DataMovies,Info)
141   }
142   print(DataMovies>Title[i])
143 }
144
145 #Je peux ajouter mon propre score à chaque film!
146 DataMovies$persScore <- c(9.5,9,9,8,7.8,5,8,6,5)
147 # C'est quoi le score IMDB de De père en flic?
148 DataMovies$imdbRating<-DataMovies>Title=="De père en flic"]
149 #C'est quoi le meilleur film selon IMDB?
150 DataMovies>Title[DataMovies$imdbRating==max(DataMovies$imdbRating)]
151 #Et le pire?
152 DataMovies>Title[DataMovies$imdbRating==min(DataMovies$imdbRating)]
153 #Remarquez le nombre d'éléments à la sortie!
154 #C'est quoi le score moyen de notre sélection?
155 mean(DataMovies$imdbRating)
156
157
158 ~2.3 Dataframes :
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1036
1037
1038
1038
1039
1040
1041
1042
1043
1044
1045
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1056
1057
1058
1058
1059
1060
1061
1062
1063
1064
1065
1065
1066
1067
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1075
1076
1077
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1094
1095
1096
1096
1097
1098
1098
1099
1099
1100
1101
1101
1102
1103
1103
1104
1105
1105
1106
1107
1107
1108
1109
1109
1110
1111
1111
1112
1113
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1121
1122
1123
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1131
1132
1133
1133
1134
1135
1135
1136
1137
1137
1138
1139
1139
1140
1141
1141
1142
1143
1143
1144
1145
1145
1146
1147
1147
1148
1149
1149
1150
1151
1151
1152
1153
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1161
1162
1163
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1171
1172
1173
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1181
1182
1183
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1191
1192
1193
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1201
1202
1203
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1211
1212
1213
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1831
1832
1833
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1841
1842
1843
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1851
1852
1853
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1861
1862
1863
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1871
1872
1873
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1881
1882
1883
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1891
1892
1893
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
190
```

### 5.3.3 Data frames

Un *data frame* (il n'y a pas vraiment de traduction satisfaisante, mais le terme base de données est aussi communément utilisé) contient des données organisées en rangées et en colonnes, comme un tableau Excel. Chaque colonne peut avoir une classe différente, mais les données au sein d'une même colonne doivent être de la même classe. Généralement, les colonnes seront associées aux variables, tandis que chaque rangée correspondra à une observation.

Les *data frames* sont composés de vecteurs de la même façon que les vecteurs sont composés de constantes. On peut donc avoir un film préféré (une constante), une liste de films que nous avons vus (un vecteur) ainsi qu'un fichier Excel regroupant ces films, leur année de publication, leur genre, leur score IMDB, etc.<sup>6</sup> Bien qu'il soit possible de créer des *data frames* par vous-même, il est rare qu'un chercheur construise manuellement sa base de données directement dans **R**. Le plus souvent, une base de données existante est téléchargée ou créée dans d'autres logiciels comme un fichier Excel.

Comme les constantes et les vecteurs sont à la base du fonctionnement de **R**, le *data frame* est l'objet le plus important pour le travail des analystes de données. En effet, les données que vous analyserez ne viendront pas sous forme de vecteur, ce serait un cauchemar d'organisation. Imaginez avoir à calculer la moyenne d'un.e étudiant.e pour un cours avec 20 évaluations. Créer un vecteur avec les notes de tous les étudiant.es pour chaque évaluation serait beaucoup trop inefficace. La solution serait de créer un fichier Excel avec une colonne pour les noms des étudiant.es et une colonne par évaluation. Ainsi, au lieu d'avoir 20 vecteurs nommés (rappelez-vous de la fonction `names()` à la figure 12), vous auriez 1 fichier avec 21 colonnes. Ceci vous permet non seulement de calculer la moyenne par étudiant, mais également de croiser vos données. Par exemple, vous pourriez ajouter l'âge des étudiant.es ou leur nombre de sessions universitaires complétées afin d'évaluer si l'une de ces variables semble avoir un effet sur la réussite du cours. C'est ici que la puissance de **R** peu vraiment être exploitée, et c'est pourquoi il est très important de bien comprendre comment manipuler le *data frame*. Il en va de votre capacité à analyser les données. Les prochains paragraphes pourront paraître plus obtus, mais il faut faire un effort pour bien les comprendre.

L'aspect important à reconnaître, c'est qu'un *data frame* a deux dimensions au lieu d'une (les rangées et les colonnes). Pour se déplacer dans un *data frame*, il faut donc ajouter un élément à nos crochets (`[]`) habituel. Dans **R**, le premier élément retrouvé dans les crochets correspond au numéro de la rangé, le second au numéro de la colonne. Essentiellement les `r` et les `c` de la figure 15. Bien que **R** soit un outil puissant, il nous demande souvent de faire des abstractions, comme d'indiquer exactement la rangée et la colonne qui nous intéresse. Une manière de visualiser ceci est d'imaginer un fichier Excel ou un jeu d'échecs qui vous indique les colonnes avec des lettres et les rangées avec des chiffres. Ainsi, si je m'intéresse au deuxième élément de la troisième colonne d'un *data frame* nommé `Data`, je peux l'extraire avec `Data[2,3]`. Une erreur que font plusieurs codeurs est d'inverser rangées et colonnes. Il faut se souvenir: rangé, colonne, ou `Data[r,c]`!

Il est cependant rare que nous nous intéressions à des colonnes ou à des cellules particulières d'un *data frame* en termes de numéros de rangée et de colonne. Le plus souvent, nous voulons faire une opération sur une colonne comme une moyenne afin d'avoir des statistiques descriptives. À la figure 14, nous nous intéressons à la moyenne des scores IMDB d'une sélection de films. Pour ce faire, nous utilisons le \$ pour dire à R que nous nous intéressons à la colonne `imdbRating` de `DataMovies` (`Data$nomDeLaColonne`).

De la même façon, il est également possible de combiner le `$` avec les crochets (`[]`) pour sélectionner des éléments d'un *data frame* sous certaines conditions. Ceci est un outil puissant pour explorer vos données. Dans le cas du

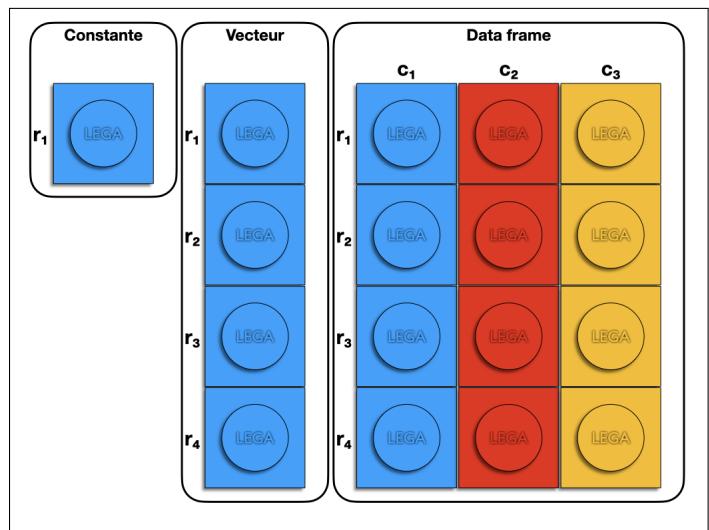


Figure 15: Représentation des types d'objets **R**

<sup>6</sup> Vous remarquerez que je fais appel au package `imdbapi` à la figure 14. Il s'agit d'une librairie contenant des fonctions qui me permettent de communiquer avec l'API d'IMDB et de lui faire des requêtes. Dans le code, je fournis une clé pour y accéder, mais pour celles et ceux qui désireraient leur propre clé c'est [ici](http://www.omdbapi.com/apikey.aspx) → <http://www.omdbapi.com/apikey.aspx>. Pour de la documentation sur le package c'est [là](https://cran.r-project.org/web/packages/imdbapi/imdbapi.pdf) → <https://cran.r-project.org/web/packages/imdbapi/imdbapi.pdf>.

score IMDB de « De père en flic », la ligne se lit : dans la colonne `imdbRating` du *data frame* `DataMovies`, sélectionne les éléments dont la valeur à la même rangée dans la colonne `Title` du *data frame* `DataMovies` est égale à "De père en flic". Les possibilités de conditionnalité sont donc infinies. Dans l'exemple on s'intéresse aux scores maximum et minimum, mais il est possible de combiner des tests booléens (vrai ou faux, `TRUE/FALSE`) avec les arguments conditionnels "`|`" (OR) et "`&`" (AND).

Enfin, les mêmes opérations mathématiques décrites à la figure 13 pour les vecteurs s'appliquent aux colonnes des *data frames*. C'est normal, rappelez-vous, une colonne d'un *data frame* c'est comme un vecteur. Il n'y a qu'à choisir la colonne avec le `$` et on peut appliquer les fonctions de moyenne, d'écart-type, de variance, etc.

The screenshot shows the RStudio Cloud interface. In the top navigation bar, it says "POL 2000-Z Méthodes Quantitatives / TutorielRstudioCloud". The main area is a code editor with the following R script:

```

157
158
159
160 - ##### 3. Importation des données #####
161 Data <- read.csv("/Users/williampoirier/Dropbox/Travail/Ulaval/Contrats/acmq_ul/dataIMDB/movieData.csv")
162 # Je suis sur Rstudio Cloud, donc ça ne fonctionne pas.
163 Data <- read.csv("./cloud/project/dataSets/movieData.csv") // Line 163 highlighted by a red box
164 # Beaucoup mieux!
165
166
164:17 □ 3. Importation des données □

Console Terminal Jobs
R 4.1.2 : /cloud/project/ □
> DataMovies$imdbRating[DataMovies$title=="De père en flic"]
numeric(0)
#C'est quoi le meilleur film selon IMDB?
> DataMovies$title[DataMovies$imdbRating==max(DataMovies$imdbRating)]
[1] "The Lord of the Rings: The Fellowship of the Ring"
> #Et le pire?
> DataMovies$title[DataMovies$imdbRating==min(DataMovies$imdbRating)]
[1] "De père en flic 2"
> #Renarez le nombre d'éléments à la sortie!
> #C'est quoi le score moyen de notre sélection?
> mean(DataMovies$imdbRating)
[1] 7.855556
>
>
> ##### 3. Importation des données #####
> Data <- read.csv("/Users/williampoirier/Dropbox/Travail/Ulaval/Contrats/acmq_ul/dataIMDB/movieData.csv") // Line 164 highlighted by a red box
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file '/Users/williampoirier/Dropbox/Travail/Ulaval/Contrats/acmq_ul/dataIMDB/movieData.csv': No such file or directory
> # Je suis sur Rstudio Cloud, donc ça ne fonctionne pas.
> Data <- read.csv("./cloud/project/dataSets/movieData.csv") // Line 165 highlighted by a red box
> # Beaucoup mieux!
>

```

A red arrow points from the text "L'arborescence provient de l'environnement Rstudio Cloud lui-même!" to the "Cloud" section of the file browser. Another red arrow points from the error message "R vous dit assez clairement lorsqu'il y a une erreur dans la compilation de votre code. N'hésitez pas à mettre directement le code d'erreur dans Google pour vous aider à identifier le problème." to the error line in the code editor.

Figure 16: Importer des données à un environnement de travail

## 6 Importation de données

### 6.1 Importation de données

Avant de débuter l'analyse, il faut des données à analyser. En science politique, les bases de données auxquelles vous ferez face seront typiquement rectangulaires. Elles auront donc la même structure qu'un *data frame* tel que compris par **R**: un cas par ligne, une variable par colonne. Si vous êtes familier avec Excel, vous reconnaîtrez certainement cette structure.

**R** étant un langage *open source*, la communauté de développement donne accès à plusieurs librairies (*packages*) dédiées à la lecture de différents formats de données. Dans le cadre du cours, vous n'aurez uniquement qu'à traiter des données enregistrées en format `.csv`, valeurs séparées par des virgules (*comma-separated values*). C'est ce qui se cache derrière les tableaux Excel. Or, les formats `.sav`, `.spss`, `.rds`, et `.tsv` sont également populaires. Si vous rencontrez un jour l'un de ces formats, il est certain qu'une fonction pour les ouvrir existe déjà. À ce moment, **Google** sera votre meilleur ami! Une fois la bonne fonction identifiée, la procédure est assez simple si l'on comprend bien le concept d'arborescence. Je vous invite donc à réviser la section ?? avant de continuer.

Pour ouvrir un fichier `.csv`, il faut exécuter la fonction `read.csv()` et assigner le résultat à un objet (typiquement appelé *Data*) comme le montre la figure 16. Il est important de constater que le seul argument requis est l'arborescence menant au fichier `.csv` que vous voulez ouvrir. Dans le cas de la figure 16 à la ligne 161, il s'agit

---

d'une arborescence d'ordinateur Mac typique. Pour un PC, un `C:` aurait été ajouté devant `/Users`. La procédure est similaire pour un projet **RStudio Cloud**. Or, il faut d'abord télécharger le fichier dans le projet **RStudio Cloud**.

**RStudio Cloud** est une plateforme web. Ce faisant, elle n'a pas directement accès à votre ordinateur comme c'est le cas si vous utilisez **RStudio**. Référez-vous à la figure 3 pour savoir comment créer un projet. Une fois le projet créé, vous pouvez ajouter un fichier de données en appuyant sur le bouton *upload*. Vous n'aurez toutefois pas à le faire normalement. En effet, les projets **RStudio Cloud** nous permettent de créer un environnement identique pour tous les étudiants et d'y ajouter les fichiers nécessaires. Ainsi, si vous créez votre propre projet à partir de l'espace de travail *POL 2000-Z Méthodes Quantitatives*, vous aurez accès à tout ce dont vous aurez besoin.

## 7 Analyse - Description

L'analyse descriptive des données est l'outil de base du chercheur voulant explorer une base de données. Cette section présentera les principaux tests, visualisations et mesures d'analyse descriptive.

### 7.1 Univariée

Une analyse univariée est, comme son nom l'indique, une analyse s'effectuant sur UNE SEULE variable. Vous avez déjà rencontré les principales mesures lui étant associé à la figure 13. En effet, ce qui nous intéresse dans ce genre d'analyse est la distribution d'une variable. Est-elle distribuée normalement ou est-elle asymétrique? Quel est son mode? Sa variance? Les **mesures de dispersion** permettent d'identifier le positionnement des observations sur la distribution et de répondre à ces questions. Ainsi, pour obtenir la moyenne d'une variable, il faut utiliser la même fonction qu'à la figure 13 en indiquant cette fois la bonne colonne du *data frame* où se trouve la variable d'intérêt (`mean(Data$nomDeLaVariable)`). Rappelez-vous: les colonnes d'un *data frame* ne sont que des vecteurs déguisés!

### 7.2 Bivariée

Vous vous en doutez certainement, si une analyse univariée se concentre sur une seule variable, une analyse bivariée en compare deux. Étonnant, n'est-ce pas?! Plusieurs tests existent pour comparer deux variables, nous vous en présenterons deux: le test-t de Student et le r de Pearson.

Le test-t de Student permet d'effectuer un test d'hypothèse statistique. Il est souvent utilisé pour déterminer si les distributions (ou les moyennes) de deux variables sont significativement différentes. Par exemple, si on s'intéresse à la différence de revenu collecté pour différents types de films, nous pouvons tester les moyennes grâce à un test-t. La figure 17 montre comment utiliser la fonction `t.test()`. Celle-ci prend comme arguments principaux deux variables. Dans notre cas, un vecteur du revenu des films romantiques et un vecteur du revenu des films d'action. Remarquez comment sont spécifiés les vecteurs d'intérêt. Si nous utilisions uniquement `Data$revenue` en x et en y, le test-t ne remarquerait aucune différence puisque c'est la même variable. Il est également possible de spécifier l'intervalle de confiance désiré. N'oubliez pas, si vous souhaitez plus d'informations sur l'utilisation d'une fonction, inscrivez et exécutez `?maFonction()` dans la console. Une fenêtre d'information s'ouvrira.

Le r de Pearson est un test de corrélation simple et pratique qui vous permettra de tester la relation entre deux variables. Distinguez bien les deux concepts. Le test-t permet de tester si deux distributions sont différentes. La corrélation permet, elle, de tester s'il y a une relation entre deux distributions. Attention: une corrélation entre deux variables ne veut pas dire qu'il y a causation! La fonction **R** pour ce test est utilisée de la même manière que pour le test-t. Il suffit d'identifier les arguments x et y (les deux variables) de la fonction `cor.test()` comme le montre la figure 18 et le tour est joué.

The screenshot shows the RStudio interface with the following details:

- Top Bar:** POL 2000-Z Méthodes Quantitatives / TutorielRstudioCloud
- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help
- Addins:** Go to file/function, Addins
- Code Editor:** A script named "sourceACMQ\_UL.R" containing R code for a t-test. A red box highlights the following lines:

```
166 - ##### 4. Analyse - Description #####
167 - ##### -4.1 test-t #####
168 #Est-ce que les films romantiques ont systématiquement plus de revenus que les films d'action?
169 names(Data)
170
171 cashRomance <- Data$revenue[Data$Romance==1]
172 cashAction <- Data$revenue[Data$Action==1]
173 t.test(x=cashRomance,
174         y=cashAction,
175         conf.level=0.95)
176 #ou
177 t.test(x=Data$revenue[Data$Romance==1],
178         y=Data$revenue[Data$Action==1],
179         conf.level=0.95)
180 #Dans les deux cas t=-0.23187 et p=0.8167
181 #Pour qu'il y ait une différence, on veut un grand t!
182 # p < 0.001 => ***
183 # p < 0.01 => **
184 # p < 0.05 => *
185
186 # ou
187 # -4.1 test-t :
```
- Text Output:** The R console shows the results of the t-test:

```
R 4.1.2 - /cloud/project/
+ y=Data$revenue[Data$Action==1],
+ conf.level=0.95)

Welch Two Sample t-test

data: Data$revenue[Data$Romance == 1] and Data$revenue[Data$Action == 1]
t = -0.23187, df = 444.04, p-value = 0.8167
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-14446452 11397336
sample estimates:
mean of x mean of y
40933136 42457694
```
- Environment Tab:** Shows the global environment with objects like Data, DataMovies, Info, Values, and datasets like cleanCES2011.csv and movieData.csv.

Figure 17: Test t de Student

The screenshot shows the RStudio interface with the following components:

- Script Editor (top-left):** Displays R code for calculating Pearson correlations between revenue and family/fantasy genres. Lines 214 and 215 are highlighted with red boxes.
- Console (bottom-left):** Shows the execution of the code. The output for the first correlation shows a p-value of `p-value < 2.2e-16`, which is highlighted with a red box. The output for the second correlation shows a p-value of `2.752e-07`, also highlighted with a red box.
- Environment (right):** Shows the global environment with objects like Data, DataMovies, and movieData.
- Files (bottom-right):** Shows files in the project's dataSets folder, including cleanCES2011.csv and movieData.csv.

A red box highlights the p-value output in the console for both correlations.

Figure 18: Test de corrélation (r de Pearson)

**La fonction qui assigne le modèle de régression à l'objet «modelA».**

```
sourceACMQ_UL.R
290 ~ ### 5. Régression ###
291 #Est ce qu'on fait de meilleurs films qu'avant?
292 modelA <- lm(averageRating ~ year,
293   data=Data)
294 summary(modelA)
295
296
297:16 5. Régression
```

**La fonction «summary()» permet d'extraire le modèle de régression. Il faut s'assurer de repérer :**

- 1) les coefficients de régression, et
- 2) les valeurs p.

Une autre valeur utile est le «R-squared».

**Lorsque les valeurs sont trop petites ou trop grandes, R utilise la notation scientifique. Le petit e correspond à 10 et le chiffre qui suit à son exposant. Ainsi,  $2e-16$  est égale à  $2 \times 10^{-16}$  ou  $0.0000000000000002$**

Figure 19: Régression linéaire simple

Même principe que pour la régression linéaire simple, mais on ajoute des variables à l'aide de «+».

R vous aide à repérer les effets significatifs en ajoutant des étoiles aux valeurs p. Il vous donne même le guide pour les interpréter!

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-2.891e+01	2.629e+00	-10.996	<2e-16 ***
year	1.791e-02	1.315e-03	13.618	<2e-16 ***
runtimeMinutes	-1.938e-03	9.999e-04	-1.938	0.0527 .
budget	1.166e-09	8.018e-10	1.454	0.1461
revenue	-5.740e-11	1.976e-10	-0.291	0.7714
Animation	-6.177e-02	4.143e-02	-1.491	0.1360
Comedy	2.901e-03	4.849e-03	0.060	0.9523
Drama	-9.143e-06	6.977e-02	-0.1310	0.1901
Fantasy	8.255e-05	6.710e-02	0.001	0.9990
Horror	5.219e-02	8.353e-02	0.625	0.5321
Action	1.568e-01	1.080e-01	1.452	0.1467
History	7.130e-02	6.938e-02	1.028	0.3042
Sci-Fi	2.244e-01	1.455e-01	1.542	0.1230

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.298 on 4963 degrees of freedom  
(2379 observations deleted due to missingness)

Multiple R-squared: 0.04669 Adjusted R-squared: 0.04438  
F-statistic: 20.25 on 12 and 4963 DF, p-value: < 2.2e-16

Figure 20: Régression linéaire multiple

## 8 Analyse - Régression linéaire simple et multiple

La régression est sans doute l'outil le plus utile de l'arsenal de l'analyse quantitative. Elle est à la base de plusieurs modèles statistiques plus avancés allant même jusqu'à l'apprentissage machine. Le principe de la régression devrait vous être plus familier à la suite du cours. Cette section se concentrera donc sur la façon d'en faire avec **R**.

La fonction `lm()` permet de calculer un modèle de régression comme le montre la figure 19. Le premier argument est toujours la variable dépendante (ou variable prédictive) suivie d'un tilda (~) et de la ou des variables indépendantes. Celles-ci s'ajoutent à l'aide du `+`. Une fois toutes les variables du modèle spécifiées, il faut ajouter une virgule suivie de l'argument `data=` et du bon *data frame*. Attention, il est important d'utiliser le même nom de variable que les noms associés aux colonnes du *data frame* en question. À la figure 19, on assigne l'extrant de la fonction à un objet appelé `modelA`. De cette façon, il est possible d'utiliser la fonction `summary(modelA)` pour obtenir plus d'informations sur le modèle. La figure 20 présente un modèle de régression multiple. La différence? Il y a plus de variables après les `+`. Voilà! Vous savez coder un modèle de régression linéaire en **R** maintenant. C'est aussi simple que ça!

## 9 Visualisation

Bien souvent, il est difficile de se souvenir de l'interprétation des différents tests et les chiffres sont rarement le meilleur moyen de transmettre l'information à un plus large public. Grâce à **R** et la fonction `ggplot()`, il est facile de présenter visuellement les résultats de nos analyses.<sup>7</sup> Nous ne ferons ici qu'une introduction aux principales fonctionnalités de `ggplot2`. Pour une revue exhaustive des fonctionnalités, consultez ce [lien](#).<sup>8</sup>

`ggplot` s'attend à recevoir au moins 3 éléments présentés aux lignes 225–26 de la figure 21 : 1) un *data frame*, 2) un *aesthetic*, et 3) un `geom`. Le *data frame* contient les informations qui composeront le graphique. Il peut s'agir de la base de données originale, d'un sous-échantillon ou même des résultats d'une analyse quelconque (une suite de moyennes par année, par exemple). L'*aesthetic* (ou `aes`) contient les arguments permettant d'identifier le rôle des variables d'intérêt du *data frame*. C'est là où sont spécifiés l'axe des X, l'axe des Y, et la façon d'associer les groupes par couleur ou par texture. Par exemple, aux lignes 225–26 de la figure 21, il est seulement spécifié que la variable `runtimeMinutes` correspond à l'axe des X. Pourquoi n'est-il pas spécifié un axe des Y? C'est là qu'entrent en jeu les `geom`. Ce sont essentiellement les fonctions qui vous permettent de spécifier ce qui doit être dessiné. Voici une liste des principaux `geom`:

- `geom_histogram()` → un histogramme
- `geom_bar()` → un graphique à barres
- `geom_col()` → un graphique à barres<sup>9</sup>
- `geom_point()` → un nuage de points
- `geom_smooth()` → une courbe de régression

Ce qu'il faut bien comprendre, c'est que les arguments de l'*aesthetic* doivent bien s'agencer avec le `geom` voulu. Par exemple, un histogramme n'a pas besoin d'une position en Y puisque le `geom` calcule la fréquence lui-même. Or, pour un nuage de points, il faut spécifier à quelle variable correspond l'axe des X et l'axe des Y. Ainsi, le *data frame*, l'*aesthetic* et le `geom` doivent être en harmonie pour que la fonction retourne le graphique désiré. Pour vous faciliter la vie, je vous propose de toujours répondre à ces 3 questions avant de tenter la production d'un graphique :

1. Comment vous imaginez-vous le graphique ?
2. Quelles sont les variables d'intérêt ?
3. Est-ce que mon *data frame* a le bon format ?

De cette façon, vous vous assurerez de ne pas faire d'erreurs de logique lors de l'écriture de la fonction. Bien entendu, ces trois questions et les `geom` ne vous permettent d'accéder qu'à une partie de l'ensemble des possibilités et des fonctionnalités qu'offre `ggplot`. Que ce soit les couleurs, la taille des points ou des barres, l'espace des

<sup>7</sup> En fait, `ggplot()` est une fonction issue de la librairie `ggplot2`, elle-même issue de la meta-librairie `tidyverse`.

<sup>8</sup> Le voici si vous lisez sur papier <https://ggplot2.tidyverse.org/reference/>

<sup>9</sup> La différence entre `geom_bar()` et `geom_col()` est que `geom_bar()` prend une fonction en Y (`..count..` par exemple) alors que `geom_col()` prend une variable.

barres, le nom des axes, leur gradation, l'ajout d'un titre, d'un sous-titre, d'une note en bas de page, d'annoter le graphique, d'ajuster la légende, la taille et l'angle du texte, de modifier la grille, la couleur de fond, etc. Tout ceci s'apprend sur le tas à l'aide de Google. Si vous avez une question sur une idée de graphique, forte chance que quelqu'un se soit déjà posé cette question. Demandez à Google!

Le dernier concept à bien comprendre pour utiliser `ggplot()` à son plein potentiel est le concept de couche. C'est ce qui rend `ggplot()` aussi versatile. Vous avez sans doute remarqué les petits + à la fin de chaque ligne à la figure 21. Ceux-ci se traduisent pour R comme «ajoute ce qui suit au-dessus de ce qui se trouve avant». Il est donc possible de superposer les `geom` afin de présenter plus d'informations aux lecteurs ou de mettre en relief certaines informations. Par exemple, à la figure 22, un `geom_smooth()` est ajouté par-dessus un `geom_point()` et un `geom_line()` afin de présenter l'augmentation du nombre de films produits au fil des années. La figure 24 quant à elle présente le code du graphique de la figure 23. Vous y trouverez un code de visualisation plus avancé qui cartographie les fonctions principales de modification de l'aspect visuel du graphique. Il est fortement recommandé d'exécuter le code ligne par ligne afin d'identifier ce que fait chaque partie de la fonction. Comme pour toute chose, c'est en pratiquant que vous apprendrez!

La fonction `ggsave()` permet d'enregistrer sous différents formats (pdf, png, jpeg, gif, mp4, etc.) les graphiques produits à l'aide de `ggplot()`. Pour ce faire, il faut spécifier l'arborescence en la terminant par le nom du fichier suivi de l'extension (.png) désirée. Le ratio largeur:hauteur (`width:height`) peut être spécifié ou non. Dans le cas de la figure 23, le ratio spécifié permet de respecter le standard des images Twitter de 16:9.

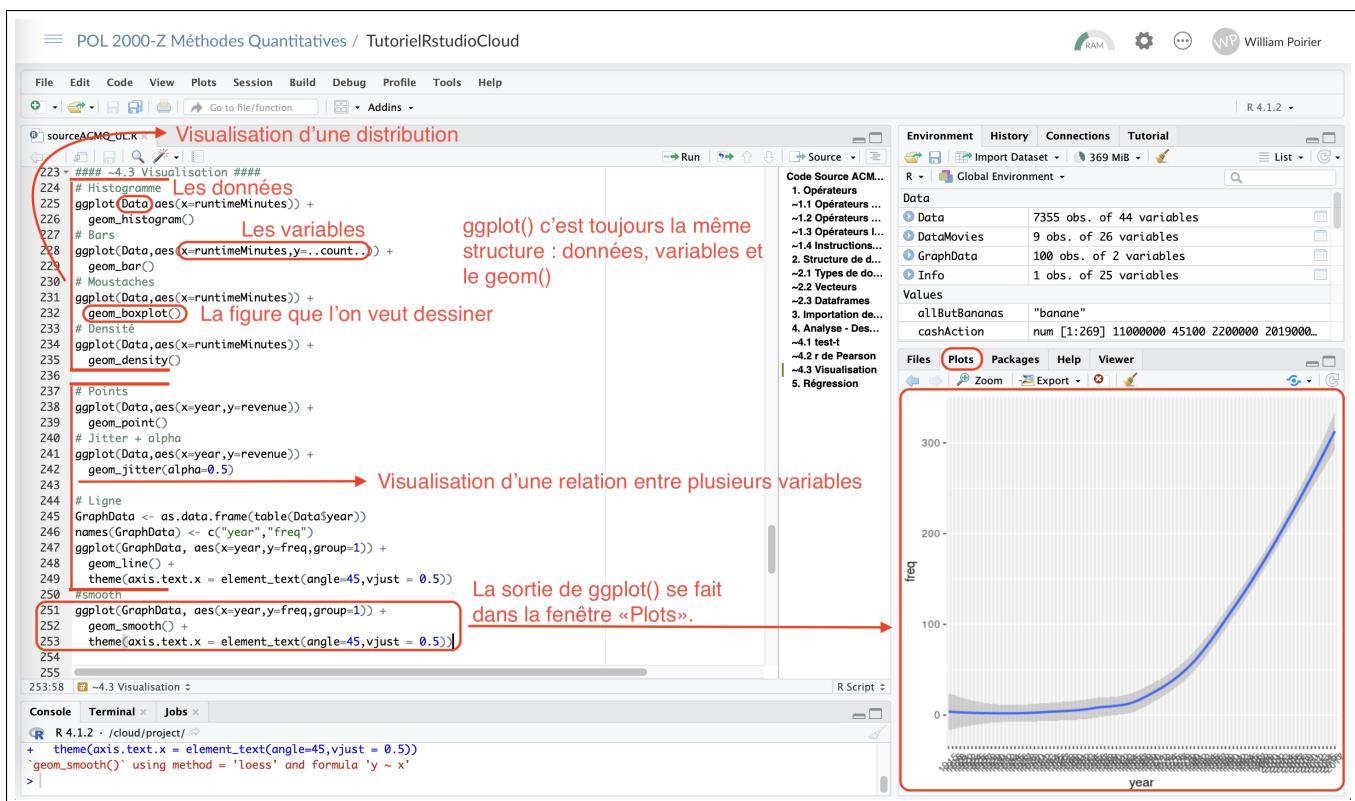


Figure 21: Quelques exemples d'utilisation de `ggplot2`

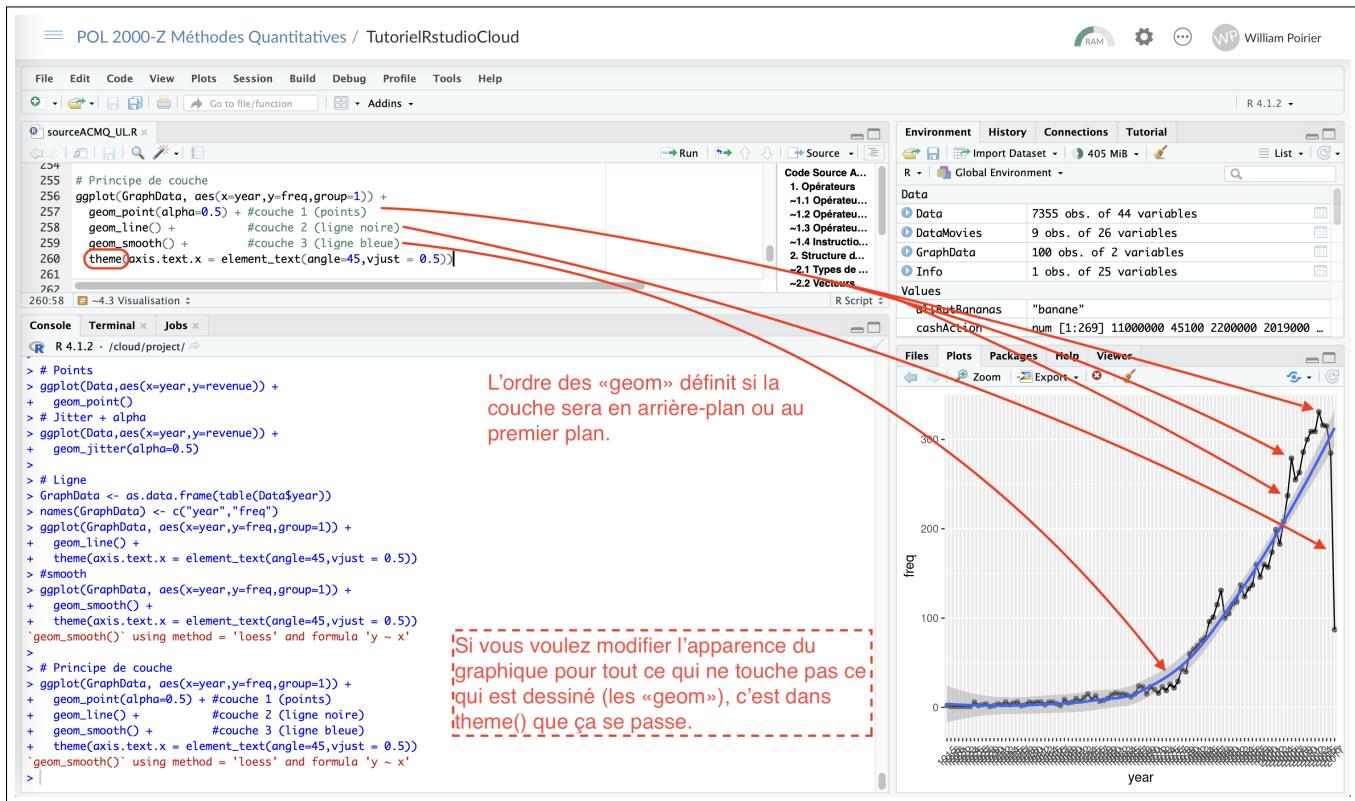


Figure 22: Illustration du principe de couche

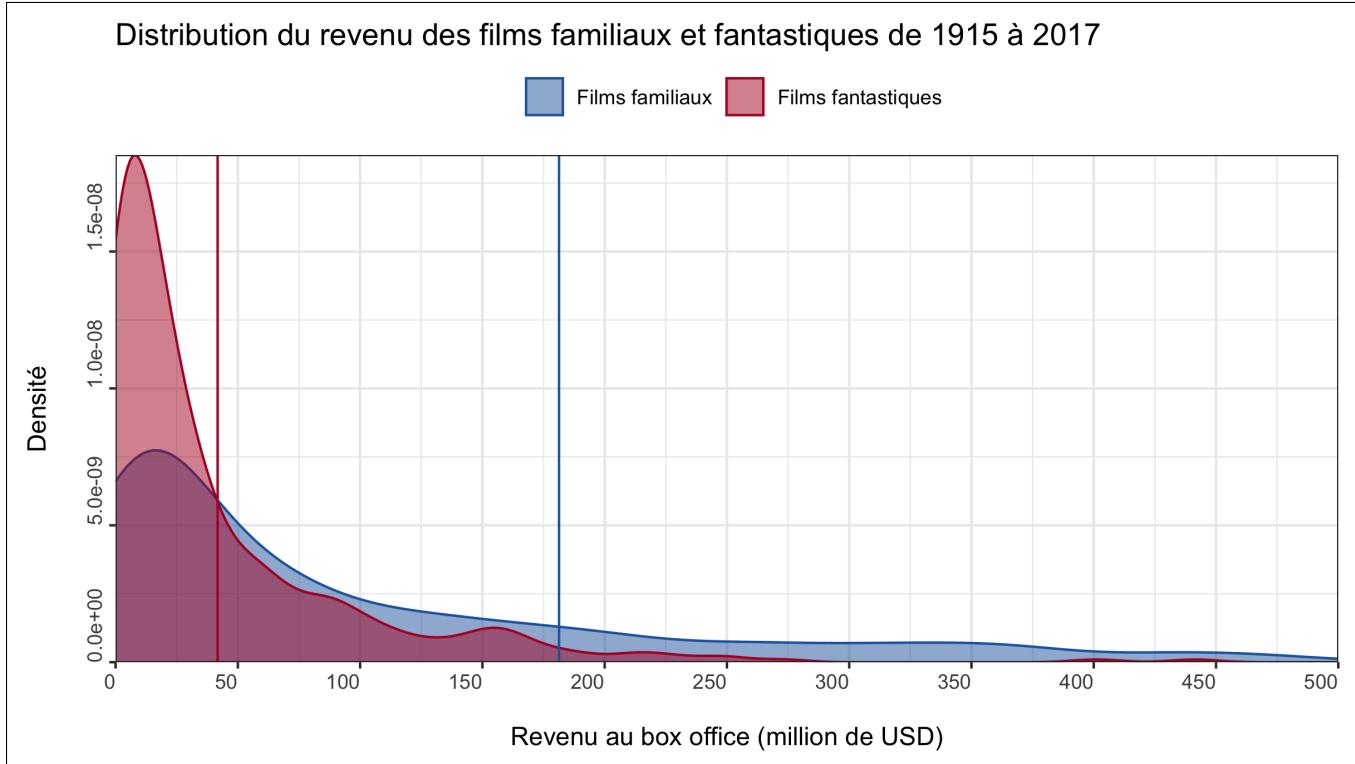


Figure 23: Exemple d'un graphique plus avancé

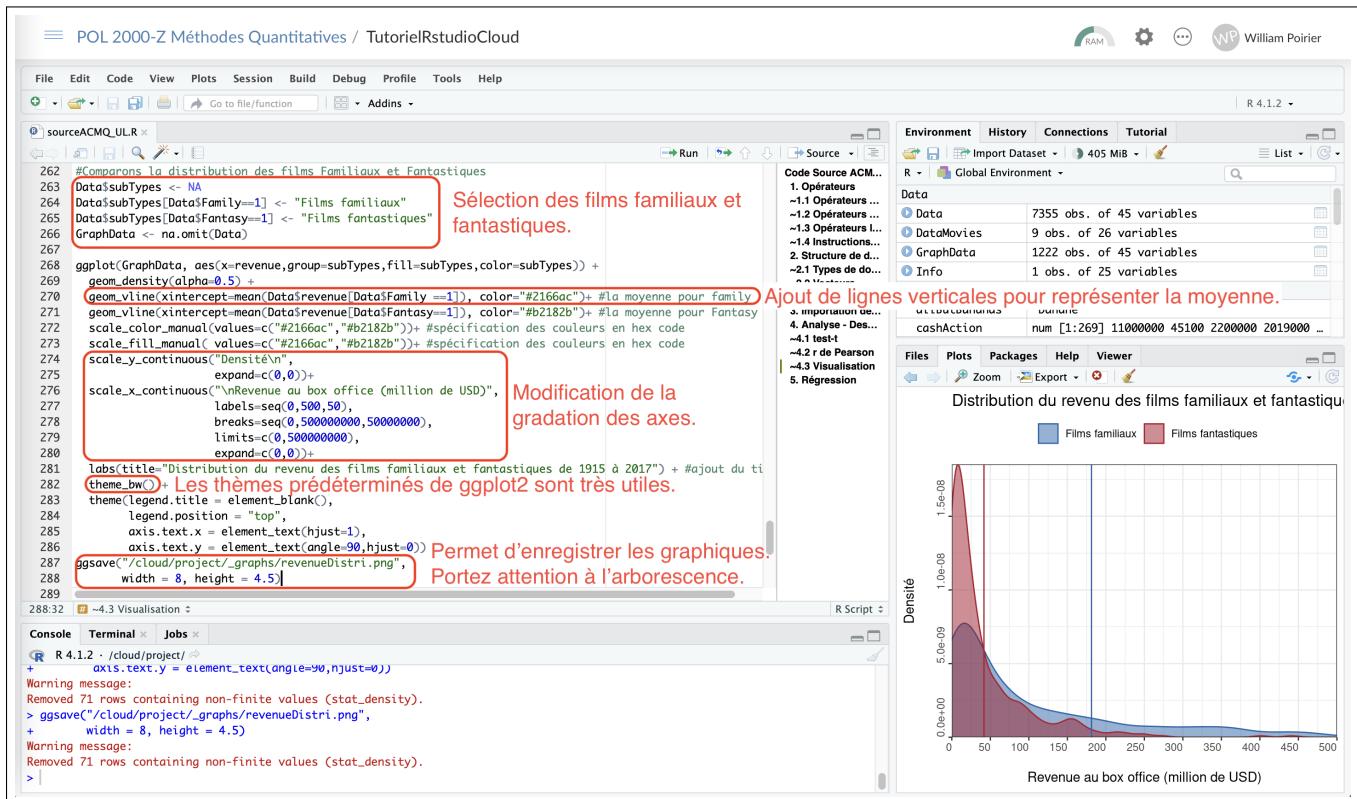


Figure 24: Code d'un graphique plus avancé