Petite introduction à LaTeX en Sciences Sociales

Julio Ricardo Davalos

Septembre 2022

Introduction

Ce document a pour but d'introduire à l'utilisation du langage LaTeX à un public venant des SHS. En effet, ce langage est assez commun dans les sciences expérimentales mais reste inconnu (ou alors connu que de nom) pour beaucoup de chercheurs et étudiants en SHS. Cela peut être dû à son côté austère ou encore au fait qu'il peut paraître destiné uniquement à mettre en forme des mathématiques, capacité que l'on peut considérer comme secondaire voire superflue dans notre cas.

C'est quoi LaTeX ?

LaTeX est un système de composition de document qui permet de produire une mise en page la plus proche possible des normes typographiques d'une langue donnée. Ce système permet de se concentrer sur le fond et le contenu du document à produire en automatisant sa forme. Il fonctionne généralement dans un éditeur de texte – TeXmaker par exemple mais il en existe bien d'autres – dans lequel on écrit l'ensemble des commandes qui paramètrent le document avant de le compiler.

Un éditeur de texte n'est pas un traitement de texte, ce dernier comprenant la mise en forme alors que l'éditeur contient des lignes de code. L'exemple le plus connu des éditeurs est le bloc-note de Windows. Il comprend donc le texte brut que l'on veut éditer et de potentielles commandes informatiques qui formatent leur aspect dans le document final. Le traitement de texte doit donc en permanence rafraîchir la mise en forme du texte afin de montrer en même temps ce que l'on est en train d'écrire et le rendu final, là où l'éditeur ne montre que les lignes de code et ne montrerait le rendu final que s'il comprend un compilateur.

Mais quel est l'intérêt de l'utilisation de LaTeX ?

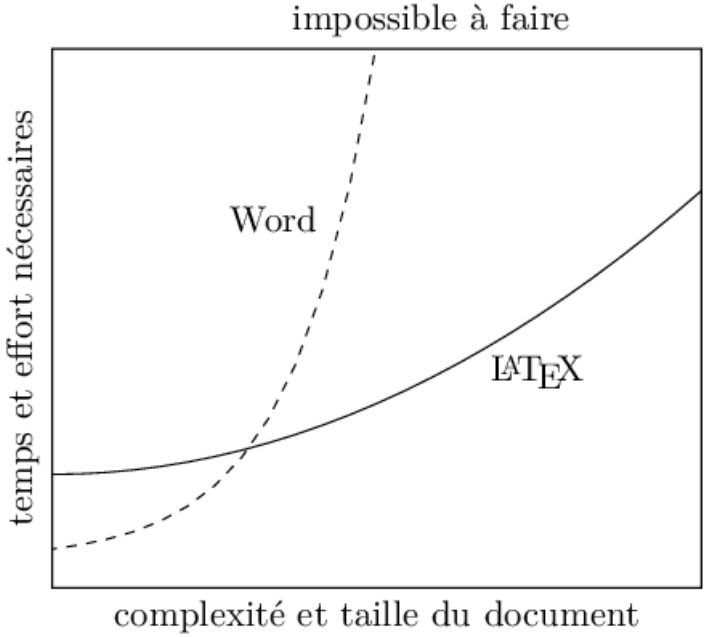
Beaucoup d'arguments sont énoncés concernant les gains à l'utilisation de LaTeX notamment par rapport à un traitement de texte du type Word, j'en retiens les principaux qui pourraient retenir l'attention pour une utilisation en SHS.

Comme dit plus haut, il est utilisé pour ses équations et cela peut paraître comme sa seule utilité. Pourtant, il s'agit avant tout d'un langage permettant *une mise en page harmonieuse*, ce qui est fortement requis dans un champs où l'on a tendance à beaucoup écrire. Débarrassé du besoin d'afficher le rendu du document en temps réel par rapport à un traitement de texte, LaTeX ne prend *in fine* le temps de calculer l'agencement des paragraphes, des titres, des tableaux, des figures etc. qu'en une seule fois et donc de manière plus rapide au total. Il peut ainsi offrir une qualité typographique supérieure nécessitant des calculs qui seraient insoutenables pour un traitement de texte. Ce dernier aspect permet de faire des fichiers bien plus volumineux comme des livres, thèses, mémoires etc. sans que l'interface ne plante : on aura alors simplement une compilation plus longue mais l'écriture n'est pas impactée. Par ailleurs, LaTeX permet d'automatiser une partie du processus de mise en page relativement facilement, en appelant un simple package ou en ajoutant une commande.

On peut multiplier les exemples de points où LaTeX est plus pertinent que Word comme le fait de ne pas dépendre d'un format propriétaire vu que le PDF est libre de droits et permet en plus de garder toujours le même aspect quel que soit l'ordinateur ou l'impression. Un autre point fort est sa gestion automatique des abréviations et des références internes ou externes (bibliographie, numérotation des

parties, des figures, des tableaux etc.) de manière très simple. Cela permet ainsi de ne se préoccuper de la forme du texte que dans un deuxième temps – ou même de ne presque plus s'en préoccuper si l'on a tout laissé par défaut ou que l'on a créé ses propres macros.

FIGURE 1 – Schéma de la marge de progression entre LaTeX et Word



Source : Pierre-Louis PEETERS. *Introduction à LaTeX*. PLPeeters.com. 4 sept. 2013. URL : `https://www.plpeeters.com/blog/fr/post/110-introduction-a-latex` (visité le 31/08/2022)

Enfin, avec RSweave on peut y intégrer des statistiques que l'on a produit avec R sans faire de copier-coller et d'aller-retour entre plusieurs logiciels. Plus globalement, la construction de documents est facilitée : la révision voire le travail collaboratif peut se faire avec l'utilisation de Git qui est intégrée à Rstudio notamment.

C'est donc pour ces raisons que je vous présente aujourd'hui cette petite introduction à LaTeX. Nous verrons dans un premier temps quels sont les notions de base du langage, avant de s'attarder sur les commandes principales puis nous verrons comment cela s'adapte à un document RSweave, c'est-à-dire en lien avec R.

1 Les principes de base de LaTeX

1.1 Installation

1.1.1 Windows

Le plus simple est de télécharger MikTeX. Cependant, j'ai eu pas mal de soucis de compatibilité sur Windows, notamment avec RSweave. N'étant pas fan de Windows, j'ai préféré opter pour une solution que j'avais sur MacOS et sur Linux : TeXLive. Vous le trouverez ici mais il vous faudra peut-être télécharger à part l'ISO pour éviter les échecs de téléchargement via l'installateur rapide. Toute la manipulation est expliquée là.

1.1.2 MacOS

Le plus simple cette fois est de directement télécharger TeXLive. Tout le nécessaire vient avec sans complication. Voici le lien correspondant : http://www.tug.org/mactex/.

1.1.3 Linux

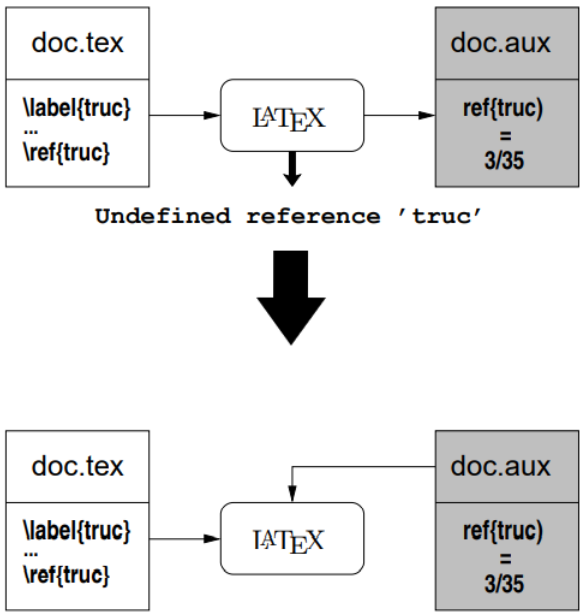
Sous Linux, il suffit de faire la commande suivante dans le terminal :

```
sudo apt-get install texlive-full
```

1.2 Notion de compilation

On l'a vu, LaTeX ne permet pas de visualiser le résultat en direct, ce qui fait que l'on doit compiler pour voir ce que ça donne. On a un peu simplifié en disant qu'il n'y avait qu'une seule compilation, en effet selon la complexité du document on peut en avoir plusieurs. Le cas le plus typique est celui des références internes au document (bibliographie, renvois vers une figure comme ici : 2). On aura alors une première compilation pour obtenir un fichier `.aux` qui stocke les étiquettes (on peut créer des labels ou des sections) que l'on a créées ainsi que les lignes dans lesquelles on les mentionne. Ensuite une deuxième compilation est lancée en prenant en compte le fichier `.aux`, ce qui ajoute les liens hypertextes si besoin mais surtout numérote les références correctement. Ainsi, LaTeX ne peut pas se tromper en numérotant une classe d'objet puisqu'il référence toutes les objets et les numérote sauf indication contraire.

FIGURE 2 – Schéma explicatif de la compilation



Source : Vincent LOZANO. *Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais oser le demander 1.0 ou comment utiliser LaTeX quand on n'y connaît goutte*. Cergy-Pontoise : In Libro Veritas, 2008

En cas d'apparition de la table des matières, le processus est le même mais avec un fichier `.toc` et pour la bibliographie, une compilation de plus avec `biber` ou `BibTeX` produit un fichier `.bbl`.

1.3 Notions principales

1.3.1 Packages

Si vous êtes familiers avec R, la notion de package ne vous est pas étrangère. Dans le langage LaTeX, on a la même logique : chaque fonction est définie dans un package que l'on se doit d'appeler avant d'appeler la fonction. En effet, cela demanderait bien trop de mémoire de charger l'ensemble des fonctions existantes dans la documentation. Cependant, d'expérience, la documentation et les ressources sur les forums sont bien plus nombreux, détaillés et accessibles ! Ce qu'il est important de noter donc, c'est

que si vous voulez réaliser quelque chose alors quelqu'un l'aura très probablement fait avant vous et il vous suffira de charger le package et donc les fonctions désirées. Dans le pire des cas, vous créerez une macro à partir d'une fonction existante.

1.3.2 Préambule

Contrairement à R où l'on fait ce qu'on veut, on appelle les packages dans le préambule du fichier `.tex`, c'est-à-dire à l'extérieur de notre environnement de texte – on y reviendra. On aura ainsi comme modèle de fichier `.tex` par défaut, un code impérativement délimité comme suit :

```
% Les commentaires sont indiques par un pourcentage
% On se trouve ici dans le preambule
\documentclass{article} % la classe de document est obligatoire, elle permet à LaTeX
↪  de charger des styles par defaut. Ici j'ai choisi article mais cela peut etre
↪  report, memoir, book, letter, ou bien d'autres
% On peut aussi creer sa propre classe

\usepackage{package1} % J'appelle ici package1, on verra plus tard lesquels peuvent
↪  nous être utiles

\begin{document}
% Ici on ne se trouve plus dans le preambule. L'ensemble de ce qui apparait dans le
↪  PDF se trouve la.
\end{document}
```

1.3.3 Classe

Comme mentionné plus haut, j'ai utilisé ici la classe de documents `article`. Celle-ci répond à certains critères notamment en terme de structure de document. On dénombre sept niveaux de titres au total :

Commande	Sens	Niveau
`\part{}`	Partie	-1
`\chapter{}`	Chapitre	0
`\section{}`	Section	1
`\subsection{}`	Sous-section	2
`\subsubsection{}`	Sous-sous-section	3
`\paragraph{}`	Paragraphe	4
`\subparagraph{}`	Sous-paragraphe	5

A titre d'exemple, la classe `article` ne commence qu'au niveau 1 alors que `book` commence au -1. Pour vous faire une idée, ce document utilise cette classe.

Voici quelques exemples de classes :
- `article` : pour des textes relativement cours comme des articles scientifiques ;
- `report` : pour des documents longs de plusieurs chapitres ;
- `book` : pour des livres de plusieurs centaines de pages ;
- `memoir` : pour écrire des mémoires (c'est ce que j'ai utilisé pour le mien, mais `report` semble également pertinent)

1.4 Environnement

Un environnement est un espace délimité dans lequel certaines règles précises et certaines fonctions ont cours. Il permet de créer ou d'importer des objets qui se définissent eux-mêmes comme des en-

vironnement aussi. L'environnement le plus utilisé est, logiquement `document` que l'on vient de voir. Il est obligatoire pour créer un document. Pour comprendre un peu mieux, prenons l'environnement `equation`. Dans celui-ci, les outils mathématiques – contenus dans le package `amsmaths` – ont un sens et n'en ont aucun en dehors. Voici un exemple :

```
\documentclass{article}
\usepackage{amsmaths}

\begin{document}
  \begin{equation}
  f(x)=\sqrt{x} % racine carrée
  \end{equation}
\end{document}
```

Cela donne :

$$f(x) = \sqrt{x} \tag{1}$$

une équation numérotée basique.

Certains environnements sont appelés *flottants* car ils ne font pas partie du texte et sont placés par LaTeX là où cela est le plus optimal possible compte tenu du texte au-dessus et en-dessous, on verra plus bas qu'il s'agit surtout des tableaux et figures.

2 Construction d'un document LaTeX

Nous pouvons maintenant passer à une partie un peu plus pratique.

2.1 Choix courants dans le préambule

Dans le préambule, on renseigne également le nom de l'auteur, le titre et la date. Le code du document présent comporte les commandes suivantes :

```
\documentclass[a4paper, 11pt]{article} % entre crochet on peut integrer des options
↪  a la commande de classe du document mais aussi aux packages
\usepackage{hyperref} % pour les liens hypertextes, ici cela ajoute le lien pour
↪  m'envoyer un mail
\usepackage{titling} % pour le titre en gras
\author{\href{mailto:julioricardo.davalos@ehess.fr}{Julio Ricardo Davalos}}
\title{Petite introduction à \LaTeX{} en Sciences Sociales}
\date{Septembre 2022}

% packages
\begin{document}
    \maketitle
%%%% La suite
\end{document}
```

Le titre apparaît alors comme vous pouvez le voir plus haut, avec la date de dernière compilation en mode par défaut. Si je veux forcer une date, je fais comme dans l'exemple (il est possible de formater la commande pour qu'elle ne dise pas le jour, mais c'est tout aussi simple pour notre niveau).

2.2 Environnements et packages usuels

2.2.1 Environnement `tabular` et ses dérivés

Comme son nom l'indique, c'est un environnement qui sert à construire des tableaux. Il n'est pas flottant par défaut, ce qui fait qu'on le positionne généralement dans un flottant comme `table` ou `figure` si jamais on veut mettre une légende par exemple. Ces flottants prennent des arguments qui ont trait à leur position dans la page.

```
\begin{table}[!ht] % permet de forcer la position "here top"
    \caption{Un tableau centré} % titre du table, ne marche que dans un flottant
    \centering % je centre dans la page
    \begin{tabular}{|c|c|} % 2 colonnes, texte centré, avec les lignes verticales
    ↪  tracées
    \hline %ligne horizontale
     Case $(1,1)$ & Case $(1,2)$ \\ %saut de ligne, "&" délimite une colonne
    \hline %ligne horizontale
    Case $(2,1)$ & Case $(2,2)$ \\
    \hline  %ligne horizontale
    \end{tabular}
\end{table}
```

Ce code donne :

TABLE 1 – Un tableau vide et centré

Case $(1,1)$	Case $(1,2)$
Case $(2,1)$	Case $(2,2)$

Les arguments de base de `tabular` permettent de sélectionner le nombre de colonnes, leur alignement (`c`, `l` et `r`) et l'apparition des bordures par le symbole |. Les lignes de séparation des colonnes ne sont pas obligatoires, ainsi, on peut supprimer la ligne du milieu en indiquant `|cc|`. On peut construire autant de lignes et de colonnes que l'on veut, faire autant de `tabular` dans un `table` que l'on veut.

Evidemment cela est un peu fastidieux à écrire, mais beaucoup d'éditeurs existent, par exemple ici, mais aussi dans l'éditeur TEXmaker qui est plutôt pratique bien que moins adapté à R. Pour fusionner des cases, il faudra faire appel à la fonction `\multicolumn{}` et `\multirow{}` (cette dernière, vient du package du même nom).

2.2.2 Environnement `figure`

Cet environnement est également essentiel car il permet d'ajouter des images, qu'elles soient en PDF ou en format image, voire produite par votre document, on le verra ensuite. Il nécessite le package `graphicx`.

```
\begin{figure}[!ht] % permet de forcer la position "here"
        \caption{Une image prise au hasard} % titre, bien placé dans un flottant
        \centering % on centre
        \includegraphics[scale=0.4]{Images/bourdieu.jpg} % insertion d'une image
        % taille = 40% de la largeur d'un paragraphe, on met le chemin de l'image
\end{figure}
```

La ligne 4 peut être utilisée toute seule pour insérer une image tout comme `tabular` mais celle si sera placée là où LATEX calcule qu'elle a sa place optimale et elle n'aura pas de titre, donc il vaut mieux la garder dans un flottant. Ce code donne :

FIGURE 3 – Une image choisie complètement au hasard

On peut également combiner les deux environnements en faisant des tableaux (avec ou sans lignes) contenant des images.

2.2.3 Environnements `itemize` et `enumerate`

Ces deux environnements sont très importants également car ils permettent de faire des listes. `itemize` permet de faire une suit de tirets (ou autre symbole que l'on choisit) et `enumerate`, une liste numérotée. Ils fonctionnent exactement de la même façon.

```
\begin{itemize}
  \item point 1
  \item point 2
\end{itemize}
```

Ce code nous donne :
- point 1
- point 2

On peut modifier les types de tirets ponctuellement en ajoutant ce que l'on veut entre crochets après la commande `item`[]

2.2.4 Packages d'encoding et de typographie

Les packages permettant d'écrire en français avec les bons accents et la typographie standard sont les suivants :
- `\usepackage[utf8]{inputenc}` : permet à LaTeX de comprendre l'encodage d'entrée, notamment pour les accents.
- `\usepackage[french]{babel}` : permet d'avoir les typographies françaises.
- `\usepackage[T1]{fontenc}` : permet d'encoder le PDF de sortie en T1.
- `\usepackage{hyperref}` : permet de créer des liens hyper-textes.
- `\usepackage[left=cm,right=cm,top=cm,bottom=cm]{geometry}` : permet de définir les marges.
- `\usepackage{tabularx}` : variante de `tabular` avec plus d'options.

2.3 Corps du texte

Nous avons vu plus haut les commande pour la structuration des titres. Concernant la rédaction, il y a plusieurs choses à savoir :
- Plusieurs espaces à la suite produisent une seule espace.
- Une ligne vide produit un changement de paragraphe. Plusieurs lignes vides produisent le même effet. Pour redémarrer avec l'indentation, on termine un paragraphe avec `\\`. Mais cette manière de faire n'est pas très propre, le mieux est de préciser dans le préambule quelle indentation on veut et quel espace entre les paragraphes on souhaite. Dans ce document j'ai simplement appelé le package `parskip` qui convient en version par défaut.

— Un retour à la ligne est compris comme une espace.
— Un espace *après* un signe de ponctuation est obligatoire, mais LaTeX comprend qu'il faut en faire un *avant* et l'insèreun espace si besoin. Par ailleurs, il vaut mieux ne pas séparer la ponctuation du mot précédent, sans quoi LaTeX risque de faire apparaître le signe sur la ligne suivante si le mot était le dernier de la ligne.
— quatre caractères spéciaux ne s'obtiennent pas avec \[mon caractère spécial] :
  1. \ : qui s'insère avec `textbackslash` ;
  2. ~ : qui s'insère avec `textasciitilde` ;
  3. ^ : qui s'insère avec `textasciicircum`.
— Les guillemets s'insèrent avec `\enquote{mon texte}`.
— L'italique s'obtient en insérant : `\textit{mon texte}`.
— Le gras s'obtient en insérant : `\textbf{mon texte}`.

Enfin, dans le corps du texte on trouvera des liens hypertextes que l'on marque avec `\label{mon label}` dans l'environnement auquel appartient le label puis que le pointe avec `\ref{mon label}`.

2.4 Bibliographie

Pour ce qui est de la bibliographie, beaucoup de configurations existent et ce document n'a pas vocation à en donner une vue globale tant cela serait long. On peut également écarter la question de la création des fichiers `.bib` car ils sont aisément produits par Zotero par exemple. Un outil important à ajouter à ce dernier est l'extension BetterBibLaTeX que l'on trouve ici et qui permet de standardiser ses sorties `.bib` et de les garder à jour. Ainsi, vos modifications dans Zotero se répliqueront dans le PDF final après re-compilation.

Une fois un fichier `.bib` joint au fichier `.tex`, on peut appeler le package `biblatex`. Pour une utilisation avec Rsweave, je recommande de mettre l'option `backend=biber` car ce processus fonctionne généralement mieux avec. Ensuite, selon vos préférences, vous pouvez régler les options selon votre style de citation favori. Zotero aura attribué une clé en fonction des préférences, par exemple « AuteurDate », cette clé servira alors à appeler la ressource bibliographique ainsi :

```
\cite{Rouquette2011}
```

donne ici :

Maïeul Rouquette. *(Xe)LaTeX Appliqué Aux Sciences Humaines*. 2011.

Je vous encourage d'ailleurs à aller voir la partie bibliographie de ce livre qui est particulièrement bien fournie et adaptée pour les SHS.

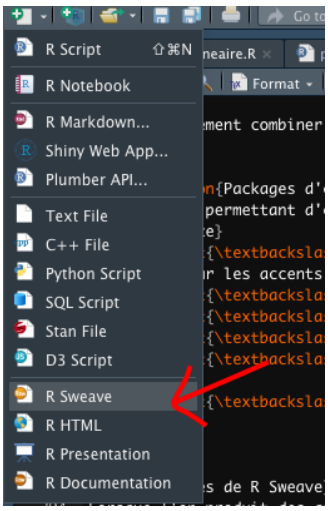
3 Mise en forme de résultats statistiques avec LaTeX

Nous allons maintenant revenir à ce qui nous intéresse en tant qu'ingénieurs ou chercheurs : la publication. Pour produire des traitements statistiques, nous pouvons notamment utiliser R. RSweave permet de lier R et LaTeX.

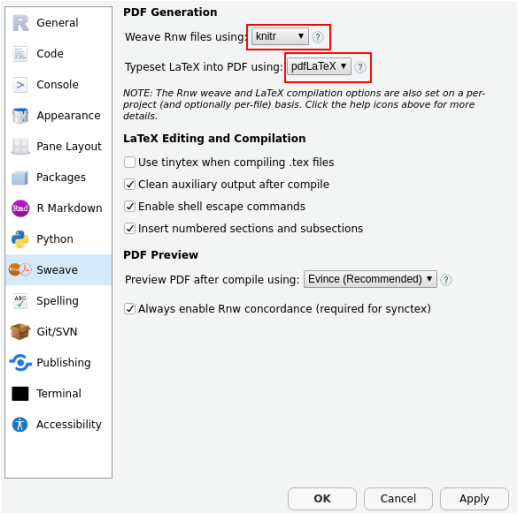
3.1 Logique de RSweave

3.1.1 Interface

RSweave est inclu dans Rstudio. On le trouve comme sur l'image qui suit :



Pour obtenir la version de Sweave la plus efficace et la plus documentée, il faut cependant sélection `knitr` (qui vient de Markdown), dans les paramètres. De même, comme vous avez installé LaTeX, vous pouvez sélectionner pdfLaTeX et oublier `tinytex` qui est une version légère de LaTeX.



3.1.2 Notion de Chunk

Le chunk est l'objet qui permet de faire le lien entre LaTeX et R. Il s'insère dans un fichier `.Rnw` via le code suivant :

```
# <<>>=
print("une commande R")
@
```

qui donne :

```
## [1] "une commande R"
```

NB : ce fichier étant codé sur RSweave, j'ai dû ajouter des dièses pour qu'il ne prenne pas en compte le Chunk. Si quelqu'un a une solution pour réparer ça, j'en serais ravi.

Les chunks doivent comprendre toutes les commandes du code R dont on a besoin, y compris le chargement de répertoire, des bases de données etc. Il est possible d'en imprimer le contenu, la sortie ou les deux dans le PDF LaTeX. Il dispose de différents arguments que l'on va utiliser ici :

— `eval=[T ou F]` : permet de lancer le code ou non. *Sa valeur par défaut est* ***T***.
— `echo=[T ou F]` : permet d'imprimer ou non le code R contenu dans le chunk. *Sa valeur par défaut est* ***T***.
— `fig=[T ou F]` : permet d'avoir une sortie de type figure ou non. Cela revient à faire \includegraphics, il vaut donc mieux penser à ajouter un environnement `figure` autour comme on l'a vu plus haut, même si cela n'est pas absolument nécessaire. *Sa valeur par défaut est* ***F***.
— `results=` : permet d'afficher une sortie R paramétrée pour être au langage LaTeX ou autres, pour LaTeX on remplit par "asis". *Sa valeur par défaut est* ***hide***.
— `label=` : permet de donner un nom à la figure afin de potentiellement y faire appel plus tard.
— `fig.width=` et `fig.height=` : permet de donner une taille à la sortie s'il s'agit d'une image (par exemple un graphique).

3.1.3 Le hic : la bibliographie

Il devait bien y avoir un hic dans cette histoire. Il s'agit de l'importation de la bibliographie. RStudio ne permet pas de lier `biber` à notre fichier de manière automatique. Cela implique qu'en théorie, on ne peut pas mettre de bibliographie dans un fichier Rsweave compilé depuis Rstudio. Deux solutions existent pour ce problème :

1. Vous utilisez un autre compilateur, par exemple TeXmaker qui lui est modifiable et à qui on peut donc dire où sont les outils dont il a besoin pour réaliser chaque tâche. Un autre exemple est Emacs qui comprend énormément de langages et vous permettra de faire à peu près tout. Mais cette solution implique que l'on se passe de l'interface de Rstudio qui est pourtant très ergonomique et moins aride.
2. Vous restez sur Rstudio et vous ajoutez en fin de document, le code suivant :

```
\end{document}
# <<setup, eval= TRUE, include= FALSE, cache= FALSE, echo= FALSE>>=
system (paste ("biber", sub ("\\.Rnw$", "", current_input())))
@
```

cela va forcer `biber` à compiler le document. Il faudra cependant compiler deux fois à chaque fois que vous citez une nouvelle entrée bibliographique.

3.2 Exemples de tableaux

On peut maintenant créer une base de données, je reprends ici une idée d'exemple reproductible que j'avais fait pour autre chose.

```
# <<>>=
data <- data.frame(choix = sample(c("fraise", "mangue", "orange", "cerise"),
                                  1000, replace = T,
                                  prob = c(0.5, 0.2, 0.2, 0.1)),
                   aime_kiwi = sample(c("Oui", "Non"), 1000,
                                      replace = T, prob = c(0.2, 0.8)),
                   aime_melon = sample(c("Oui", "Non"), 1000,
                                      replace = T, prob = c(0.5, 0.5)),
                   aime_ananas =  sample(c("Oui", "Non"), 1000,
                                      replace = T, prob = c(0.7, 0.3)))
@
```

On veut les fréquences des « choix », que l'on fait avec une fonction de base.

```
# <<>>=
library(questionr)
freq(data$choix)
@
```

```
##          n    % val%
## cerise  25  8.3  8.3
## fraise 158 52.7 52.7
## mangue  49 16.3 16.3
## orange  68 22.7 22.7
```

Le résultat n'est pas satisfaisant. Avec le package `kableExtra`, on peut faire bien mieux :

```
# <<result="asis">>=
# result="asis" permet de dire à Sweave que le Chunk va contenir du texte
library(kableExtra)
library(tidyverse)
freq(data$choix) %>%
 kable(format = "latex", # sinon c'est en html
       caption = "Fréquence des choix",
       label = "freqchoix",
       booktabs = TRUE) %>%  # permet d'avoir des tableaux plus jolis, necessite le
        ↪  package booktabs dans le preambule
    kable_styling(latex_options = c("hold_position"), # cree un float et lui donne
     ↪  l'argument 'h'
                  position = "center") # centre le tableau
@
```

TABLE 2 – Fréquence des choix

	n	%	val%
cerise	25	8.3	8.3
fraise	158	52.7	52.7
mangue	49	16.3	16.3
orange	68	22.7	22.7

Ce package permet de prendre en compte beaucoup de paramètres dans les tableaux. Par exemple, on peut les mettre sur une page unique, en format paysage, avec des cases de couleur etc. Je ne peux que vous recommander la lecture de sa documentation ici. A partir de maintenant, je ne mettrai plus que les codes de R grâce à l'argument `echo=T` et non plus le chunk en entier.

Pour le cas des tableaux de régression, un package dédié et très connu a été développé : `stargazer`. Imaginons que l'on veuille faire des modèles de régression dans le rendu

```
library(glm2)
data <- data %>%
 mutate(choix_fraise = case_when(choix == "fraise" ~ "Oui", TRUE ~ "Non") %>%
          as.factor(),
        choix_cerise = case_when(choix == "cerise" ~ "Oui", TRUE ~ "Non") %>%
```

```
         as.factor(),
        age = sample.int(100, 1000, replace = T),
        nb_fruits = (aime_kiwi == "Oui") + (aime_melon == "Oui") +
         (aime_ananas == "Oui") + abs(rnorm(1000, mean = log(age))))
# j'ai ajouté un bruit aléatoire en fonction de l'âge exprès

m1 <- glm(choix_fraise ~ age + aime_kiwi + aime_melon + aime_ananas,
          family = binomial, data = data)
m2 <- glm(choix_cerise ~ age + aime_kiwi + aime_melon + aime_ananas,
          family = binomial, data = data)
m3 <- lm(nb_fruits ~ age + aime_kiwi + aime_melon + aime_ananas, data = data)

library(stargazer)
stargazer(m1,m2, m3, title = "Trois beaux modèles")
```

TABLE 3 – Trois beaux modèles

	Dependent variable :		
	choix_fraise	choix_cerise	nb_fruits
	logistic	*logistic*	*OLS*
	(1)	(2)	(3)
age	−0.004	0.011***	0.027***
	(0.002)	(0.004)	(0.001)
aime_kiwiOui	0.037	−0.358	0.978***
	(0.157)	(0.297)	(0.083)
aime_melonOui	−0.081	−0.011	0.947***
	(0.127)	(0.222)	(0.067)
aime_ananasOui	0.004	−0.088	0.955***
	(0.137)	(0.234)	(0.072)
Constant	0.191	−2.791***	2.333***
	(0.178)	(0.327)	(0.094)
Observations	1,000	1,000	1,000
R^2			0.524
Adjusted R^2			0.522
Log Likelihood	−691.623	−302.110	
Akaike Inf. Crit.	1,393.246	614.221	
Residual Std. Error			1.057 (df = 995)
F Statistic			273.853*** (df = 4 ; 995)
Note :			*p<0.1 ; **p<0.05 ; ***p<0.01

Là encore, la fonction permet énormément de choses, beaucoup d'arguments sont utilisables ensemble. On peut notamment changer les noms des variables, ajouter des en-têtes, changer les coefficients par exemple pour des odds-ratio, calculer des statistiques d'ajustement etc. Des styles propres à des revues scientifiques sont pré-enregistrés dans la fonction pour faciliter son utilisation.

3.3 Exemple de figure

Pour les figures, cela est très facile également. Essayons avec un graphique.

```
# dans les arguments du chunk, on a mis:
# fig.cap="Nombre de fruits estimés en fonction de l'âge"
# fig.lp="nbfruits_age" # (label)
# fig.pos="ht!"
# fig.width=7
# fig.height=3
# fig.env="figure"
# echo=T

library(ggplot2)
data %>%
 ggplot(aes(x = age, y = nb_fruits)) +
 geom_point() +
 stat_summary(fun.data = mean_cl_normal) +
 geom_smooth(method='lm')
```

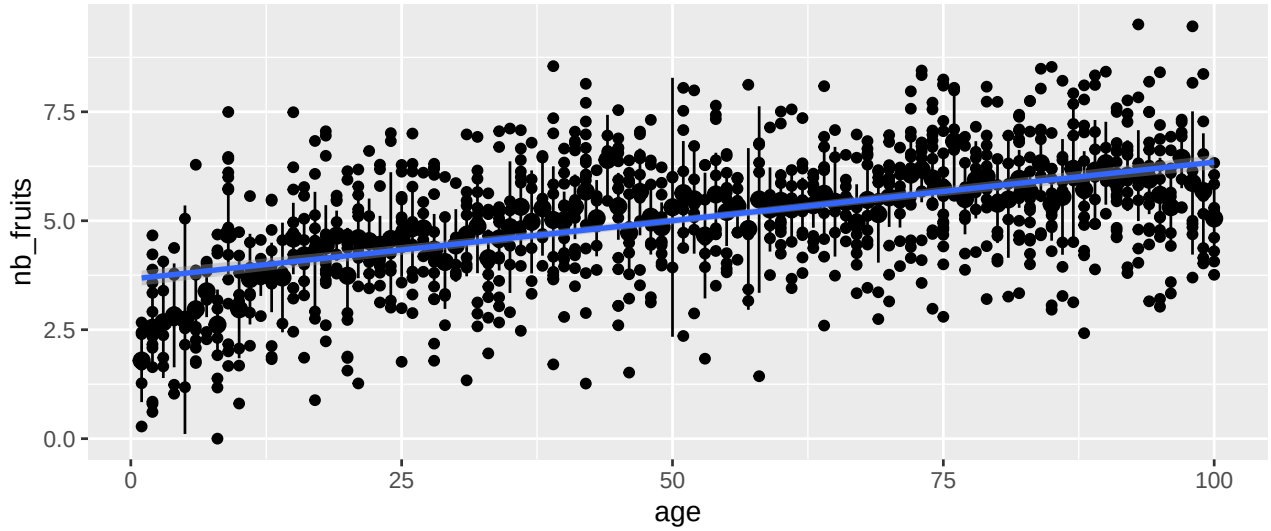


FIGURE 4 – Nombre de fruits estimés en fonction de l'âge

Pour plus d'options dans les chunk : `https://yihui.org/knitr/options/#chunk-options`.

Références

COTTRELL, Allin. « Traitements de texte : stupides et inefficaces ». In : (), p. 14.

DEPRIESTER, Dorian. *LaTeX : introduction aux présentations Beamer*. Le Blog de Dorian. 29 jan. 2012. URL : `https://blog.dorian-depriester.fr/latex/beamer/latex-introduction-aux-presentations-beamer` (visité le 06/09/2022).

LaTeX — Wikilivres. URL : `https://fr.wikibooks.org/wiki/LaTeX` (visité le 06/09/2022).

LOZANO, Vincent. *Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais oser le demander 1.0 ou comment utiliser LaTeX quand on n'y connaît goutte*. Cergy-Pontoise : In Libro Veritas, 2008.

PEETERS, Pierre-Louis. *Introduction à LaTeX*. PLPeeters.com. 4 sept. 2013. URL : `https://www.plpeeters.com/blog/fr/post/110-introduction-a-latex` (visité le 31/08/2022).

ROUQUETTE, Maïeul. *(Xe)LaTeX Appliqué Aux Sciences Humaines*. 2011.