Introduction aux documents dynamiques avec R

Stéphane Le Vu
stephane.le-vu@santepubliquefrance.fr

DMI

26 juin 2019

Introduction

Objectifs

- Comprendre l'intérêt d'un document dynamique
- Apprendre à écrire un fichier R Markdown
- Compiler le document en plusieurs formats

Intérêt

- Documents contenant les étapes de l'analyse
  - Code
  - Résultats
  - Illustrations
  - **Commentaires**
- Avantages
  - Automatisé, reproductible, traçable
  - Sépare structure et contenu
  - Evite les copier-coller et les documents *"résultats.final.base_provisoire.v2.3_bis"*
- Limites
  - Pas de mode de révision type Word
  - Longueur du code
  - Temps de compilation

Code couleurs

- *“fichier”*
- `code`
- url
- exemple

Bases de R nécessaires

- Savoir installer et charger un package

```
install.packages("le_package")
library(le_package)
```

- Savoir que `package::objet` permet d'accéder à un objet d'un package[1] . Par exemple, la fonction *fn1* du package *le_package* est invoquée par

```
le_package::fn1
```

1. La notation `package:::objet` (trois fois deux-points) accède à un objet interne (en quelque sorte 'caché') d'un package

Installation

- Si RStudio n'est pas installé

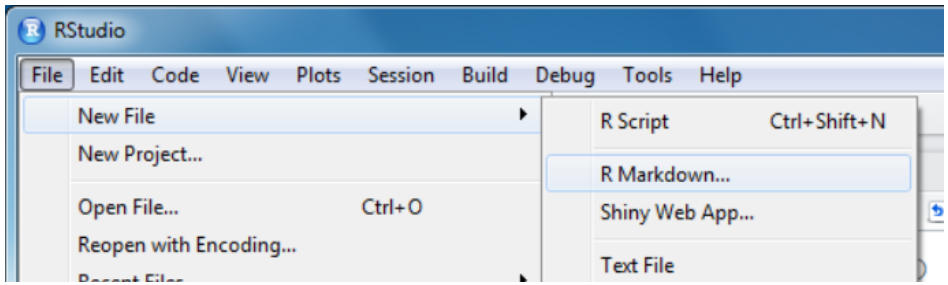
```
install.packages("rmarkdown")
```

- Si LaTeX n'est pas installé (optionnel, permet de générer des documents pdf)

```
install.packages('tinytex')
tinytex::install_tinytex()
## vérifier
tinytex:::is_tinytex()
```

Test installation complète

- Pour vérifier que R, RStudio et LaTeX sont bien installés
- Dans RStudio, ouvrir le menu *File* > *New File* > *R Markdown*



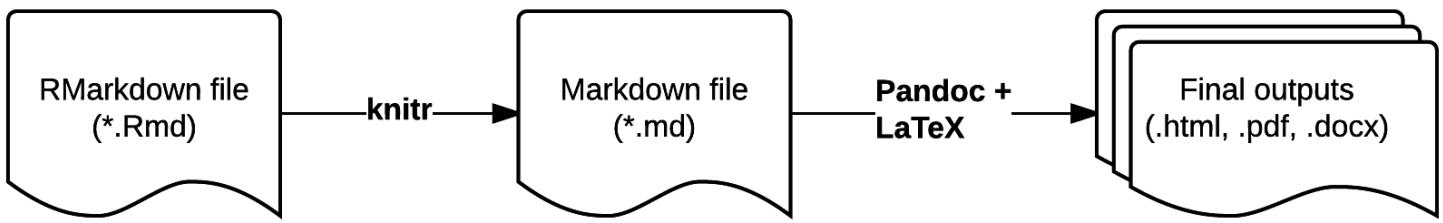
- Sélectionner *Documents* à gauche et *PDF* à droite
- Cliquer sur le bouton 

ou `Ctrl + Maj + K`
- Eventuellement accepter l'installation de packages
- Cela doit produire un PDF [2]

2. Par défaut, RStudio crée un nouveau fichier R Markdown avec un code d'exemple. Pour la suite, ce code peut être effacé, en gardant le préambule

Outils

- Compris dans le package rmarkdown
  - **Markdown** : langage balisé (*markup*), cad du texte avec des balises de format [3]. Sa version RStudio s'appelle *R Markdown*, comme le package *rmarkdown* !
  - **knitr** : un autre package *tricote* le code R avec le texte Markdown d'un fichier *.Rmd* en un fichier *.md*
  - **Pandoc** : programme qui permet la conversion de Markdown en presque tous les formats de documents [4]
- Non compris dans rmarkdown
  - **LaTeX** : langage de rédaction privilégié des mondes universitaire et scientifique



3. comme du *html* ou du LaTeX, mais en plus simple et lisible
4. dont *.pdf* pour impression et *.html* pour écran et web

Balises

LaTeX

```latex
% commentaire
\documentclass{article}
\begin{document}
\section*{Titre}
\subsection*{Sous-titre}
\emph{Italique}
\textbf{gras}
\end{document}
```

Markdown

```markdown
<!-- commentaire  -->
# Titre
## Sous-titre
*Italique*
**gras**
```

Exemples

- Les exemples sont contenus dans des projets RStudio distincts
- Chacun s'ouvre dans une nouvelle session de R (En cliquant par exemple sur *"ex01.Rproj"* compris dans le dossier *"ex01"*)
- Ils sont indiqués en rouge dans la présentation
- Le fichier d'exemple s'ouvre alors via le navigateur de RStudio (En cliquant *"ex01.Rmd"*) et généralement est compilé en cliquant sur *Knit*
- Si une commande R vous échappe dans un des exemples, on peut en parler après [5]

5. Les exemples ne comprennent pas de *pipe* `%>%`, *tidyverse*, *ggplot2*, *etc.*

Pratique

Texte markdown et code R

Premier rapport

- Préambule en YAML
- Comprend au moins le format de sortie entouré de `---`

```
---
title: "Mon premier document RMD"
author: Richard Virenque
date: 27/06/2019
output: html_document
---
```

- Ouvrir ex00
- Créer un nouveau fichier par *File* > *New File* > *Text File*
- Ecrire ce préambule
- Sauver (`Ctrl + S`) en `ex00.Rmd` et *Knit*

Syntaxe RMarkdown

- Permet de mettre en forme du simple texte
- Ouvrir ex01 et compiler (*Knit*)
- Lire la sortie
- La syntaxe est résumée dans cet aide-mémoire_rmarkdown

Premier morceau de code R

- Le morceau comprend une accolade `{langage nom, options}` et du code, le tout entouré par trois accents graves (*backticks*) [6]
- Le nom du morceau ne doit pas contenir de tiret bas `_`
- Le raccourci `Ctrl + Alt + I` crée (*insert*) un nouveau morceau (*chunk*)

````
```{r morceau1}
x <- 4
x
x < 0
```
````

- Ouvrir ex02.0 et compiler (*Knit*)

6. Nommer tous les morceaux, avec des noms différents, permet de débugguer plus facilement

Options du morceau

- Principales options
  - `echo = FALSE` masque le code
  - `results = 'hide'` masque la sortie
  - `include = FALSE` masque code et sortie[7]
  - `eval = FALSE` n'exécute pas le code
  - `warnings = FALSE` masque les avertissements
- Modifier l'exemple ex02.0 en testant ces options

7. Le code est quand même exécuté

Options globales des morceaux

- Les options peuvent être globales à tous les morceaux (voir ex02.1), avec

```
knitr::opts_chunk$set(...options...)
```

- Attention à garder la traçabilité de l'analyse (voir ex02.2)

Environnement

- Le document Rmd a son propre environnement, indépendant de la session dans laquelle il est ouvert
  - Les objets doivent y être créés
  - Les packages doivent y être chargés
  - Les chemins doivent être relatifs à l'emplacement du script *.Rmd*
- Exécuter ex03.0 et corriger

Solution de ex03

- Dans l'exemple ex03.1

Code inline

- Le texte Markdown peut inclure du code R, dit *inline*
- Cela s'écrit entre deux accents graves avec l'expression R précédée de `r`, comme cela : `` `r 1+1 == 3` ``
- Essayer ex04

Equations, figures et tableaux

Equations

- Utilisent la syntaxe TeX
- Les symboles sont généralement précédés d'une barre oblique inversée `\` (*backslash*)
- Dans une phrase, l'équation est balisée par `$...$`
- Par exemple, `$ A = \pi \times r^2 $` donne $A = \pi \times r^2$
- L'équation séparée est délimitée par `$$...$$` ou `\[...\]`
- Par exemple, `$$\hat{R_1} \approx \frac{A_1}{B}$$` donne :

$$\hat{R_1} \approx \frac{A_1}{B}$$

- Les symboles sont documentés ici : aide_formules_TeX

Figures et images

- L'insertion de figures R est simple
  - Les figures générées par R dans un morceau s'affichent dans la sortie
  - Les images externes sont insérées depuis un morceau par `knitr::include_graphics("chemin_image.ext")` ou directement dans le texte markdown par `![légende](chemin_image.ext)` [8]
- Le formatage requiert des options
  - `fig.width` et `fig.height` (en *inch* ou cm) modulent les figures R,
  - `out.width` (en %) module figures et images
  - `fig.cap` définit une légende
- Voir ex05

8. L'extension peut être *.png*, *.jpeg*, *.pdf*, *etc.*

Tableaux

- L'insertion de "beaux" tableaux est plus difficile
- Voir ex06 pour les tableaux html et ex07 pour les tableaux pdf

```
cars[1:3,]
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
```

```
knitr::kable(cars[1:3,],
 format = "latex",
 booktabs = TRUE)
```

speed	dist
4	2
4	10
7	4

Cache, script externe et références

Cache des morceaux

- A chaque compilation, tout le code est exécuté *de novo*
- Pour garder le résultat d'un morceau, on utilise l'option `cache = TRUE`
- Ce résultat est enregistré séparément et réutilisé sans que le morceau ne soit ré-exécuté [9]
- C'est dangereux si ce résultat dépend d'un élément (précédent) susceptible de changer
- Dans ce cas, on peut déclarer la dépendance par `dependson = "morceau_n-1"`
- Voir ex08

9. Tant qu'il n'est pas modifié

Intérêt d'un script R externe

- Plus pratique de tester son code dans un simple script R
- Evite des doublons de code
- Permet d'utiliser le même script pour plusieurs rapports
- Améliore la lisibilité du rapport

`script.R`
`rapport.Rmd`

Utilisation d'un script R externe

1. Utiliser tout un script R (ou plusieurs scripts différents) pour alimenter un rapport Rmd, avec `source("script.R")`, par exemple pour importer des données

- Ouvrir ex09, compiler ex09.1 et corriger

2. Utiliser des morceaux[10] d'un ou plusieurs scripts avec `knitr::read_chunk("script.R")`
   - Les morceaux "externes" sont alors appelés par `{r nom-du-chunk-externe}`

- Ouvrir ex09, compiler ex09.2 et corriger

10. Les morceaux doivent être nommés et délimités par un bloc de 4 tirets `----`.

Solution de ex09.2

- Dans ex10

Insertion de références

- Des références bibliographiques[11] peuvent être insérées dans le texte *markdown* par leur identifiant précédé de `@` :
  - `@identifiant`, pour une insertion dans la phrase
  - `[@identifiant]`, en fin de phrase
- Le fichier contenant les entrées bibliographiques et le style de citation désiré (fichier *csl*)[12] sont déclarés dans le préambule YAML par `bibliography:` et `csl:`
- Après compilation, la bibliographie s'ajoute à la fin du texte

11. au format *BibTeX, RIS, Mendeley, EndNote, etc.*
12. Le style *csl* correspond à un format de citation d'un journal par exemple

Exemple d'utilisation de références

- Ceci est une entrée BibTeX :

```
@article{identifiant,
  title={...},
  author={...},
  journal={...},
  pages={...},
  year={...},
}
```

- Voir ex11 et les différents styles csl

Formats de sortie

Formats de sortie

- La sortie est définie dans le préambule YAML par `output:`
- Les sous-options doivent être tabulées
- Par exemple :

```
output:
  html_document:
    toc: yes
```

Sorties documents

- `html_document`
- `pdf_document`
- `word_document`
- aussi `odt`, `rtf`, `latex`, etc.
- Voir ex12 et compiler ex12.1 avec *Knit to ...*
  - *html*
  - *pdf*
  - *word*

Sorties présentations

- `ioslides_presentation`
- `slidy_presentation`
- `revealjs::revealjs_presentation`
- `beamer_presentation` (pdf)
- `powerpoint_presentation` [13]
- Voir ex12 et compiler ex12.2 avec *Knit to . . .*

13. Non testé. Nécessite Pandoc v2.0.5

Autres formats

- Cahier d'analyse (*R Notebook*)
  - Permet d'exécuter les morceaux un par un interactivement [14]
- Article, mémoire, livre
  - Pour documents à plusieurs sections/chapitres et références internes (très courant avec LaTeX, voir bookdown)
- Poster
  - Plusieurs solutions à partir de LaTeX et Beamer mais en développement pour R Markdown (voir posterdown)
- Shiny
  - Voir Document Shiny et App Shiny
- Tutoriel interactif
  - Voir learnr

14. L'intérêt n'est pas évident par rapport à un ou plusieurs scripts R appelés dans un document Rmd

Rapports itératifs

Rapports itératifs

- Le principe est de créer des plusieurs rapports différents à partir d'un rapport type
- Le rapport type *Rmd* contient des paramètres dans le préambule YAML qui peuvent changer via un programme
- Dans l'exemple ex13, nous allons générer des rapports d'analyse par région
- Le script *ex13.R* utilise la fonction `rmarkdown::render()` [15] pour compiler le rapport type *ex13.Rmd* pour chaque région
- Commencer par ouvrir *ex13_instructions.txt*

15. l'équivalent du bouton *Knit*

Exemple complet

Présentation de l'exemple

- Nous allons créer un document illustrant l'estimation de la constante Pi par simulation de Monte-Carlo
- Le principe est de tirer aléatoirement des coordonnées et d'utiliser la proportionnalité entre le nombre de points inscrits dans un cercle et π

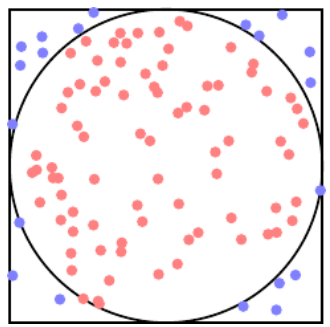


FIGURE 1 – Points inscrits dans le cercle

Instructions

- Ouvrir ex14
- Utiliser le document *draft_ex14* pour créer un fichier *ex14.Rmd*
- Générer un document contenant :
  - Le principe de l'analyse accompagné des équations
  - L'image *mcpi_fig0.png*
  - Le code de l'analyse, appelé du script externe *mcpi.r*
  - Le résultat chiffré en code *inline*
  - La figure générée
  - La référence contenue dans *ref_mcpi.bib*
- Faire deux rapports générés par `rmarkdown::render()` avec deux valeurs de *n*

Solution de l'exemple complet

- Ouvrir ex15
- Compiler (*Knit*) *ex15.Rmd*
- Ouvrir *ex15_iterations.R* et compiler avec *Source*

Documentation

Documentation complémentaire

- Générales
  - Documentation rmarkdown officielle
  - Forum stackoverflow
  - Introduction en français
  - aide-mémoire_rmarkdown
- Particulières
  - Options des morceaux de code
  - Options YAML des sorties html
  - Options YAML des sorties pdf

Notes : l'enfer de l'encodage du français (1/3)

- Dans RStudio, changer l'encodage des sources : *Tools* > *Global Options* > *Code* > *Saving* > *UTF-8*

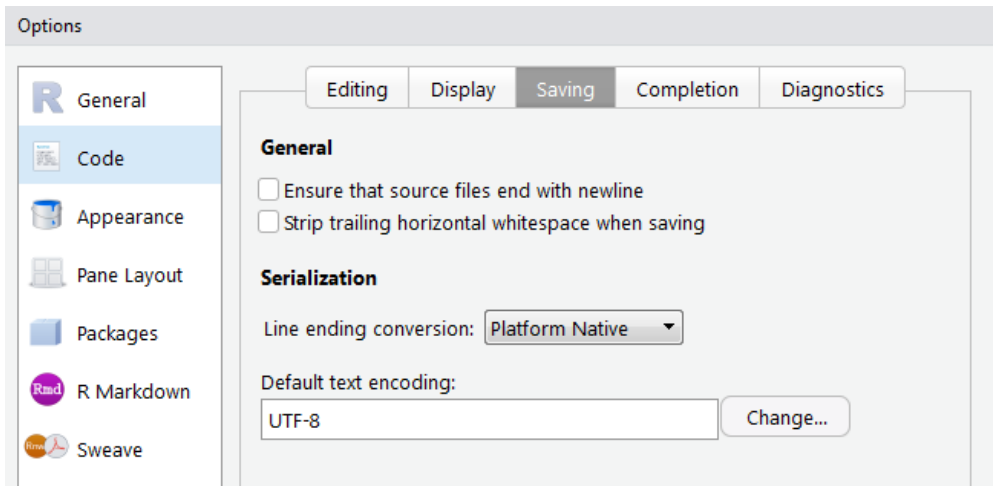


FIGURE 2 – Option globale UTF-8

Notes : l'enfer de l'encodage du français (2/3)

- De même, le projet requiert ce paramétrage : *Tools* > *Project Options* > *Code Editing* > *Text encoding* > *UTF-8*

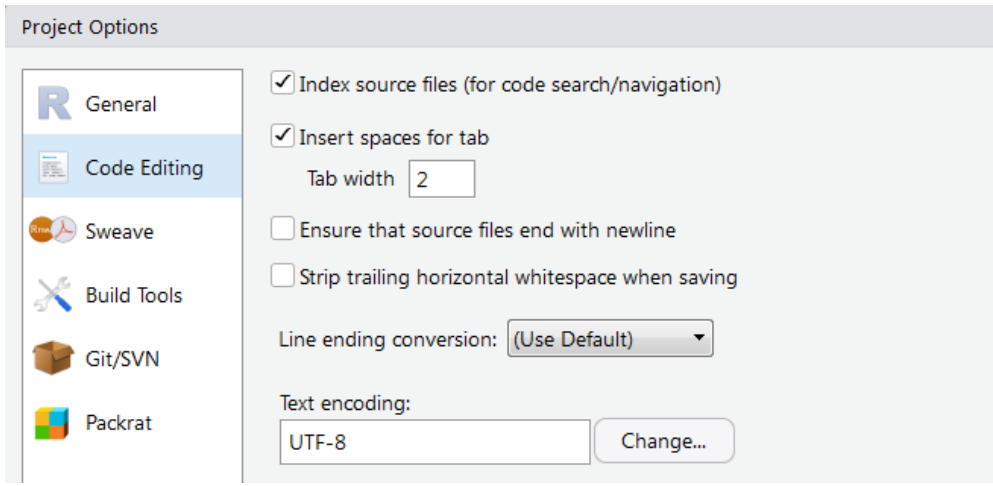


FIGURE 3 – Option projet UTF-8

Notes : l'enfer de l'encodage du français (3/3)

- Si les caractères accentués s'affichent mal, c'est généralement que l'encodage n'a pas été précisé, par exemple :
  - Convertir les scripts "mal" encodés avec *File > Reopen with encoding > UTF-8*
  - Ouvrir les tableurs avec l'option : `read.csv("fichier.csv", encoding = "UTF-8")`
  - Remplacer `knitr::read_chunk(path = "code.R")` par

```
knitr::read_chunk(
  lines = readLines("code.R", encoding = "UTF-8")
)
```

Matériel de cette formation

https://slevu.github.io/formation-Rmd/