A short tutorial on RMarkdown and knitr

Julien Stoehr

stoehr@ceremade.dauphine.fr

Université Paris-Dauphine

CIRM, Marseille – 22 Octobre 2018

Introduction

A common situation:

1. Manipulating data
2. Statistical analysis
3. Writing a report
4. Changing the data, the code, the analysis, ...

Principle: gathering in a single document comments/text, R code chunks and R output.

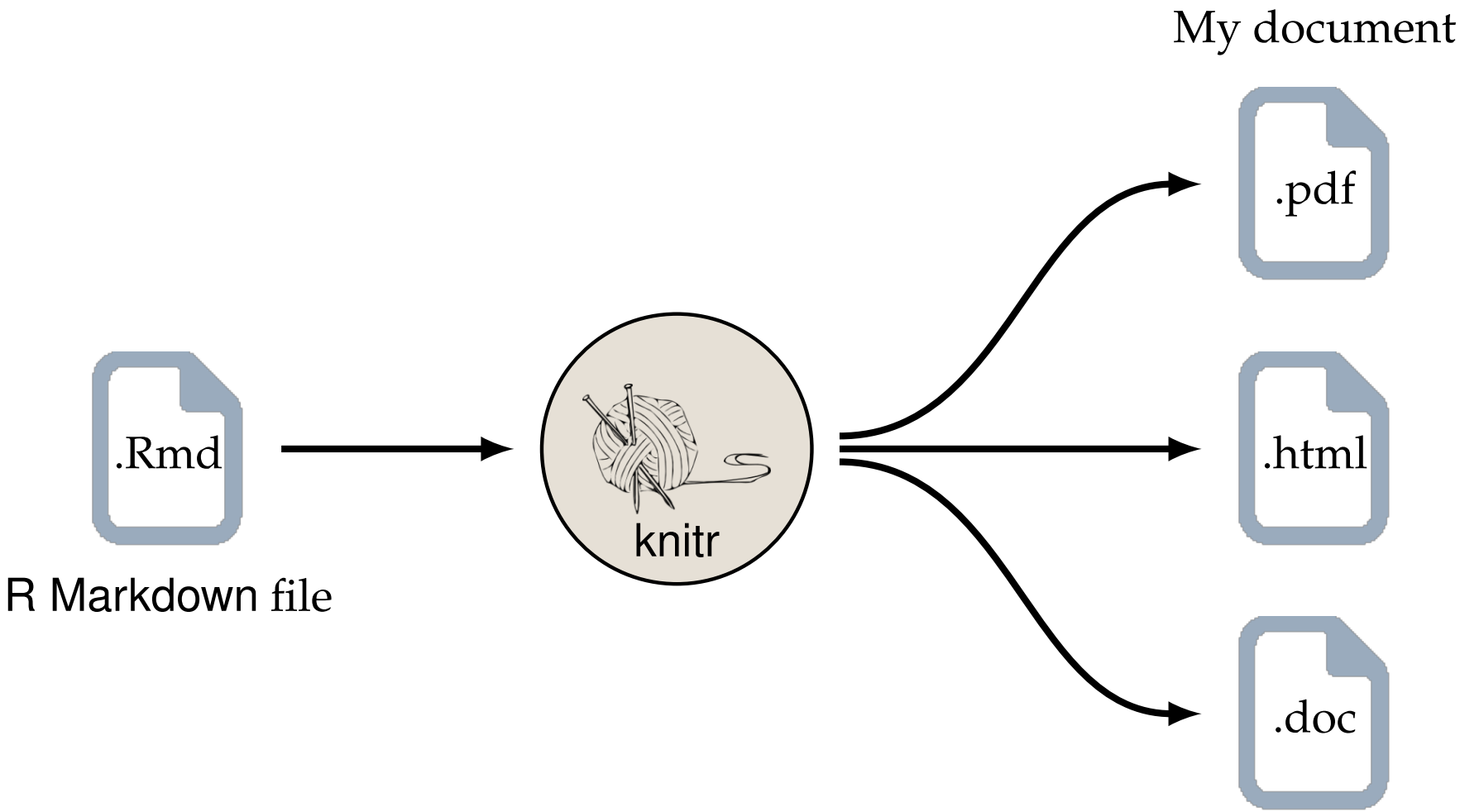
What is R Markdown?

Markdown by John Gruber & Aaron Swartz:

- Easy-to-read and easy-to-write plain text format (HTML code but without tags)
- Can be converted into HTML code
- Usual HTML code can also be added

RMarkdown: Markdown + R code chunks

knitr by Yihui Xie



What do I need?

R Sweave and R Markdown files:

- **Using RStudio:** all you need is already installed. You just need to install the `knitr` package if you want to use it.

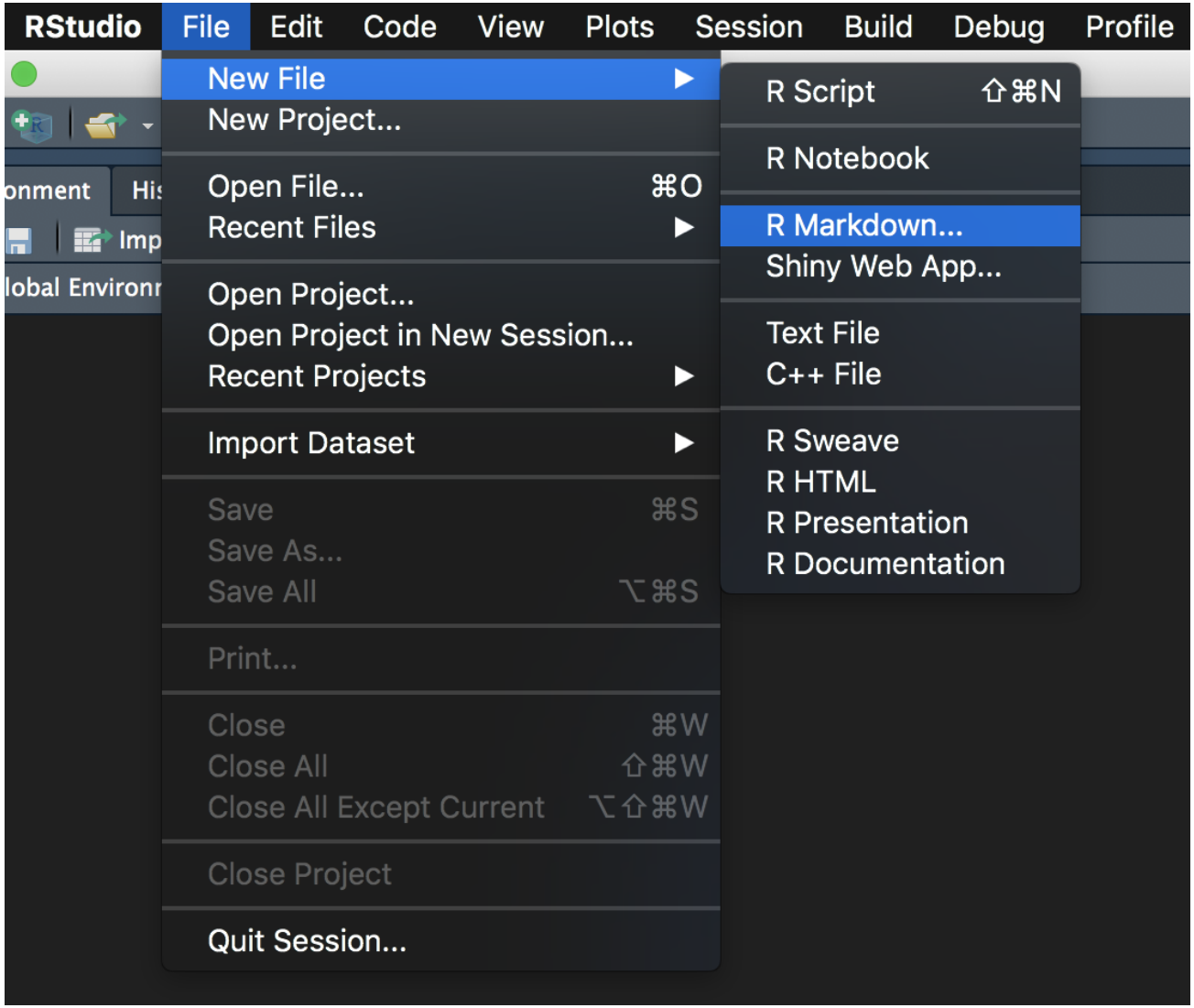
```
install.packages("knitr", dependencies = TRUE)
```

- **Without RStudio:** the `rmarkdown` package and its dependencies are required.

```
install.packages("rmarkdown", dependencies = TRUE)
```

To generate PDF files: a LaTeXdistribution is needed (MikTeX, MacTeX, TeX live, ...) is required to generate PDF documents.

Starting with R Markdown



Configuration and compilation of the document

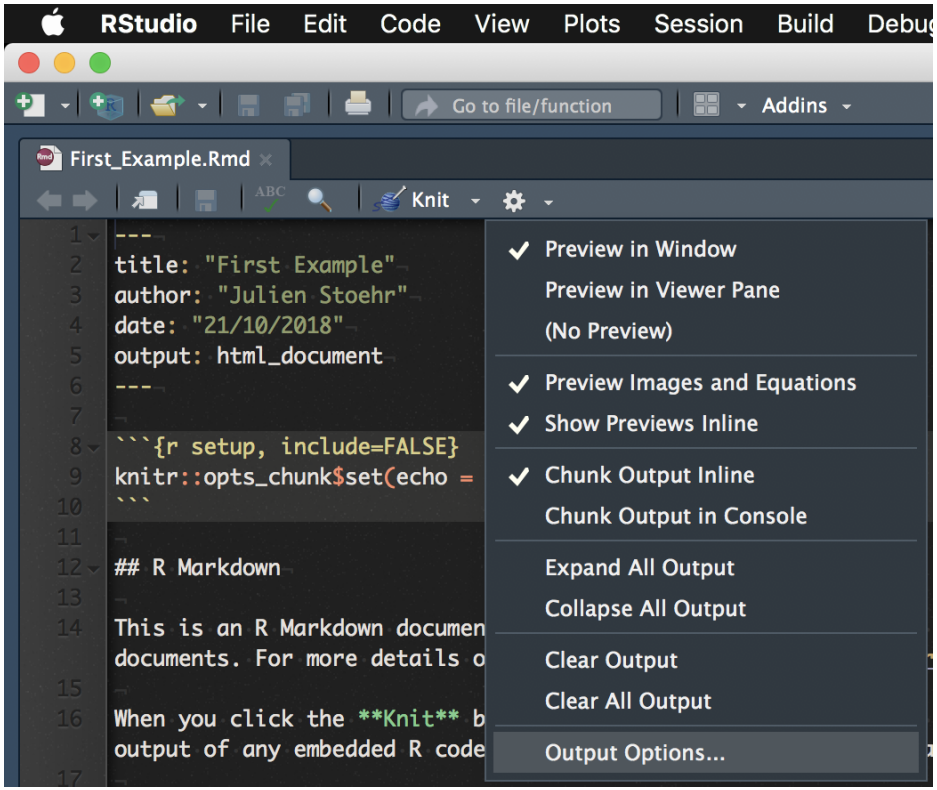
A YAML header:

```
---
title: "First Example"
author: "Julien Stoehr"
date: "22/10/2018"
output: html_document
---
```

- Defines the metadata
- Defines the output format
- Defines options of the output format

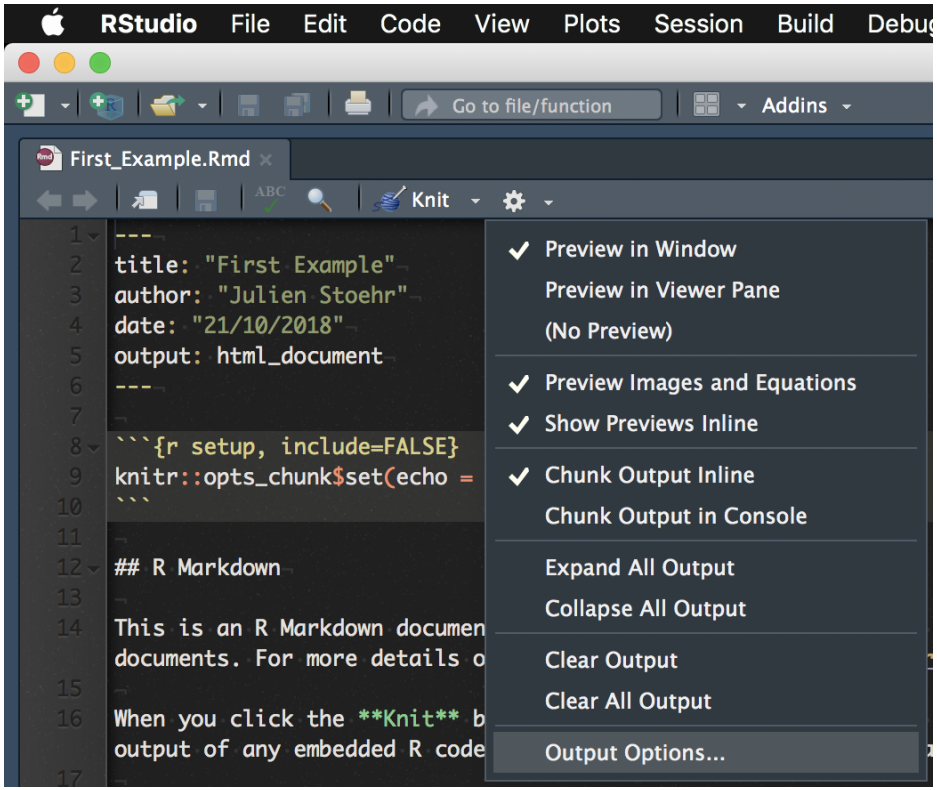
Configuration and compilation of the document

Output options:



Configuration and compilation of the document

Compilation: the `Knit` button on RStudio or the function `render` from the `rmarkdown` package at command line.



Text:

```
# My title
## My subtitle

*some text in italic*
**some text in bold**
```

Tables:

```
First Header | Second Header
- - - - - - - - - - | - - - - - - - - - - - -
Content Cell | Content Cell
Content Cell | Content Cell
```

Images:

```
![image_name](image_address)
```

Links:

```
<my_url>
[link_name](my_link)
```

Equations: the same syntax than in LaTeX

Chunk of R code:

````
```{r}
x <- 3 + 2
```
````

Inline R code:

````
The value of x is `r x`.
````

Chunk of R code:

````
```{r}
x <- 3 + 2
```
````

Inline R code:

````
The value of x is `r x`.
````

You are (almost) ready to use R Markdown!

Customising R code chunks

Local chunk options:

````
```{r chunk_name, chunk_options}
my_code
```
````

Chunk options:

- `echo (default=TRUE)`
- `eval (default=TRUE)`
- `tidy (default=FALSE)`
- `message (default=TRUE)`
- `include (default=TRUE)`
- `warning (default=TRUE)`

```
test <- function(n){message("testing chunk options")
  warning("This warning is useless")
  for(k in 1:n){y=dnorm(0.5,k);print(y)}}
test(5)
```

Chunk options:

- echo (default=TRUE)
- eval (default=TRUE)
- tidy (default=FALSE)
- message (default=TRUE)
- include (default=TRUE)
- warning (default=TRUE)

```
test <- function(n){message("testing chunk options")
  warning("This warning is useless")
  for(k in 1:n){y=dnorm(0.5,k);print(y)}}
test(5)
```

Chunk options:

- `echo (default=TRUE)`: display code in the document
- `eval (default=TRUE)`: run the code in the chunk
- `tidy (default=FALSE)`: tidy the code displayed
- `message (default=TRUE)`: display code messages in the document
- `include (default=TRUE)`: include the chunk in the document
- `warning (default=TRUE)`: display code warnings in the document

- Option `tidy = FALSE`

```
for(k in 1:10){y=dnorm(x,k);print(y)}
```

- Option `tidy = TRUE`

```
for (k in 1:10) {
    y <- dnorm(x, k)
    print(y)
}
```

- Option `tidy = FALSE`

```
for(k in 1:10){y=dnorm(x,k);print(y)}
```

- Option `tidy = TRUE`

```
for (k in 1:10) {
    y <- dnorm(x, k)
    print(y)
}
```

Remark. The version 1.20 of knitr uses `formatR` to reformat the R code. The version 1.21 (unreleased) will also support `styler` to reformat code.

About graphics:

- `fig.align ('left', 'right', or 'center')`: figure alignement
- `fig.cap (default=NULL)`: figure caption as character string
- `fig.height, fig.width`: dimensions of plots in inches

Exercise

- Create a `RMarkdown` file which reproduce the example from `Mixture_model_1.html`
- The margin of the histogram should be: bottom = 0.4", left = 0.8", top = 0.5", right = 0.1"

Customising R code chunks

Global R options:

- Managing options used by R when it computes and display results
- knitr is a package which relies on some other packages (`formatR`, `highr`)

```
options(continue = "  ", width = 50, formatR.arrow = TRUE)
```

Global chunk options:

```
knitr::opts_chunk$set(prompt = FALSE, tidy = TRUE)
```

Customising R code chunks

Global R options:

- Managing options used by R when it computes and display results
- knitr is a package which relies on some other packages (`formatR`, `highr`)

```
options(continue = "  ", width = 50, formatR.arrow = TRUE)
```

Chunk hooks: functions to be called before or after a code chunk. You can use it to define custom margin for graphics.

```
knitr::knit_hooks$set(p.mar = function(before, options,
    envir) {
    if (before)
        par(mai = c(0.8, 0.5, 0.5, 0.1), family = "serif",
            cex.lab = 1.2, cex.axis = 0.9)
})
```

Exercise

- Fill in the previous `RMarkdown` file so you are reproducing the content of `Mixture_model_2.html`

Good practice: the package `viridis` allows to generate color scales which are distinguishable for readers with the most common form of color blindness. If you want to use solely the colormap, you can install the package `viridisLite` instead.

```
install.packages("viridis", dependencies = TRUE)
```

A brief overview but you can also do...

Transforming data frame: `kable` function

```
knitr::kable(faithful[1:2, ], format = "latex")
```

eruptions	waiting
3.6	79
1.8	54

Presentation: you can create presentations with a wide range of options

- `beamer` (PDF)
- `slidy` (HTML)
- `ioslides` (HTML)

Interactive documents (more advanced): interactive graphics with `ggvis`, shiny document (HTML, RStudio ou Shiny server))

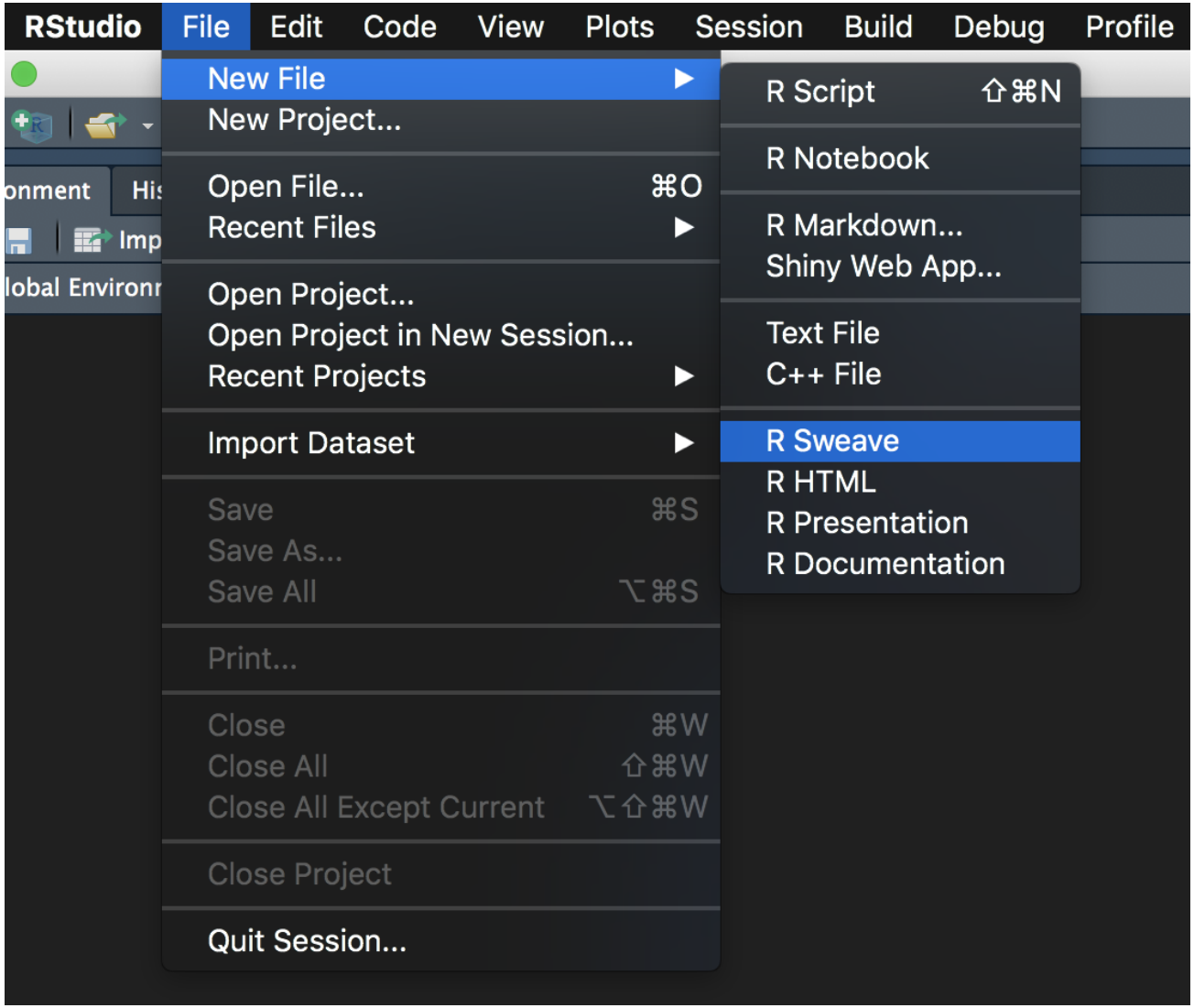
To sum up

With R Markdown

- Forget the old fashion way: saving R output + including output in a report.
- Reproducible report containing code and outputs: you can easily change the data, the code and regenerate the report
- Syntax easy to use

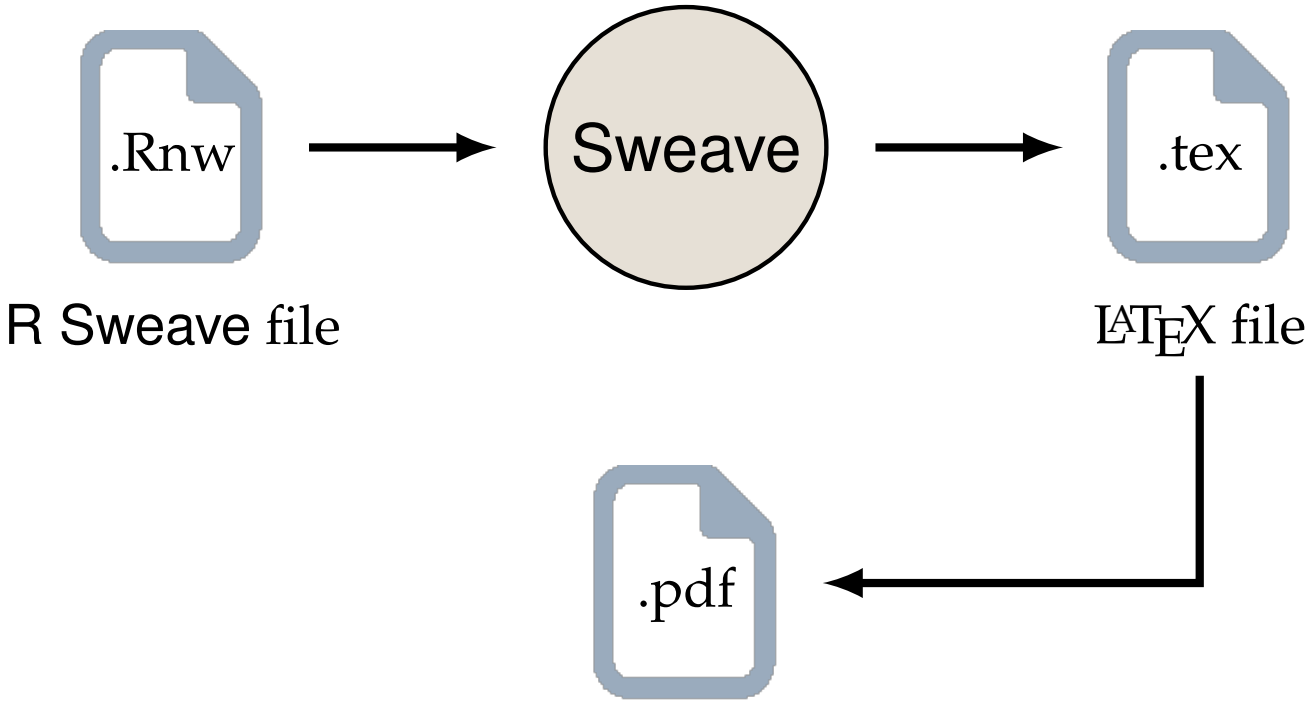
Some limitations: the editor is not as powerful as LaTeX $\rightsquigarrow$ it can be a drawback to produce a report containing a lot of math.

Going further with Sweave and knitr



First solution by Friedrich Leisch

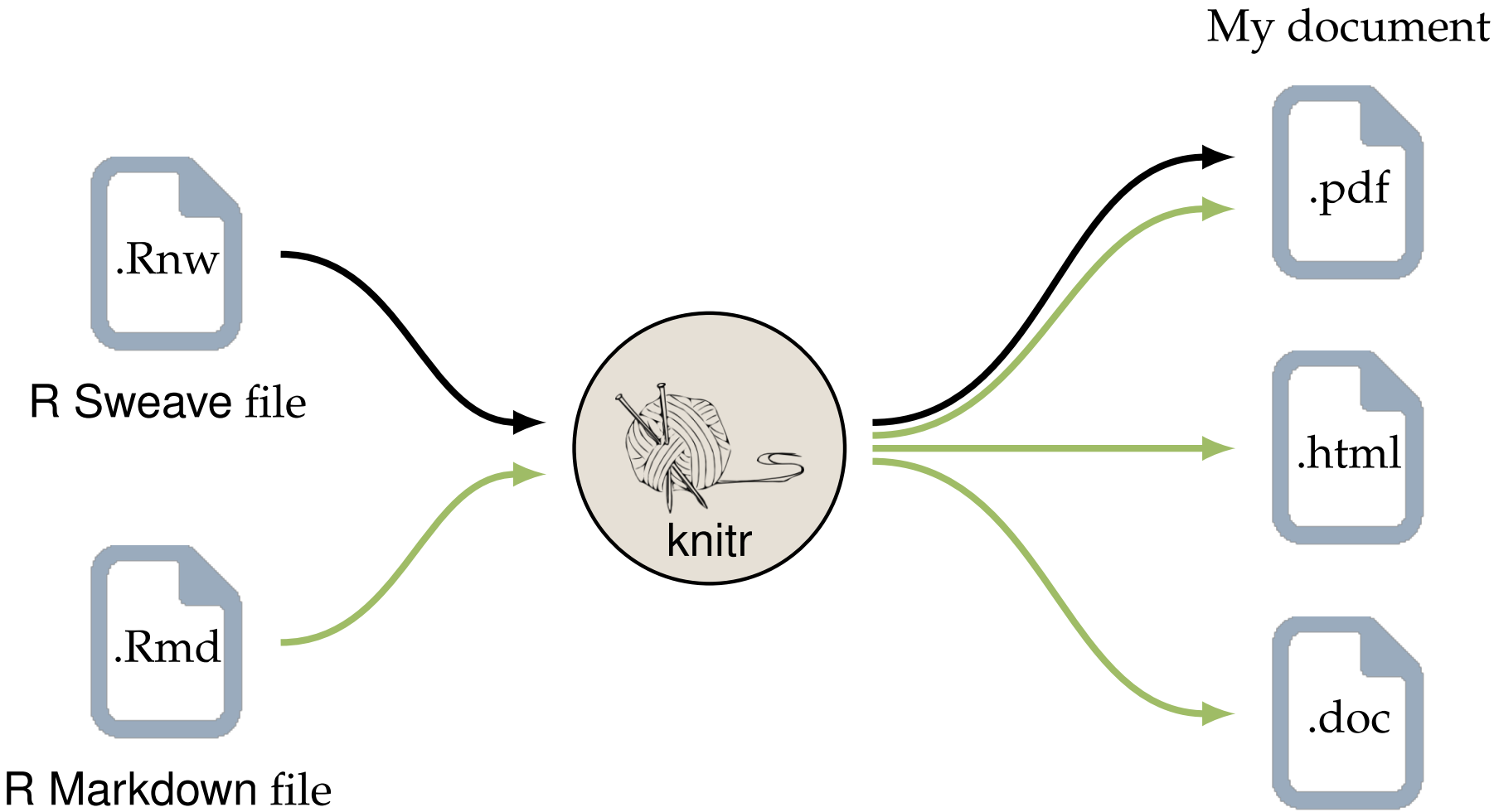
Sweave: gathering in a single document LaTeX code and R code



- **Advantages:** a nice and flexible layout
- **Drawback:** a good knowledge of LaTeX and `HTML` is required

Sweave and knitr

knitr: useful to include R code using a syntax almost similar to R Markdown



How to proceed?

1. Create a `.Rnw` file (same structure than a LaTeX file)
2. Define global options and hooks in the header
3. R code chunk are included using tags

```
<<chunk_name, chunk_options>>=
my_code
@
```

4. Inline R code can be obtained with the command `\Sexpr`

Remark. When creating a `beamer`, the option `fragile` need to be used when a frame contains R code chunks.

Some useful references

- **R Markdown website:**
  `https://rmarkdown.rstudio.com`
- **Reference Guide:**
  `https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf`
- **Cheatsheets (also available in the Help Tab of RStudio):**
  `https://rmarkdown.rstudio.com/lesson-15.html`
- **knitr website:**
  `https://yihui.name/knitr/`
- **knitr book:** Xie, Yihui (2016). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC.