UiO : PROMENTA
Det samfunnsvitenskapelige fakultet

Introduction to R Markdown

E. F. Haghish

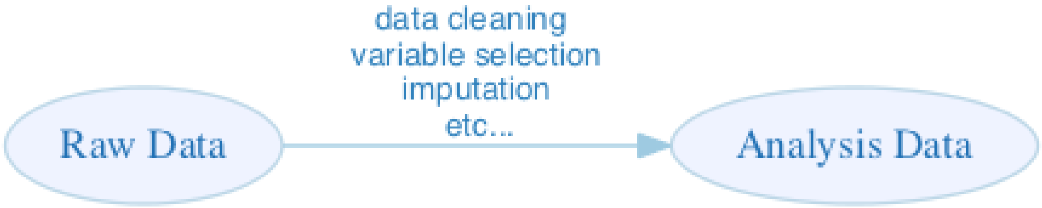
25/8/2022

Aim

- Why should you work with R Markdown
- R markdown syntax (basics)
- Including figures and tables
- Writing a reproducible document

Main idea

- write your code and document it in a single file
- use the same file for generating analysis reports and communicating your report
- make the process of cleaning the raw data reproducible



Transparent research practice

Let's consider a few scenarios:

1. Joe fabricates data! and writes an article with his exciting imaginary results.
2. Jasmin collects some data from an online questionnaire. She does not have a very large data, but after dropping a few subjects that did not repond as expected, her analysis model seems to be statistically significant and she has all that she needs to publish that study
3. Jack is trying to model his data, however he applies inappropriate methods. Nevertheless, he gets significant results and he publishes his analysis.
4. Johnathan has caried out a careful study, cleaned the data, and carried out the analysis. He is in the process of writing his journal article. However, when he writes the results of the analysis in the article, he types wrong numbers in the tables. He also includes a figure in the article that was created on the raw data and does not presents the analysis data set.

Transparent research practice

Errors can happen in any step of the study. As a statistician, at least, you want to be **as transparent as possible** in the process of:

- obtaining the data
- cleaning the raw data and selecting variables
- detecting outliers
- preparing the analysis data set
- modeling and executing the analysis
- reporting the analysis

In other words, from the moment we have access to digitalized data, we want to be able to reproduce **every single action that is carried out on the data** to allow others - and ourselves - to look back in what has been done. This process must be reproducible, from **A** to **Z.**

Installing required software

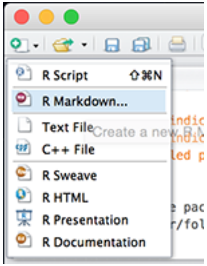
Installing Required packages for dynamic documentation

- Install RStudio
- Install `rmarkdown` and `knitr` R packages from CRAN

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

If you write with LaTeX instead of Markdown, then you also need to install LaTeX on your machine:
https://www.latex-project.org/get/

Opening a new Markdown file in R



Markdown

- Similar to LaTeX and HTML, Markdown is a language for annotating document
- Markdown has a simple syntax for writing headings in the document
- The simplicity of the syntax **improves the readability of the code**, which is a big advantage
- It is a very popular language for writing in Online forums
- It only covers the most basic for editing documents

UiO : PROMENTA

Markdown

```
Heading 1
=========

Heading 2
---------

# Heading 1

## Heading 2

### heading 3

#### Heading 4

##### Heading 5
```

The **rmarkdown** package (Allaire, Xie, McPherson, et al. 2022) was first created in early 2014. During the past four years, it has steadily evolved into a relatively complete ecosystem for authoring documents, so it is a good time for us to provide a definitive guide to this ecosystem now. At this point, there are a large number of tasks that you could do with R Markdown:

- Compile a single R Markdown document to a report in different formats, such as PDF, HTML, or Word.
- Create notebooks in which you can directly run code chunks interactively.
- Make slides for presentations (HTML5, LaTeX Beamer, or PowerPoint).
- Produce dashboards with flexible, interactive, and attractive layouts.
- Build interactive applications based on Shiny.
- Write journal articles.
- Author books of multiple chapters.
- Generate websites and blogs.

2.3 Cheat sheets

RStudio has created a large number of cheat sheets, including the one-page R Markdown cheat sheet, which are freely available at https://www.rstudio.com/resources/cheatsheets/. There is also a more detailed R Markdown reference guide. Both documents can be used as quick references after you become more familiar with R Markdown.

2.4 Output formats

There are two types of output formats in the **rmarkdown** package: documents, and presentations. All available formats are listed below:

- `beamer_presentation`
- `context_document`
- `github_document`
- `html_document`
- `ioslides_presentation`
- `latex_document`
- `md_document`
- `odt_document`
- `pdf_document`
- `powerpoint_presentation`
- `rtf_document`
- `slidy_presentation`
- `word_document`

2.5 Markdown syntax

The text in an R Markdown document is written with the Markdown syntax. Precisely speaking, it is Pandoc's Markdown. There are many flavors of Markdown invented by different people, and Pandoc's flavor is the most comprehensive one to our knowledge. You can find the full documentation of Pandoc's Markdown at https://pandoc.org/MANUAL.html. We strongly recommend that you read this page at least once to know all the possibilities with Pandoc's Markdown, even if you will not use all of them. This section is adapted from Section 2.1 of Xie (2016), and only covers a small subset of Pandoc's Markdown syntax.

2.5.1 Inline formatting

Inline text will be *italic* if surrounded by underscores or asterisks, e.g., `_text_` or `*text*`. **Bold** text is produced using a pair of double asterisks (`**text**`). A pair of tildes (`~`) turn text to a subscript (e.g., `H~3~PO~4~` renders H_3PO_4). A pair of carets (`^`) produce a superscript (e.g., `Cu^2+^` renders Cu^{2+}).

To mark text as `inline code`, use a pair of backticks, e.g., `` `code` ``. To include n literal backticks, use at least $n+1$ backticks outside, e.g., you can use four backticks to preserve three backtick inside: ````` ```` ```code``` ```` `````, which is rendered as ```` ```code``` ````.

Hyperlinks are created using the syntax `[text](link)`, e.g., `[RStudio](https://www.rstudio.com)`. The syntax for images is similar: just add an exclamation mark, e.g., `![alt text or image title](path/to/image)`. Footnotes are put inside the square brackets after a caret `^[]`, e.g., `^[This is a footnote.]`.

Example

```
text can be **bold** or __bold__ and also *italic* or _italic_.
To refer to functions and comands, use `monospace` font.
```

results:

text can be **bold** or **bold** and also *italic* or *italic*. To refer to functions and comands, use `monospace` font.

Styling text

End a line with two spaces for line-break
otherwise continues lines will be interpreted as a single text paragraph.

2.5.2 Block-level elements

Section headers can be written after a number of pound signs, e.g.,

```
# First-level header

## Second-level header

### Third-level header
```

If you do not want a certain heading to be numbered, you can add `{-}` or `{.unnumbered}` after the heading, e.g.,

```
# Preface {-}
```

Unordered list items start with `*`, `-`, or `+`, and you can nest one list within another list by indenting the sub-list, e.g.,

```
- one item
- one item
- one item
    - one more item
    - one more item
    - one more item
```

The output is:

- one item
- one item
- one item
    - one more item
    - one more item
    - one more item

2.6 R code chunks and inline R code

You can insert an R code chunk either using the RStudio toolbar (the `Insert` button) or the keyboard shortcut `Ctrl + Alt + I` (`Cmd + Option + I` on macOS).

There are a lot of things you can do in a code chunk: you can produce text output, tables, or graphics. You have fine control over all these output via chunk options, which can be provided inside the curly braces (between ` ```{r ` and `}` ). For example, you can choose hide text output via the chunk option `results = 'hide'` , or set the figure height to 4 inches via `fig.height = 4` . Chunk options are separated by commas, e.g.,

~~~
```{r, chunk-label, results='hide', fig.height=4}
~~~

Dynamic text

```
a = 10
```

```
The avlue of __a__ equals 'r a' !
```

The avlue of **a** equals 10 !

Decorate text

```
[hypertext](http://www.sdu.dk)
```

You can also add a link or inline equation: $A = \pi * r^2$ uaing LaTeX notation

$$A = \pi * r^2$$

Making comments

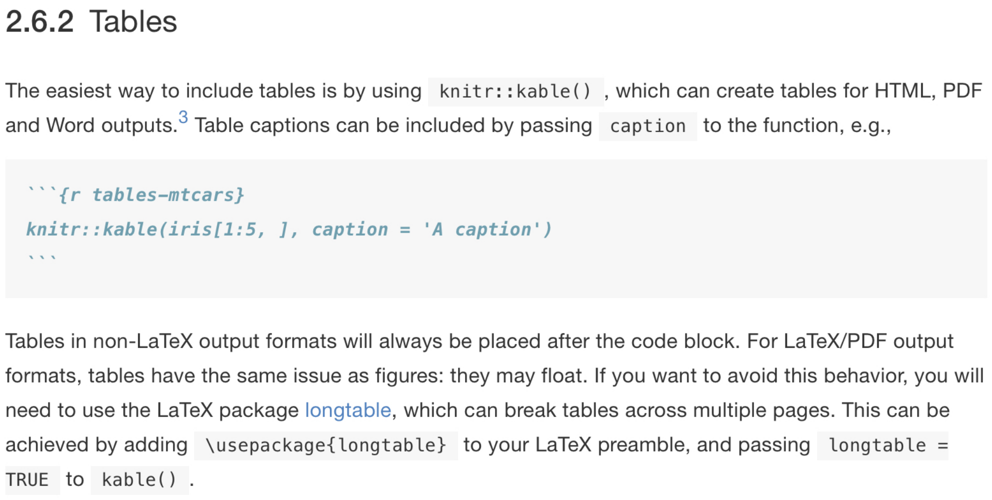
The following text will not be rendered in the document because it is included within HTML comment signs <-- and -->.

```
<!--
When you click the **Knit** button a document will
be generated that includes both content as well as
the output of any embedded R code chunks within the
document:
-->
```

UiO : F

There are a large number of chunk options in **knitr** documented at https://yihui.name/knitr/options. We list a subset of them below:

- `eval` : Whether to evaluate a code chunk.
- `echo` : Whether to echo the source code in the output document (someone may not prefer reading your smart source code but only results).
- `results` : When set to `'hide'` , text output will be hidden; when set to `'asis'` , text output is written "as-is," e.g., you can write out raw Markdown text from R code (like `cat('**Markdown** is cool.\n')`). By default, text output will be wrapped in verbatim elements (typically plain code blocks).
- `collapse` : Whether to merge text output and source code into a single code block in the output. This is mostly cosmetic: `collapse = TRUE` makes the output more compact, since the R source code and its text output are displayed in a single output block. The default `collapse = FALSE` means R expressions and their text output are separated into different blocks.
- `warning` , `message` , and `error` : Whether to show warnings, messages, and errors in the output document. Note that if you set `error = FALSE` , `rmarkdown::render()` will halt on error in a code chunk, and the error will be displayed in the R console. Similarly, when `warning = FALSE` or `message = FALSE` , these messages will be shown in the R console.



2.6.2 Tables

The easiest way to include tables is by using `knitr::kable()`, which can create tables for HTML, PDF and Word outputs.[3] Table captions can be included by passing `caption` to the function, e.g.,

~~~
```{r tables-mtcars}
knitr::kable(iris[1:5, ], caption = 'A caption')
```
~~~

Tables in non-LaTeX output formats will always be placed after the code block. For LaTeX/PDF output formats, tables have the same issue as figures: they may float. If you want to avoid this behavior, you will need to use the LaTeX package longtable, which can break tables across multiple pages. This can be achieved by adding `\usepackage{longtable}` to your LaTeX preamble, and passing `longtable = TRUE` to `kable()`.

Including Plots ==================

Plots are included automatically. You can also embed plots by importing it in the

Including an image

```
![](path/to/filename)
```

Table

```
Table Header  | Second Header
------------- | -------------
Table Cell    | Cell 2
Cell 3        | Cell 4
```

Table Header	Second Header
Table Cell	Cell 2
Cell 3	Cell 4

Table

- a better way to include a Markdown table is to generate it automatically from a matrix, to make the table dynamic.
- Will be covered later during the course

Mathematical notations

We often report some greek signs in statistical analysis. Luckily, all `LaTeX` math symbols are supported in RMarkdown. If you do not know how to write mathematical equations or values in LaTeX, use online platforms (e.g. **latex.codecogs.com**) for practicing

- https://latex.codecogs.com/eqneditor/editor.php

Mathematical text can be inline `$\beta$` that returns β or on a seperate line `$$\beta$$` that will return:

$$\beta$$

Here is another example: `$A = \pi*r^{2}$` returns $A = \pi * r^2$ and with double dollar sign, it returns:

$$A = \pi * r^2$$

Summary

- what is R Markdown about?
- why should we use it?
- how is it different from R script?
- what can you produce with it?
- does it help you learn R?

TASK

- Download R Markdown cheat sheet from module 1 on canvas
- Create an R Markdown file and type in the syntax.
- Try the syntax of R Markdown as shown in the cheat sheet.
- create a histogram in the R Markdown file from `cars` dataset, e.g. `cars$speed` variable.
  - search for help using `help(hist)` command
  - make the figure to appear like a square in the file by giving it equal width and height *(difficult)*
  - render the R Markdown document in Microsoft Word format

TASK

- go to www.tablegenerator.com/markdown.tables
- create a simple table with some random rows and columns
- Make the first line (column names) **bold**
- make the first column *italic*
- copy-paste the markdown code to R Markdown file and rerender it