

Reproducible research workflows for psychologists

Introduction to RMarkdown

Johannes Breuer & Frederik Aust

KU Leuven, 27.-28.04.2022

Dynamic documents

Dynamic documents are derived from the concept of [literate programming](#). They fuse computer code and documentation and results are embedded directly into the document.

Dynamic documents

Dynamic documents can be a partial solution to the challenge of computational reproducibility (same data, same code, same results). They can prevent transcription errors and ensure that statistics, tables, and figures represent the current analytic approach.

One solution for producing dynamic documents is R
Markdown .

What is R Markdown?

R Markdown provides an unified authoring framework for data science, combining your code, its results, and your prose commentary. R Markdown documents are fully reproducible and support dozens of output formats, like PDFs, Word files, slideshows, and more ([R for Data Science](#)).

What is R Markdown?

R Markdown is...

- an authoring framework
- a document format (`.Rmd`)
- an R package

What is R Markdown?

Markdown + R

TL;DR of the *Wikipedia* article: [Markdown](#) is a lightweight markup language for text formatting.

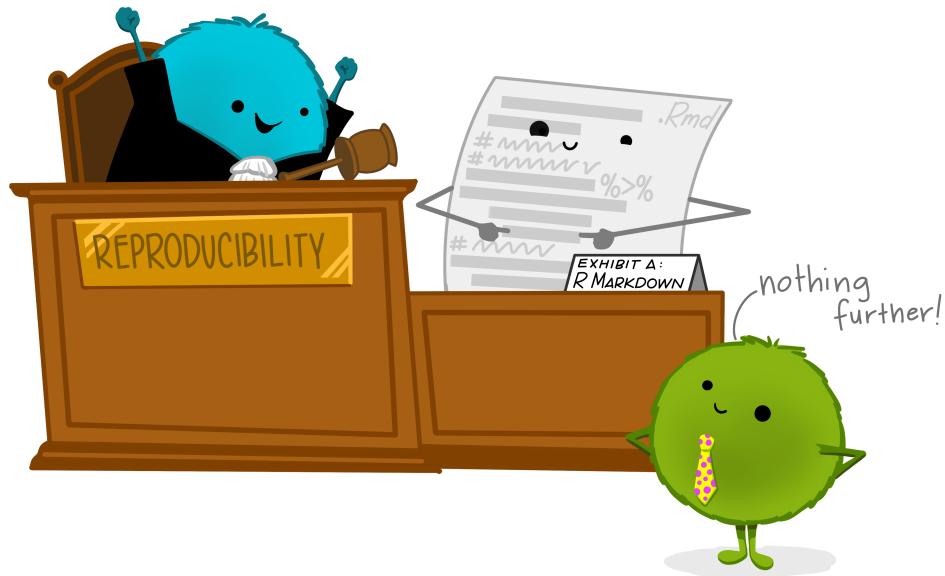
What does R Markdown do?



Artwork by @allison_horst

R Markdown and reproducibility

As it combines code, text, and outputs, R Markdown is a great tool for writing reproducible publications (papers, project reports, etc.).



What can you do with R

Markdown?

In a nutshell, with R Markdown it is possible to generate reproducible dynamic documents which...

- (can) include text, code, and output from that code
- render to many different output formats, including:
 - HTML
 - Markdown
 - PDF
 - Microsoft Word
 - Open Document
 - RTF

What can you do with R Markdown?

There are quite a few packages that offer extension output formats for R Markdown . For example:

- `xaringan` for presentations---which is what we use for our slides
- `bookdown` for books (but also for websites)
- `blogdown` for websites
- `vitae` for (data-based) Résumés and CVs
- `posterdown` for academic (conference) posters
- `flexdashboard` for interactive dashboards

... and there are many more.

R Markdown and versatility

Disclaimer: What we will cover

Covering everything you can do with `R Markdown` or even exploring all options for specific kinds of outputs, such as presentations or scientific publications, in-depth would be enough for separate workshops. Hence, this session will only cover the basics of `R Markdown`.

In the next session, we will discuss reference management with `R Markdown` and dive into the possibilities offered by the package `papaja` which can be used to prepare APA Journal Articles with `R Markdown`.

Getting started with R Markdown

If you use *RStudio* you only need to install the **R**
Markdown package:

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

Note: If you do not have *RStudio* installed, you also need to **install Pandoc**.

PDF output with R Markdown

If you want to generate PDF output with R Markdown, you need *L^AT_EX*. If you have a *L^AT_EX* distribution like MiK^TeX or TeX Live on your system, you should be all set.

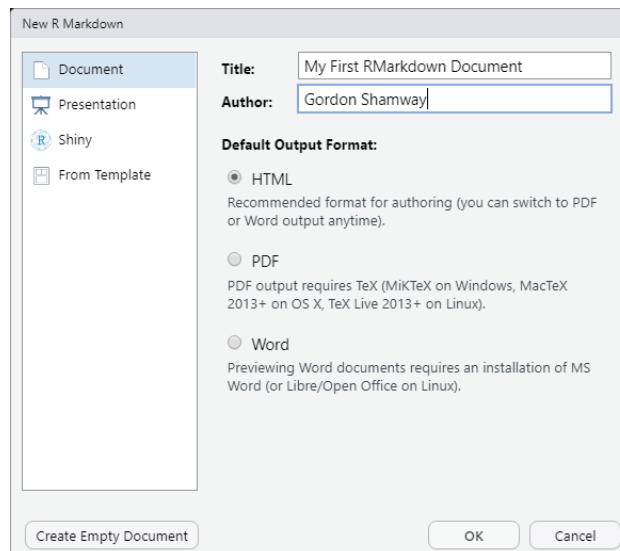
PDF output with R Markdown

If you do not have *LATEX* installed, the easiest option---especially if you do not want to use plain *LATEX*---is installing [TinyTeX](#), which is "a lightweight, cross-platform, portable, and easy-to-maintain LaTeX distribution based on TeX Live". You can do that using the [tinytex package](#). It also takes care of installing missing *LATEX* packages on the fly when knitting documents.

```
install.packages('tinytex')  
tinytex::install_tinytex()
```

Getting started with R Markdown

You can create a new R Markdown document in *RStudio* via *File -> New File -> R Markdown* in the menu. This will open a new window in which you can set the author name and title and pick an output format for your document.



Ingredients



R Markdown documents are two-part plain text documents

1. YAML front matter

- Document metadata
- Rendering options

2. Document body

- Markdown text
- R code

Anatomy of an R Markdown document

The screenshot shows an RStudio interface with an R Markdown file named "my_first_markdown.Rmd". The code editor displays the following content:

```
1 ---  
2 title: "My First RMarkdown Document"  
3 author: "Gordon Shumway"  
4 date: "30 7 2020"  
5 output: html_document  
6 ---  
7  
8 ```{r setup, include=FALSE}      setup chunk  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ```  
11  
12 ## R Markdown  Markdown text  
13  
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.  
15  
16 When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:  
17  
18 ```{r cars}    code chunk  
19 summary(cars)  
20 ```  
21  
22 ## Including Plots  
23  
24 You can also embed plots, for example:  
25  
26 ```{r pressure, echo=FALSE} ← code chunk label + options  
27 plot(pressure)  
28  
29  
30 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.  
31  
32 ## Inline Code  
33  
34 You can also include 'R' code (that will be executed) in your text: The mean temperature of the observations in the *pressure* dataset is 'r mean(pressure$temperature)'.
```

Annotations with red arrows and text labels:

- A red arrow points to the "Knit" button in the toolbar, labeled "interactive table of contents".
- A red arrow points to the "code chunk label + options" in line 26, labeled "setup chunk".
- A red arrow points to the inline code in line 34, labeled "inline code".

At the bottom of the RStudio interface, there are status bars: "34:170", "Inline Code", "Interactive table of contents including code chunks", and "R Markdown".

YAML header

```
---
```

```
title: "My First R Markdown Document"
subtitle: "A first in the series of many more to come"
author: "Gordon Shamway"
date: "27-04-2022"
output: html_document
---
```

YAML stands for "YAML Ain't Markup Language" (formerly known as "Yet Another Markup Language"). The YAML header in R Markdown documents contains metadata for the document. It provides human-readable configuration information and can include a large variety of key:values-pairs to specify what the document should look like. It needs to be at the beginning of the document and start and end with `--`.

YAML header

YAML data structures translate to **list** objects in **R**

```
---
```

```
title: "My First R Markdown"
author: "Gordon Shamway"
date: "27 04 2022"
output: html_document
```

```
---
```

```
## List of 4
## $ title : chr "My First R Ma
## $ author: chr "Gordon Shamwa
## $ date  : chr "27 04 2022"
## $ output: chr "html_document"
```

YAML header

Indentations denote the nesting structure of a **list**

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

```
## List of 1  
## $ output:List of 1  
##   ..$ html_document:List of 1  
##     ...$ toc: logi TRUE
```

toc is nested in **html_document** (i.e., it is an argument to that output function)

YAML header

Similarly, it is possible to specify vectors.

```
bibliography: ["references.bib", "r-pkg-references.bib"]
```

```
## List of 1  
## $ bibliography: chr [1:2] "references.bib" "r-pkg-references.bib"
```

Note: We will cover how to manage references in **R**
Markdown in the next session.

YAML header

Text that spans multiple lines can be declared with |

```
abstract: |
  This text spans multiple rows.
```

New lines are preserved, but note the indentation!

Note: All lines must be indented!

YAML header

You can also use the `YAML` front matter to customize the appearance of the resulting documents. For example, you can specify that you want a table of contents (TOC), how many levels that should have, or whether sections should be numbered.

YAML header & *LATEX*

If you want to use `R Markdown` to generate PDF output via *LATEX*, you can make use of additional options in the `YAML` header, e.g., for loading additional *LATEX* packages or specifying a different *LATEX* engine (by default `pdflatex` is used) or keeping the `.tex` file.

(R) Markdown text formatting

While it is not necessary to know `Markdown` to use `R` `Markdown` (though if you want to know more, you can, e.g., check out the `Markdown Guide` or this `interactive tutorial`), it helps to know some of the basics of `Markdown` text formatting as they are the same for `R` `Markdown`.

Text formatting

Syntax

italics

****bold****

*****bold & italics*****

~~strikethrough~~

Output

italics

bold

bold & italics

~~strikethrough~~

Headers

Syntax

```
# Header 1  
## Header 2  
### Header 3
```

Output

Header 1

Header 2

Header 3

Paragraphs

A new paragraph is started with a blank line before the text.

NB: If you just hit Enter/Return to move text to a new line in an `R Markdown` document, the text you enter after that will not be on a new line in the output document.

Note: When you generate `HTML` output, you can also use `HTML` commands in your `R Markdown` document. So, for example, you could insert an empty line with `
`. Likewise, when producing PDF output, you can use `LATEX` commands, such as `\newline` or `\newpage`.

Lists

Syntax

- unordered list
 - + sub-item
- 1. ordered list
- 2. ordered list
 - + sub-item
 - + sub-item

Output

- unordered list
 - sub-item
- 1. ordered list
- 2. ordered list
 - sub-item
 - sub-item

Other formatting stuff

Syntax

```
`library(tidyverse)`
```

```
[link](https://gesis.org)
```

```
> block quote
```

```
! [R Logo]("./img/Rlogo.png")
```

Output

```
library(tidyverse)
```

link

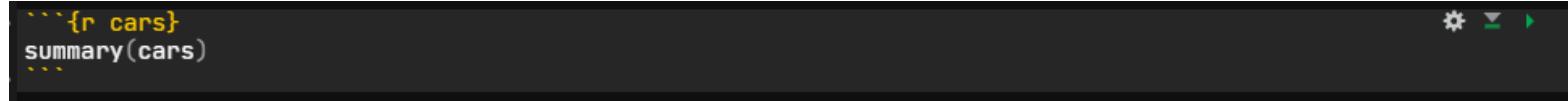
| block quote



Other formatting stuff

For more formatting options check out the [RMarkdown Reference Guide](#) which is also available in *RStudio* via *Help -> Cheatsheets -> R Markdown Reference Guide*.

Code chunks



A screenshot of a code editor window showing an R Markdown code chunk. The code is as follows:

```
```{r cars}
summary(cars)
````
```

The code is highlighted in yellow, indicating it is a code chunk. The editor interface includes a toolbar with icons for settings, file operations, and navigation.

As the name says, code chunks in [R Markdown](#) documents include code. This is typically [R](#) code, but other languages are supported as well (e.g., [Python](#) or [SQL](#)). The code is executed when the file is knitted (we'll talk about what this means in a bit).

Code chunks

You can insert a code chunk via the `Insert` button (select `R`) or using the keyboard shortcut `Ctrl + Alt + I` (*Windows & Linux*)/ `Cmd + Option + I` (*Mac*).

Note: It is possible to render an `R` script into an `R` Markdown report using `knitr::spin` and, vice versa, to convert an `R` Markdown document to an `R` script via `knitr::purl()`.

Code chunks

It is good practice to name code chunks. In the example on the previous slide `{r cars}` specifies the language for the code `r` and a name `cars`. By naming code chunks it is, e.g., possible to reference them in other code chunks and they will also appear in the interactive ToC at the bottom of the tab for the `R Markdown` document.

Chunk names may never be used twice in a single document and should not include spaces or underscores.

Chunk options

```
```{r pressure, echo=FALSE}
plot(pressure)
````
```

You can also set a variety of options for code chunks. In the above example, we set `echo = FALSE` which means that the code itself will not be displayed in the output document (only its output). Other exemplary chunk options are `eval = FALSE`, meaning that the code is not executed, or `warning = FALSE` or `message = FALSE` which mean that warnings or messages produced by the code are not shown in the output document. Yihui Xie, the main author of the `knitr` package, keeps an updated list of all code chunk options.

Setup chunk

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

It generally makes sense to include a setup chunk in your document (right after the YAML header). Here you can set global options for your code chunks (which can be overridden by setting options for individual chunks), general options for , or already load packages.

Inline code

```
You can also include `R` code (that will be executed) in your text: The mean temperature of the observations in the *pressure* dataset is `r mean(pressure$temperature)`.
```

It is also possible to execute code within text. That way, the output is automatically updated if it is compiled again after the input (usually the data) has changed. Inline code needs to be enclosed in backticks and has to start with a specification of the language (typically `r`) if the code should be executed when the document is compiled. Only the result(s) of the inline code (not the code itself) will be displayed in the output document.

Comments

It is also possible to include comments in an **R Markdown** document that will not be displayed in the output.

To comment something out, you can select it and use the keyboard shortcut **Ctrl + Shift + C** (*Windows & Linux*)/
Cmd + Shift + C (*Mac*).

A comment in **R Markdown** looks like this: **<!-- This is
a comment -->**

Excursus: Tables in R Markdown

As with many things in R, there are many options for creating tables that can be used with R Markdown. Discussing all of them would be too much for this workshop (but we will see some further examples in the session on `papaja`). An easy-to-use and quite versatile option is `knitr::kable()` which can be nicely extended using the `kableExtra` package.

Excursus: Tables in R Markdown

```
library(dplyr) # for wrangling
library(gapminder) # for exemplary data
library(kableExtra) # for table formatting

gapminder |>
  filter(year == 2007,
        continent == "Europe") |>
  select(country,
         lifeExp,
         pop,
         gdpPercap) |>
  head(10) |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "
```



Excursus: Tables in R Markdown

| country | lifeExp | pop | gdpPerCap |
|------------------------|---------|----------|-----------|
| Albania | 76.423 | 3600523 | 5937.030 |
| Austria | 79.829 | 8199783 | 36126.493 |
| Belgium | 79.441 | 10392226 | 33692.605 |
| Bosnia and Herzegovina | 74.852 | 4552198 | 7446.299 |
| Bulgaria | 73.005 | 7322858 | 10680.793 |
| Croatia | 75.748 | 4493312 | 14619.223 |
| Czech Republic | 76.486 | 10228744 | 22833.309 |
| Denmark | 78.332 | 5468120 | 35278.419 |
| Finland | 79.313 | 5238460 | 33207.084 |
| France | 80.657 | 61083916 | 30470.017 |

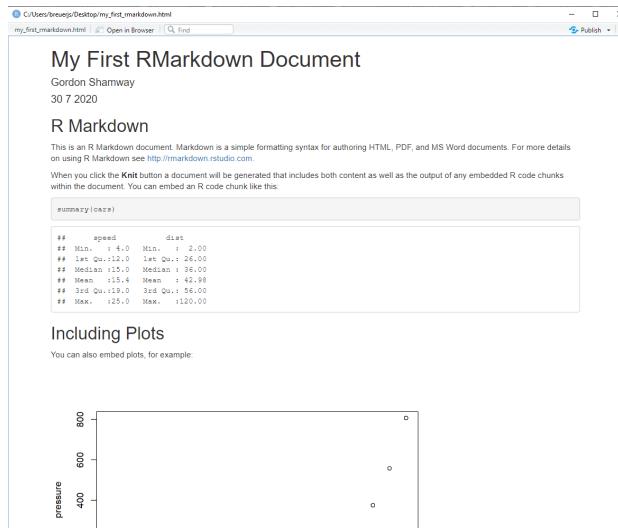
Excursus: Tables in R Markdown

Note: If you want to reference tables (or figures) in R Markdown documents, you need to use the bookdown output format `html_document2` (or `pdf_document2`).

Knitting



To compile the **R Markdown** document (in this case into a **HTML**) document, you simply need to click the **Knit** button. Doing this will generate the **HTML** file (by default) in the directory where the **.Rmd** file is stored. It will also open a preview window in *RStudio*.



Knitting



Instead of using the *Knit* button in the *RStudio* GUI you can also use the `render()` command from the `rmarkdown` package.

Knitting



Knitting an `R Markdown` file...

1. Starts a new `R` session
 - No variables defined
 - No packages loaded
2. Sets the working directory to the location of the `R Markdown` file
3. Executes all `R` code chunks from top to bottom
 - Variables are available in subsequent chunks

Note: For computationally intensive tasks, you can set the option `opts_chunk$set(cache = TRUE)`. It will cache chunk calls and their results as long as you do not edit them.

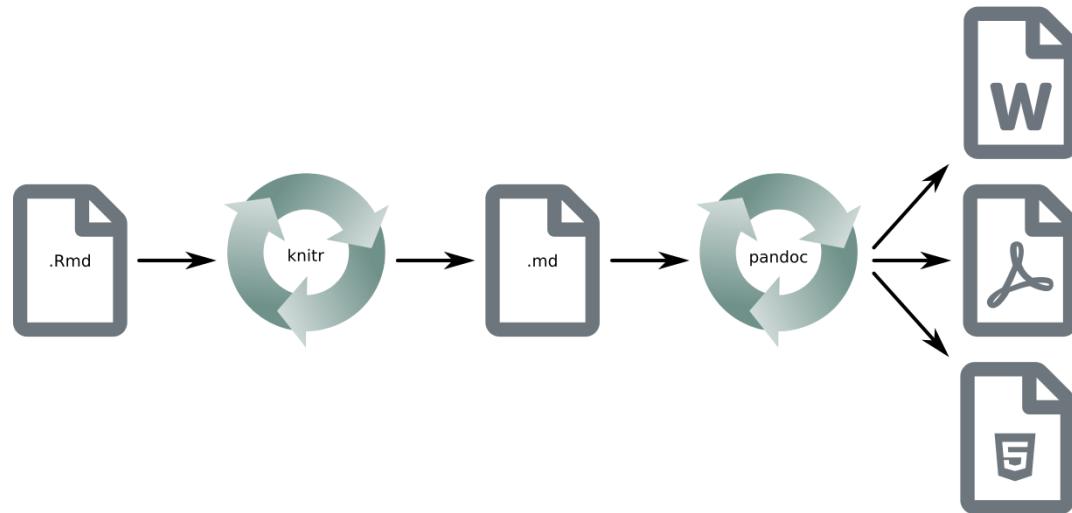
How R Markdown works



Artwork by @allison_horst

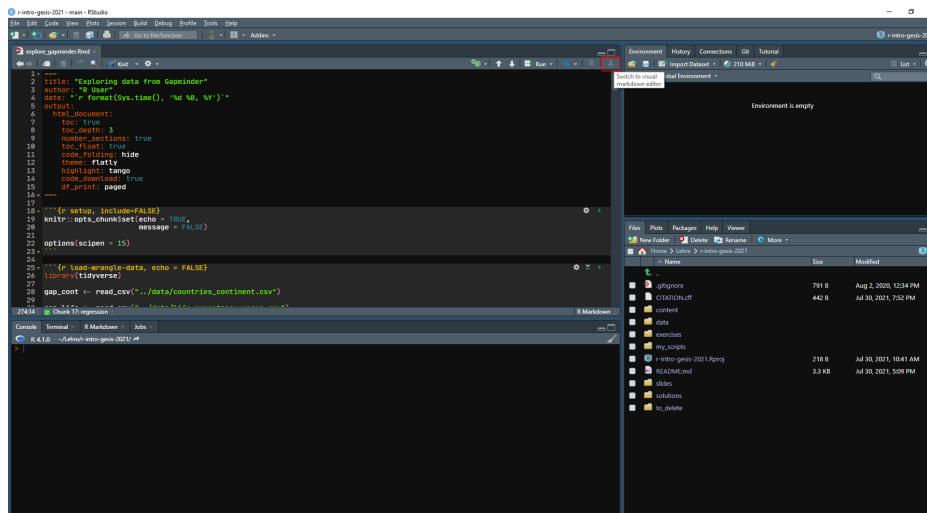
How R Markdown works

Behind the scenes, R Markdown uses `knitr` to execute the code and create a Markdown (`.md`) document with the code and output included, and `pandoc` to convert to a range of different output formats.



Visual R Markdown editor

If **WYSIWYG** is more your thing, you can rejoice as new(er) versions of *RStudio* (v. 1.4 or higher) now offer a **Visual R Markdown** editor. If you have an **.Rmd** document open in *RStudio*, you can open the visual editor via the GUI (in the **Source** pane).



Visual R Markdown editor

You can use the visual editor in *RStudio* for editing your R Markdown document similar to *Microsoft Word*.

The screenshot shows the RStudio interface with a dark theme. The left pane contains the R Markdown document:

```
explore_gapminder.Rmd*
```

The data

The data we are using comes from [Gapminder](#). We will use data on population, life expectancy, fertility rate, and GDP per capita. The full combined dataset contains data from `n_distinct(gap_full$country)` countries. As *Gapminder* collects the data from different sources and there are differences in the availability of different kinds of data, the individual datasets (for life expectancy, GDP per capita, etc) differ regarding the number of countries and years they cover. Notably, all of them except the one on GDP per capita also contain projected data until the year `max(gap_pop$year)`. The data on population, life expectancy, and fertility rate also all go back to `min(gap_pop$year)`. The data on GDP per capita, however, are only available for the years from `min(gap_gdp$year)` to `max(gap_gdp$year)`. Also, there is missing data for many countries for different years.

Missing data

As we do not have data for all variables for all countries and years, we want to know whether there are identifiable patterns of missingness in our data.

```
{r vis-miss}
library(visdat)
vis_miss(gap_full)
```

Data for 2018

Given that we only have data on GDP per capita from `min(gap_gdp$year)` to `max(gap_gdp$year)`, we will focus on data from the year `max(gap_gdp$year)` in our analysis. The dataset for `max(gap_gdp$year)` contains data for `nrow(gap_2018)` countries from `n_distinct(gap_2018$continent)` continents. Let's have a look at this dataset.

```
{r full-table}
gap_2018
```

Missing data in the 2018 dataset

The right pane shows the environment and file browser:

Environment: Environment is empty

Files: 1.1_Getting_Started.Rmd, 1.2_Data_Types_Import_Export.Rmd, 1.3_Appendix_Setup_Workflow_Help.Rmd, 2.1_Data_Wrangling_Basics.Rmd, 2.2_Data_Wrangling_Advanced.Rmd, 2.3_Appendix_Relational_Data.Rmd, 3.1_Exploratory_Data_Analysis.Rmd, 3.2_Data_Visualization_Part_1.Rmd, 4.1_Confirmatory_Data_Analysis.Rmd, 4.2_Data_Visualization_Part_2.Rmd, 5.1_R_Markdown.Rmd, 5.2_Outlook.Rmd, sessions_list.csv

Some best practices for R

Markdown

- Load all packages in the first code chunk
 - Never include `install.packages()`
- Use relative paths or load files from a permanent location
 - Do not use `setwd()`
- Use meaningful chunk names
- Keep R code close to the corresponding prose
- Set seeds for random number generators
(`set.seed()`)

Reproducibility information

To further increase the reproducibility of your `R` `Markdown` document you can include some information about your `R` (e.g., the OS, `R` version, and packages that you have used).

```
sessionInfo()
```

Reproducibility information

```
## R version 4.1.3 (2022-03-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252
## [3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.1252
##
## attached base packages:
## [1] tools      stats       graphics   grDevices  utils      datasets   methods
## [8] base
##
## other attached packages:
## [1] kableExtra_1.3.4      gapminder_0.3.0      dplyr_1.0.6
## [4] tweetermd_0.0.8       emo_0.0.0.9000     xaringantheme_0.3.4
## [7] xaringanExtra_0.4.0    knitr_1.37          fs_1.5.0
## [10] rmarkdown_2.11         rprojroot_2.0.2     tarchetypes_0.4.1
## [13] targets_0.10.0
##
## loaded via a namespace (and not attached):
## [1] Rcpp_1.0.7           svglite_2.0.0        lubridate_1.7.10    png_0.1-7
## [5] sysfonts_0.8.8       ps_1.6.0            assertthat_0.2.1    digest_0.6.27
## [9] utf8_1.2.1           mime_0.10          R6_2.5.0            backports_1.2.1
## [13] evaluate_0.14        httr_1.4.2          xaringan_0.21       highr_0.9
## [17] unilur_0.4.0.9000    pillar_1.6.1        rlang_0.4.11       curl_4.3.1
## [21] uuid_0.1-4           rstudioapi_0.13    data.table_1.14.0   whisker_0.4
## [25] callr_3.7.0          jquerylib_0.1.4    klippy_0.0.0.9500  webshot_0.5.2
## [29] stringr_1.4.0        munsell_0.5.0       igraph_1.2.6       hunspell_3.0.1
```

Exercise time 💪💪🏃‍♂️🚴‍♂️

Solutions

R Markdown resources

The *RStudio R Markdown Cheatsheet*

The *R Markdown materials by RStudio*

The *R Markdown chapter in R for Data Science* by Hadley Wickham

R Markdown: The Definitive Guide by Yihui Xie, J. J. Allaire, and Garrett Grolemund

R Markdown Cookbook by Yihui Xie, Christophe Dervieux, and Emily Riederer

R Markdown for Scientists by Nicholas Tierney

R Markdown Tips and Tricks by Indrajeet Patil

Outlook

R Markdown is a great tool (esp. for reproducibility) and will continue to be used and extended...

BUT... there is a potential (or likely?) successor in the wings: "Quarto is a multi-language, next generation version of R Markdown from RStudio, with many new features and capabilities"

Outlook: Quarto

- support for R, Python, Julia, and Observable
- can also be used with Jupyter notebooks
- even more output formats

For further details check out the Quarto documentation and this [blog post by Alison Hill](#).

Note: For R, Quarto uses R Markdown under the hood, so everything you learn here is fully compatible with Quarto.