gesis Leibniz Institute for the Social Sciences

Tools and Workflows for Reproducible Research in the Quantitative Social Sciences

Introduction to R Markdown

Johannes Breuer, Bernd Weiss, & Arnim Bleier

2022-11-17

Leibniz Association

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`

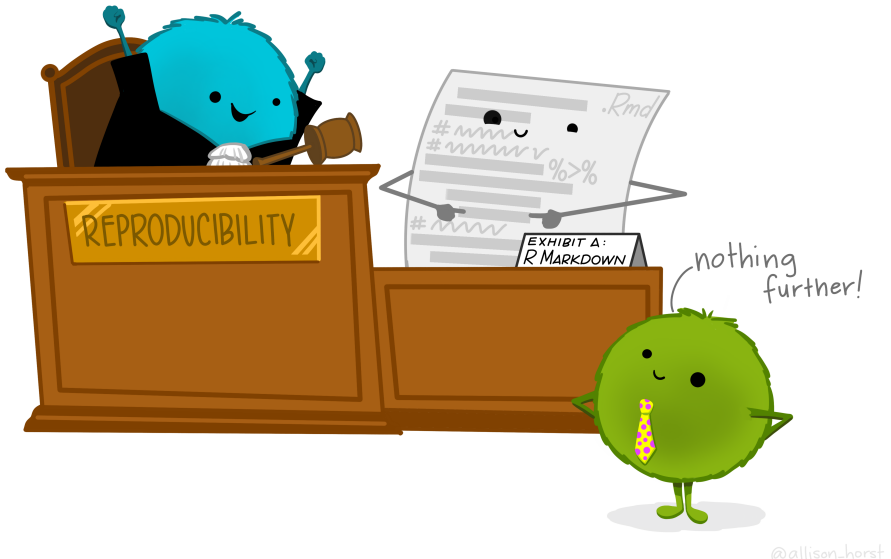
TL;DR of the *Wikipedia* article: `Markdown` is a lightweight markup language for text formatting.

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.).



Artwork by Allison Horst

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`

For a full list of supported output formats, see the `rmarkdown` package documentation.

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 these slides)
- `bookdown` for books (but also for websites)
- `blogdown` for websites
- `vitae` for (data-based) Résumés and CVs
- `posterdown` for academic (conference) posters
- `papaja` for APA-style manuscripts

... and there are many more.

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`.

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.

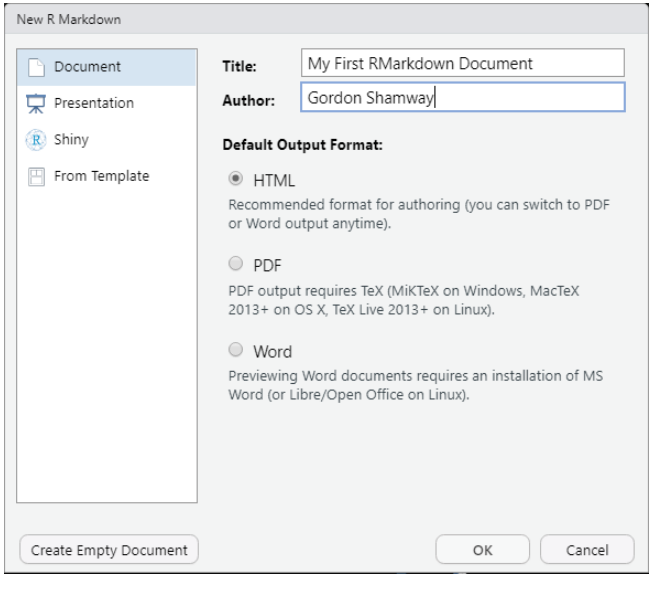
Output format

In this session, we will focus on generating `HTML` output with `R Markdown`.

If you want to generate PDF output with `R Markdown`, you need a LaTeX installation. 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.

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.

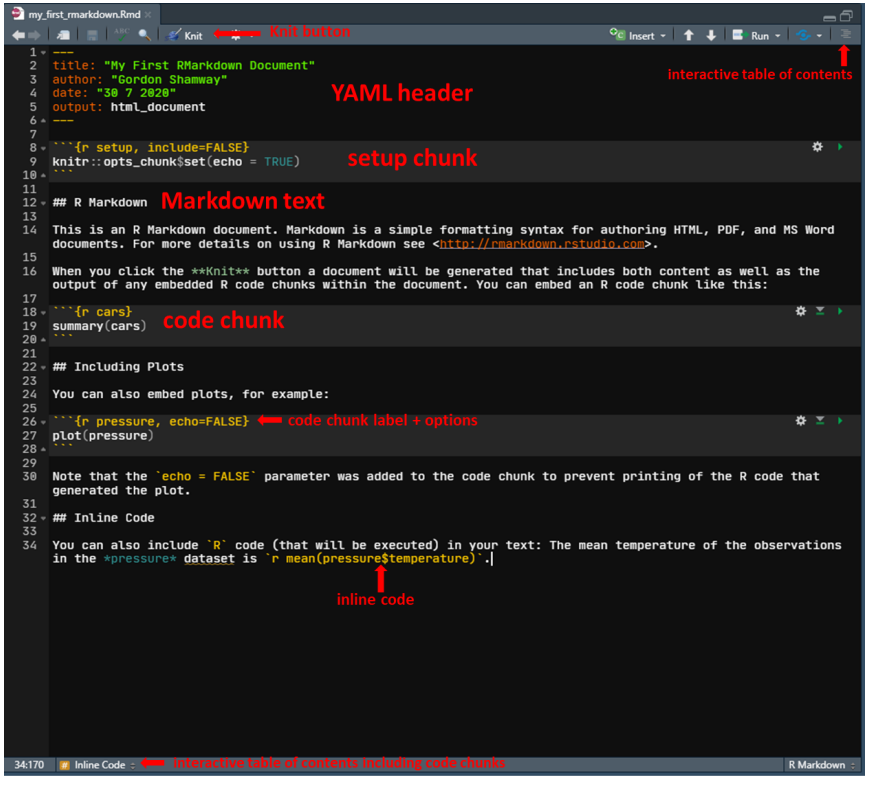


Ingredients 

`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



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 `---`.

Note: There is an `R` package called `ymlthis` for creating extended YAML headers in and with `R`.

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.

```
---
title: "My life with Chiroptophobia"
subtitle: "How fear can make us strong"
author: "Bruce Wayne"
date: "27-04-2022"
output: html_document:
    toc: true
    toc_depth: 2
    number_sections: true
---
```

(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.

Basic 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 `<br>`.

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

R Markdown formatting

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

Code chunks

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., `bash`, `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 extract an `R` script from an `R Markdown` document 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. For further options, you can check the 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 `R`, 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 -->`

Exemplary data for this session

In this session, we will use a synthetic data set based on the data from the *GESIS Panel Special Survey on the Coronavirus SARS-CoV-2 Outbreak in Germany*. This synthetic data set was created by Bernd using the `synthpop` package. Apart from being synthetic, the data we use here differ from the original data set in two ways: 1) They only include numeric variables (and no value or variable labels), and 2) all values < 0 have been recoded as `NA`. You can find this file in the `data` folder within the workshop materials.

Original data set:

GESIS Panel Team (2020). *GESIS Panel Special Survey on the Coronavirus SARS-CoV-2 Outbreak in Germany*. GESIS Data Archive, Cologne. ZA5667 Data file Version 1.1.0, https://doi.org/10.4232/1.13520

Brief 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 (e.g., gt or flextable). Discussing all of them would be too much for this workshop. An easy-to-use and quite versatile option is `knitr::kable()` which can be nicely extended using the kableExtra package.

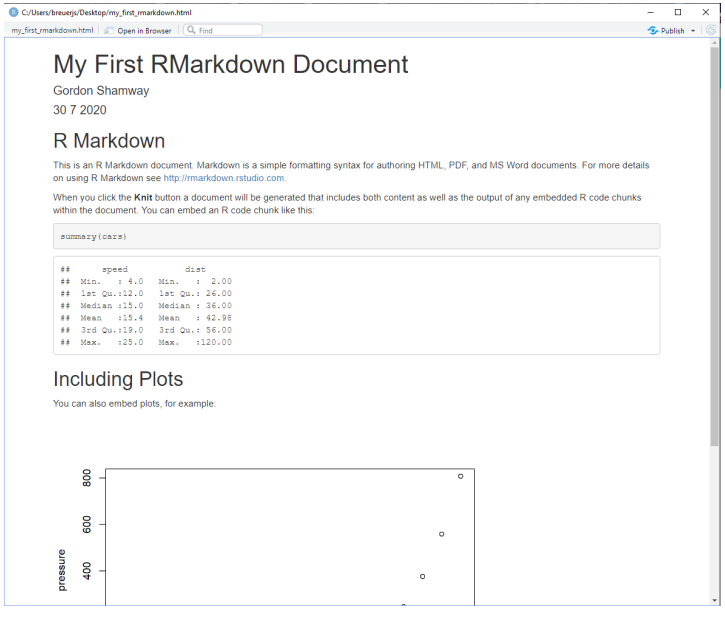
Brief excursus: Tables in R Markdown

```
library(knitr)

gp_covid <- read.csv("./data/ZA5667_v1-0-0_CSV_synthetic-data.csv")

kable(table(gp_covid$sex),
      col.names = c("Sex",
                    "Frequency"))
```

Sex	Frequency
1	1933
2	1832

Knitting

To compile the `R Markdown` source file (in this case into an `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.

```
rmarkdown::render('my_report.Rmd',
                  output_file = '../output/my_report.html')
```

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.

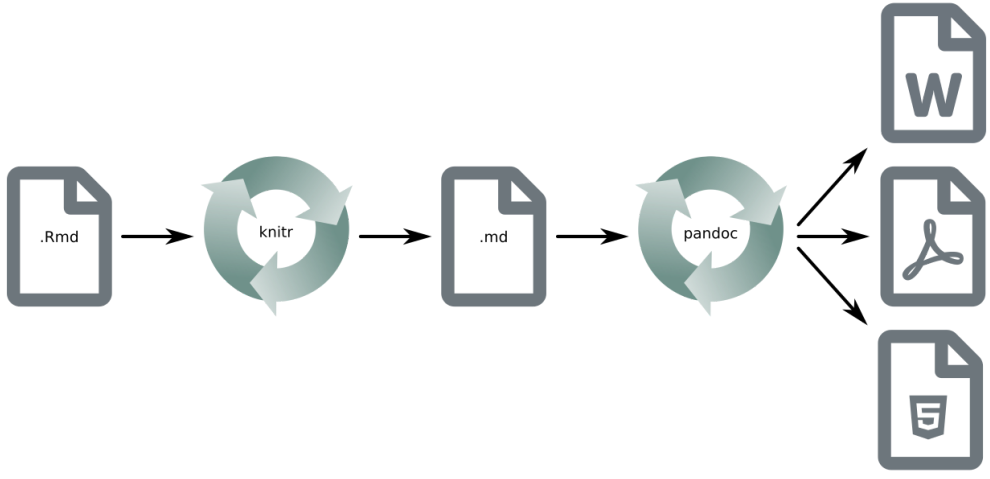
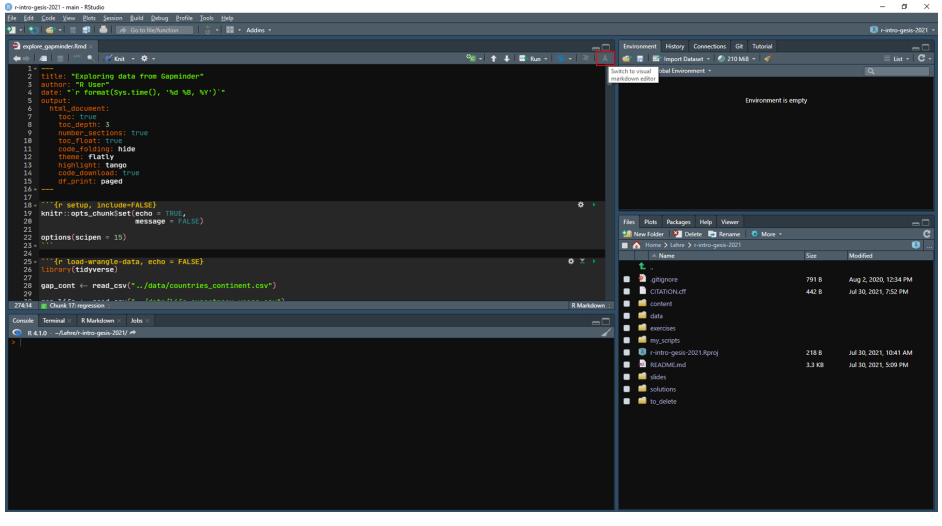


Figure by Andrew Collier

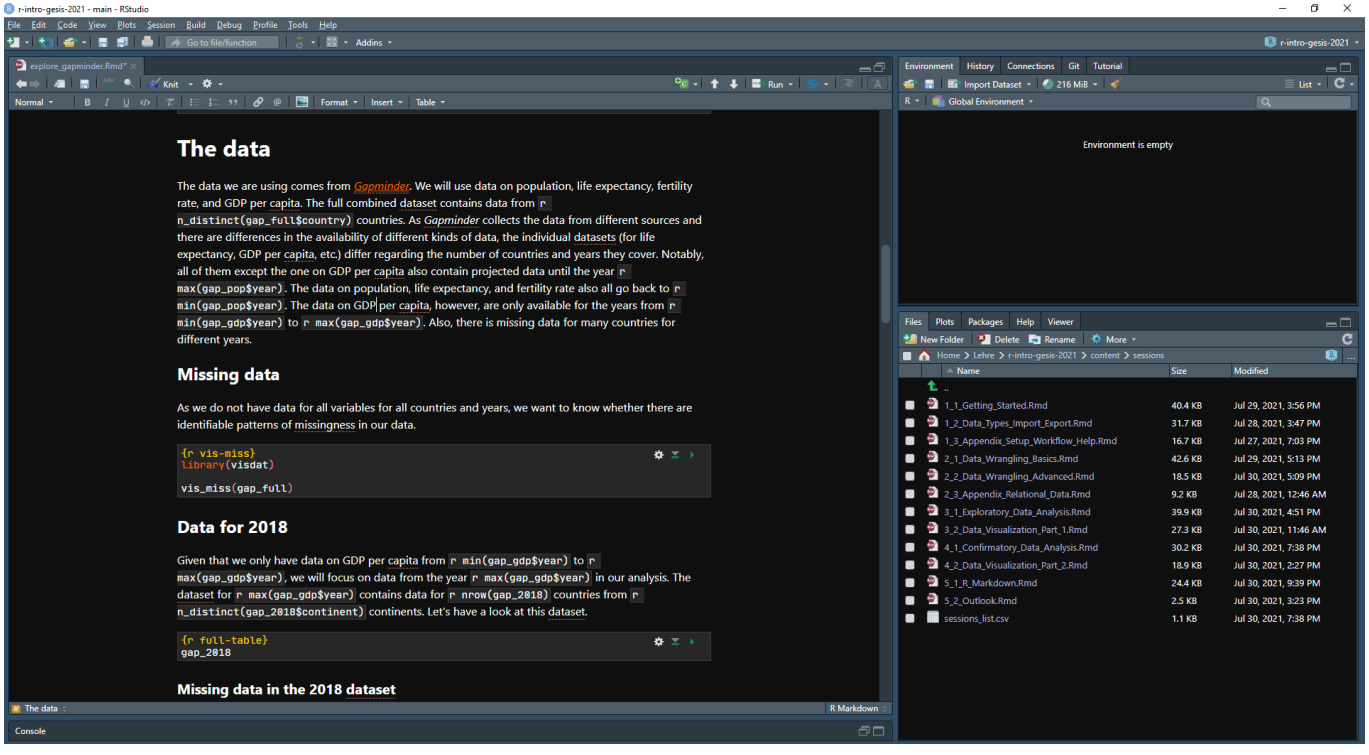
Visual `R Markdown` editor

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*.



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

```
sessionInfo()
```

```
## R version 4.1.3 (2022-03-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## 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] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] depgraph_0.1.0      emo_0.0.0.9000      webshot2_0.0.0.9000 tweetrmd_0.0.8
##  [5] knitr_1.37          forcats_0.5.1       stringr_1.4.0       dplyr_1.0.6
##  [9] purrr_0.3.4         readr_1.4.0         tidyr_1.1.3         tibble_3.1.2
## [13] ggplot2_3.3.6       tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] websocket_1.4.0     colorspace_2.0-1    ellipsis_0.3.2      easypackages_0.1.0
##  [5] woRkshoptools_0.1.0 rprojroot_2.0.2     fs_1.5.0            xaringanExtra_0.7.0
##  [9] rstudioapi_0.13     farver_2.1.0        xaringan_0.25       remotes_2.3.0
## [13] ggrepel_0.9.1       chromote_0.0.0.9003 fansi_0.4.2         lubridate_1.7.10
## [17] xml2_1.3.2          cachem_1.0.5        pkgload_1.2.1       jsonlite_1.7.2
## [21] broom_0.7.6         dbplyr_2.1.1        png_0.1-7           compiler_4.1.3
## [25] httr_1.4.2          backports_1.2.1     assertthat_0.2.1    fastmap_1.1.0
## [29] cli_3.2.0           later_1.2.0         miniCRAN_0.2.16     htmltools_0.5.2
```

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.

One reason why did not switch to to Quarto for this workshop (besides not having a lot of experience with it) is that extensions like xaringan or papaja do not (yet?) work with Quarto.

Also, for R, Quarto uses R Markdown under the hood, so everything you learn here is fully compatible with Quarto.