R Markdown Workshop

Creation, organisation, storage, and collaboration

Anne-Kathrin Kleine | University of Groningen

R Markdown Workshop

Creation, organisation, storage, and collaboration

Anne-Kathrin Kleine | University of Groningen

R Markdown Workshop

Creation, organisation, storage, and collaboration

Anne-Kathrin Kleine | University of Groningen

...taking the pain away

- 💅 efficient management of **code** and **results**
- 📕 easy conversion to pdf, html, word
- 🧘 guaranteed reproducibility, less mistakes

Workshop overview

Day 1:

1. R Markdown basics
2. Writing text
3. Reference management
4. Writing code
5. Figures, tables, and plots
6. R project organization

Day 1:

1. R Markdown basics
2. Writing text
3. Reference management
4. Writing code
5. Figures, tables, and plots
6. R project organization

GOAL 1 💪 Write your analysis scripts in R Markdown

GOAL 2 💪 Convert the output into a format of your choice (word, html, pdf)

GOAL 3 💪 Add text, figures/tables, and references

GOAL 4 💪 Learn proper project structuring and using templates

Day 2:

1. Git & GitHub basics

2. Collaborating using Git & GitHub

3. Osf integration

4. Blogdown - build your project website

5. Xaringan, bookdown & Co.

Day 2:

1. Git & GitHub basics

2. Collaborating using Git & GitHub

3. Osf integration

4. Blogdown - build your project website

5. Xaringan, bookdown & Co.

GOAL 1 💪 Learn how to use Git & GitHub for version control

GOAL 2 💪 Learn how to collaborate using Git & GitHub

GOAL 3 💪 Share your code and data through osf

GOAL 4 💪 Build your own data analysis project website

GOAL 5 💪 Learn how to use R & Markdown to produce a multitude of outputs (books, presentations, interactive applications)

General organization:

Presentation, then practice

You will work in breakout rooms, groups of 3 to 4

❓ ❓ Questions ❓ ❓

1. Ask you group 👰

2. Google (together) 👱‍♂️👩👩

3. Ask me 👷

Find me at...

LinkedIn
GitHub
AnneKathrinKleine.com
a.k.kleine@rug.nl

Or find me ...

Or find me ...

Or find me …

How familiar are you with using R for data analysis?

Go to https://www.menti.com/yzy8sx3mb6 🤸 (and enter the code **6959 8626**)

See results

Major resources:

Advanced R Markdown Workshop R Conf 2019

R Markdown: The Definitive Guide

R Markdown Workshop by Resul Umit

Detailed Outline

- Did you do the setup?
- Creating an R project🐣
- The YAML header
- Writing text 📝
- Reference management
- Writing code
- Figures
- Plots - ggplot2
- Structure your project
- 🎉 Yey, you're all set! 🎉
- Your turn! 🤷

Did you do the

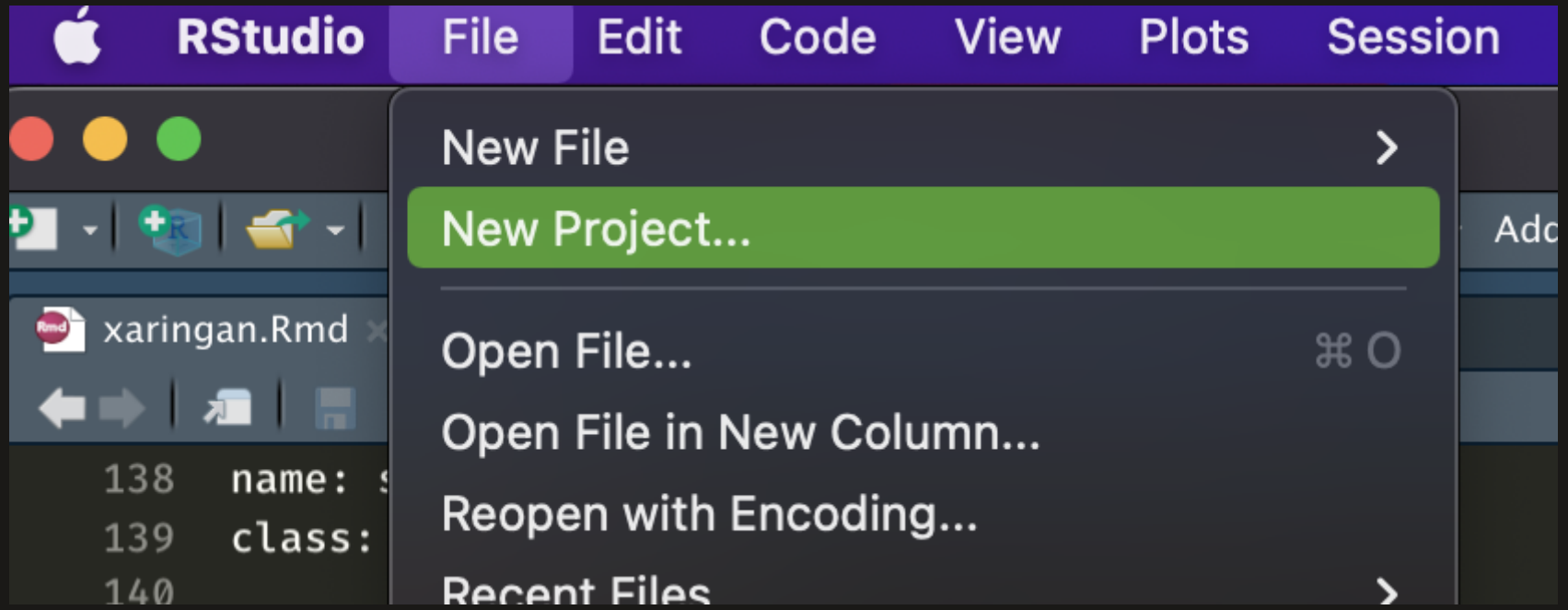
setup?

Link to setup

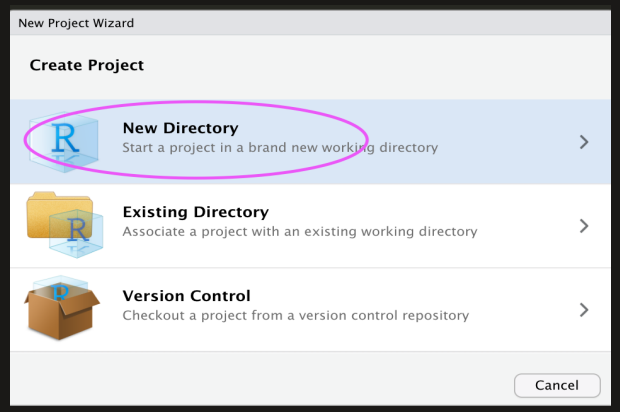
Creating an R project🐣

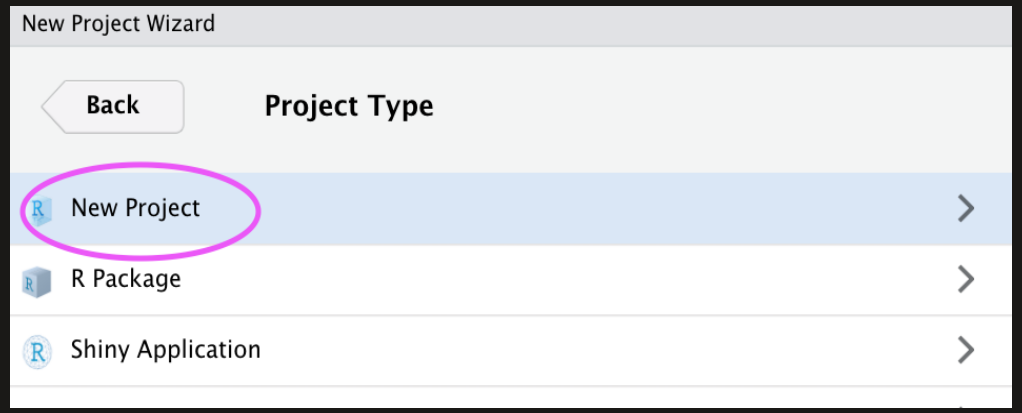
Creating an R project

Create a new R Project from within R Studio

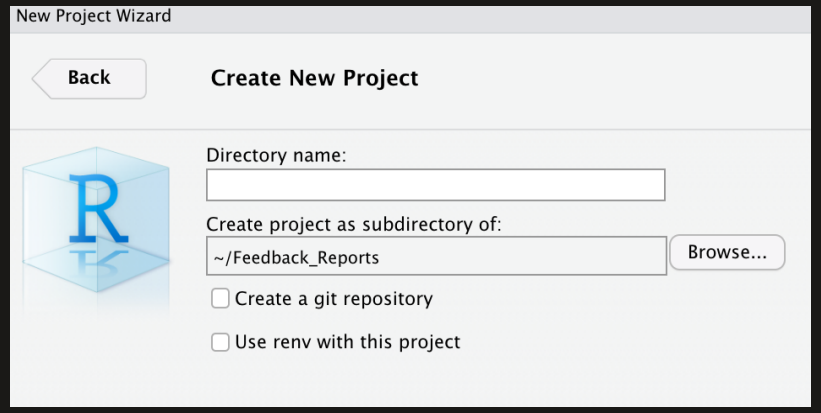


Save in new directory

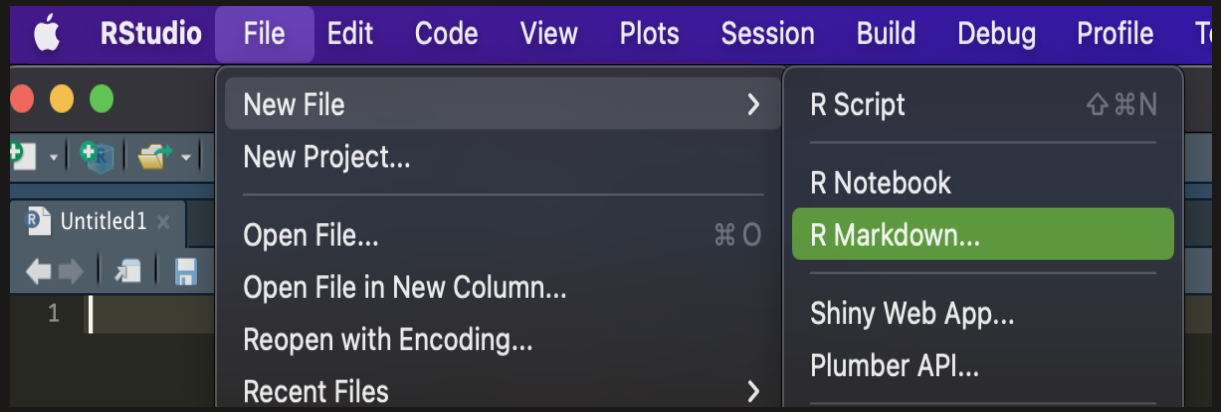




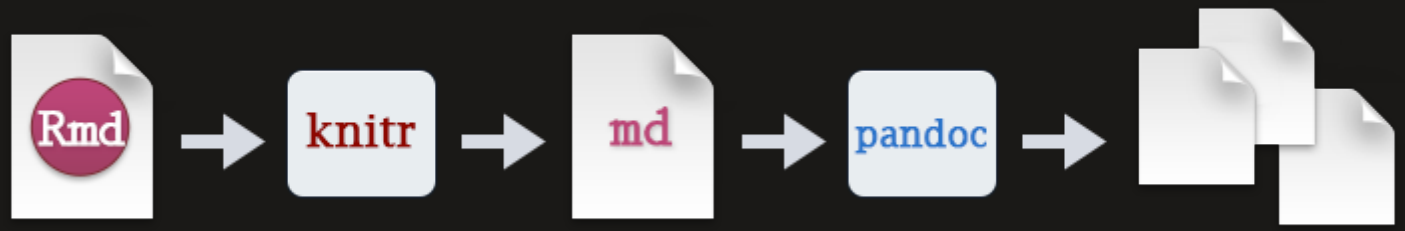
Choose the folder where you want to store your R Markdown file



In that new folder, create an R Markdown file



When you knit, the following happens:



knit executes the code if there is any, converts the resulting document from .Rmd (R Markdown) into .md (Markdown)

pandoc transforms the .md document into your preferred output format(s) (e.g., word, pdf, html)

The YAML header

YAML — General

YAML includes the metadata variables

- e.g., title, output format
- written between a pair of three hyphens -

```
---
title:
output:
---
```

YAML — Variables

- see Pandoc User's Guide and R Markdown Cheat Sheet for documentation and help
- Typical YAML variables for an research paper are as follows:

```
---
title:
author:
date:
bibliography:
csl:
output:
---
```

YAML — Variables

Variables can be provided as strings

```
---
title: "My very funny and descriptive article title"
output:
---
```

YAML — Variables

Variables can be provided as strings, options

```
---
title: "My very funny and descriptive article title"
output: pdf_document
---
```

YAML — Variables

Variables can be provided as strings, options, and sub-options

```
---
title: "My very funny and descriptive article title"
output:
    pdf_document:
        keep_tex: true
---
```

YAML — Variables

Variables can be provided as strings, options, sub-options, and code

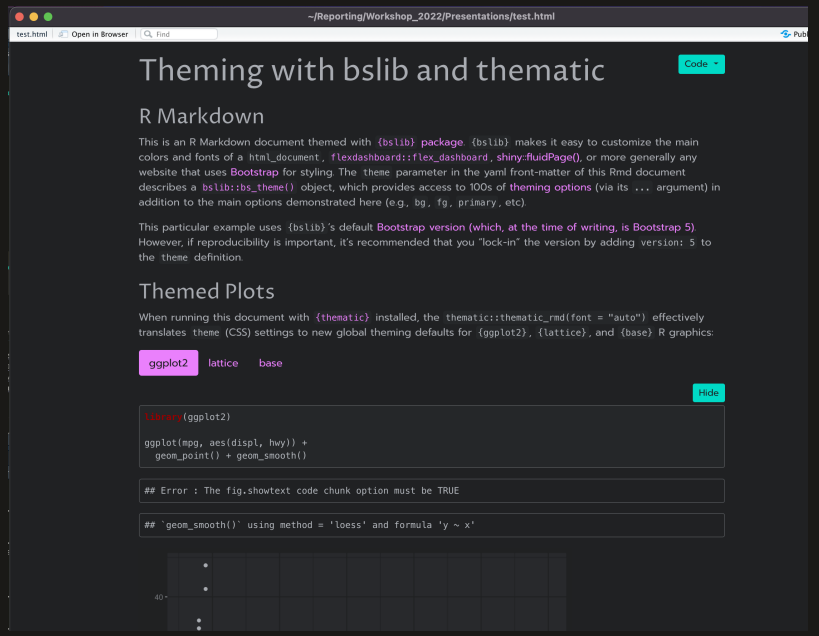
```
---
title: "My very funny and descriptive article title"
output:
    pdf_document:
        keep_tex: true
date: "\`r format(Sys.Date(), '%d %B %Y')`"
---
```

YAML — Variables — Output Formats

Documents as output formats include

- HTML

```
---
title: "Theming with bslib and thematic"
output:
  html_document:
    ...
---
```

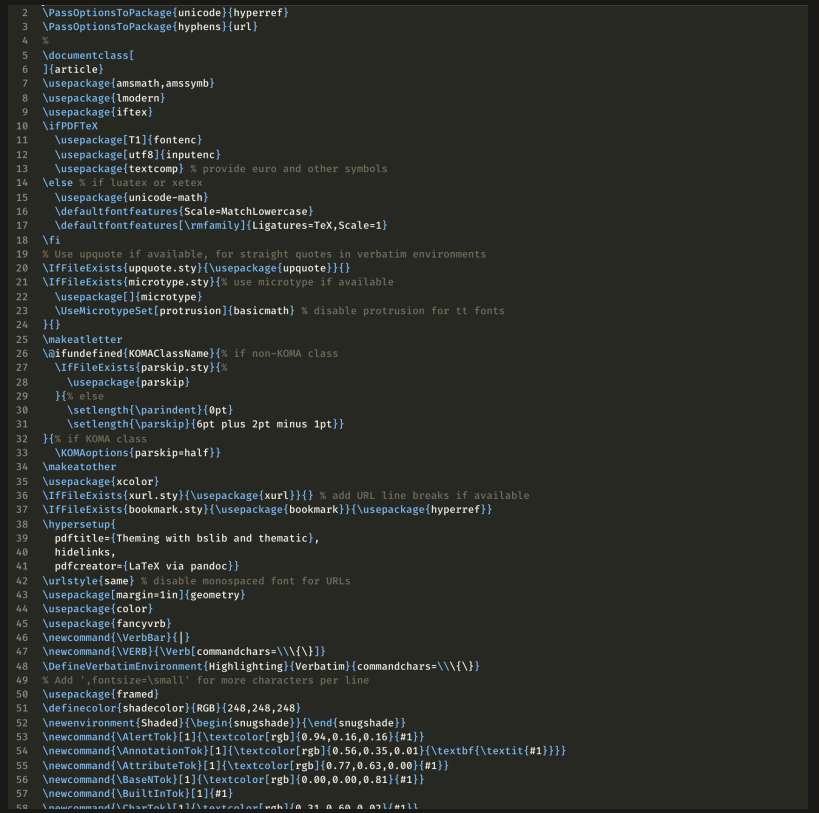


YAML — Variables — Output Formats

Documents as output formats include

- HTML
- LaTeX

```
---
title: "Theming with bslib and thematic"
output: latex_document
---
```

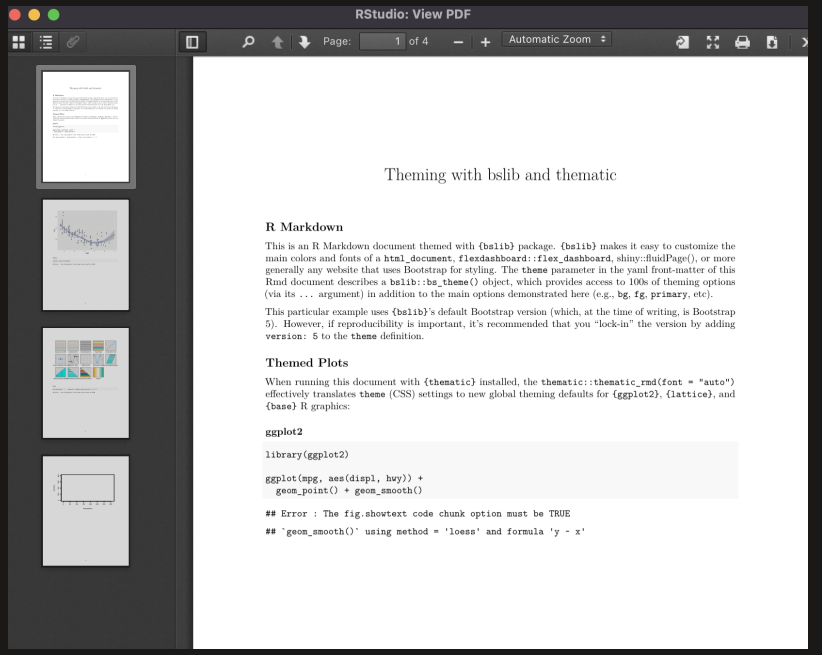


YAML — Variables — Output Formats

Documents as output formats include

- HTML
- LaTeX
- PDF

```
---
title: "Theming with bslib and thematic"
output: pdf_document
---
```

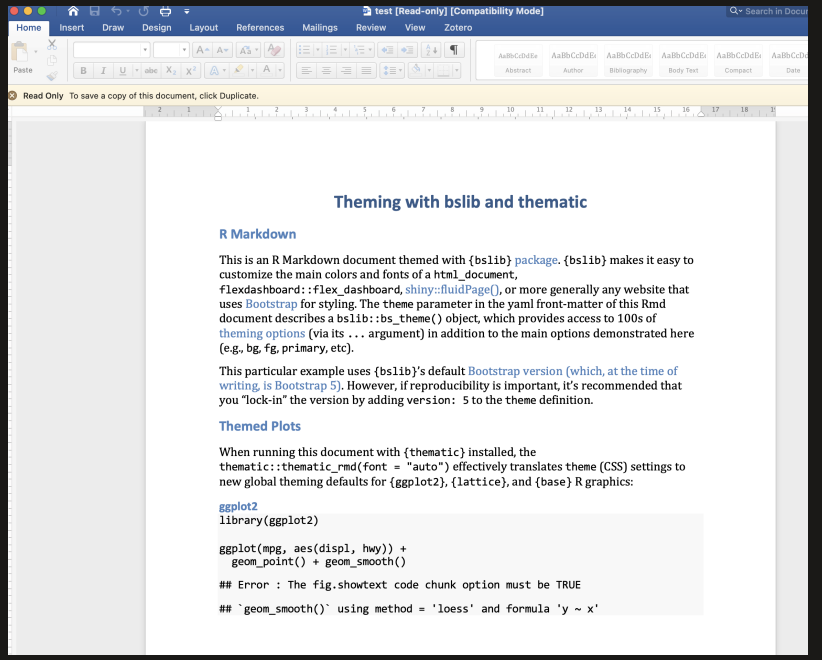


YAML — Variables — Output Formats

Documents as output formats include

- HTML
- LaTeX
- PDF
- Word

```
---
title: "Theming with bslib and thematic"
output: word_document
---
```



YAML — External Files

```
---
title: "Journal title"
author: "Jane Doe^[Department of Science, University of Random. Email: jane.doe@random.edu. Website: http://www.jane
date: 4 March 2022
bibliography: references.bib
csl: apa.csl
output: pdf_document
---
```

YAML — Strings — External Files

```
---
...
bibliography: references/ref_library.bib
csl: "../../paperstyle/csl/apa.csl" # two dots (..) mean moving one folder up
...
---
```

- ❗ ❗ locations of files re specified as relative to the working directory ❗ ❗
- for reproducibility reasons, hard-coded stings should be avoided 🙏
  - e.g., 👎 `"C:/Users/Dropbox/styles/apa.csl"` 👎

Writing text 

Lines

Multiple spaces on a given line are reduced to one

```
This is a sentence followed by four spaces.    This is another sentence on the same line.
```

This is a sentence followed by four spaces. This is another sentence on the same line.

Line endings with fewer than two spaces are ignored

```
This is a sentence followed by one space.
This is another sentence on a new line.
```

This is a sentence followed by one space. This is another sentence on a new line.

Hard Breaks

Two or more spaces at the end of lines introduce hard breaks, forcing a new line

```
This is a sentence followed by two spaces.
This is another sentence on a new line.
```

This is a sentence followed by two spaces.
This is another sentence on a new line.

Line Blocks

Spaces on lines that start with a vertical line | are kept

```
| a one-space indent
|     a five-space indent
|          a ten-space indent
```

a one-space indent
  a five-space indent
    a ten-space indent

Block Quotes

Lines starting with the greater-than sign `>` introduce block quotes*

```
> In God, we trust. All others must bring data.
>
> --- Anonymous
```

> In God, we trust. All others must bring data.
>
> — Anonymous

Paragraphs

One or more blank lines introduce a new paragraph

```
This is the first sentence of a paragraph as it is preceded by a blank line. This is the second
sentence of that paragraph, which is followed by a blank line.

This is the first sentence of a *new paragraph* as it is preceded by a blank line.
```

This is the first sentence of a paragraph as it is preceded by a blank line. This is the second sentence of that paragraph, which is followed by a blank line.

This is the first sentence of a *new paragraph* as it is preceded by a blank line.

Comments

Text with the syntax `<!-- comments -->` is omitted from output

```
<!-- This paragraph needs re-writing -->
This is the first sentence of a paragraph as it is preceded by a blank line.

This is the first sentence of a new paragraph <!-- I've removed italics --> as it is preceded
by a blank line.
```

This is the first sentence of a paragraph as it is preceded by a blank line.

This is the first sentence of a new paragraph as it is preceded by a blank line.

Headers

The number sign # introduces headers; lower levels are created with additional signs — up to total five levels

`# Introduction` becomes

Introduction

`## 1. Introduction` becomes

1. Introduction

`### 3.1 Introduction` becomes

3.1 Introduction

`#### Introduction` becomes

Introduction

`##### Introduction` becomes

Introduction

Emphases

A pair of single asterisk `*` or underscores `_` introduces italics

`*italics*` becomes *italics*

`_italics_` becomes *italics* as well

A pair of double asterisk or underscores introduces bold

`**bold**` becomes **bold**

`__bold__` becomes **bold** as well

These two rules can be combined

`**_bolditalics_**` becomes ***bolditalics***

`_**bolditalics**_` becomes ***bolditalics*** as well

Strikethrough

A pair of double tildes ~ introduces strikethrough

`~~strikethrough~~` becomes ~~strikethrough~~

Strikethrough can be combined with italics or bold

`**~~strikebold~~**` or `__~~strikebold~~__`, they both become ~~**strikebold**~~

`~~**strikebold**~~` or `~~__strikebold__~~`, they both become ~~**strikebold**~~ as well

`*~~strikeitalitcs~~*` or `_~~strikeitalitcs~~_`, they both become ~~*strikeitalitcs*~~

`~~*strikeitalitcs*~~` or `~~_strikeitalitcs_~~`, they both become ~~*strikeitalitcs*~~ as well

External links

You can link text to URLs

`[visit my website](https://anneokk.netlify.app//)` becomes visit my website

You can also link text to an email address

`[email me](mailto:a.k.kleine@rug.nl)` becomes email me

`<a.k.kleine@rug.nl>` becomes a.k.kleine@rug.nl

Equations

Inline equations go between a pair of single dollar signs "$" with no space between the signs and the equation itself

`$E = mc^{2}$` becomes $E = mc^{2}$

Block equations go in between a pair of double dollar signs

`$$E = mc^{2}$$` becomes

$$E = mc^{2}$$

Inline Notes

For inline footnotes, use the `^[footnote]` syntax

`Jane Doe^[Corresponding author.]` becomes Jane Doe[1]

[1] Corresponding author.

Lists

Lines starting with asterisk * as well as plus + or minus – signs introduce lists

```
- books
- articles
- reports
```

- books
- articles
- reports

Lists

Lists can be nested within each other, with indentation

```
+ books
+ articles
    - published
    - under review
        + revised and resubmitted
    - work in progress
```

- books
- articles
  - published
  - under review
    - revised and resubmitted
  - work in progress

Lists

List items can be numbered

```
1. books
2. articles
    - published
    - under review
        + revised and resubmitted
    - work in progress
```

1. books
2. articles
    - published
    - under review
        - revised and resubmitted
    - work in progress

Dashes

Two hyphens grouped together introduce an en-dash

`--` becomes –

Three hyphens grouped together introduce an em-dash

`---` becomes —

Subscripts and Superscripts

A pair of tildes introduces subscript

`C0~2~` becomes CO_2

A pair of carets introduces superscript

`R^2^` becomes R^2

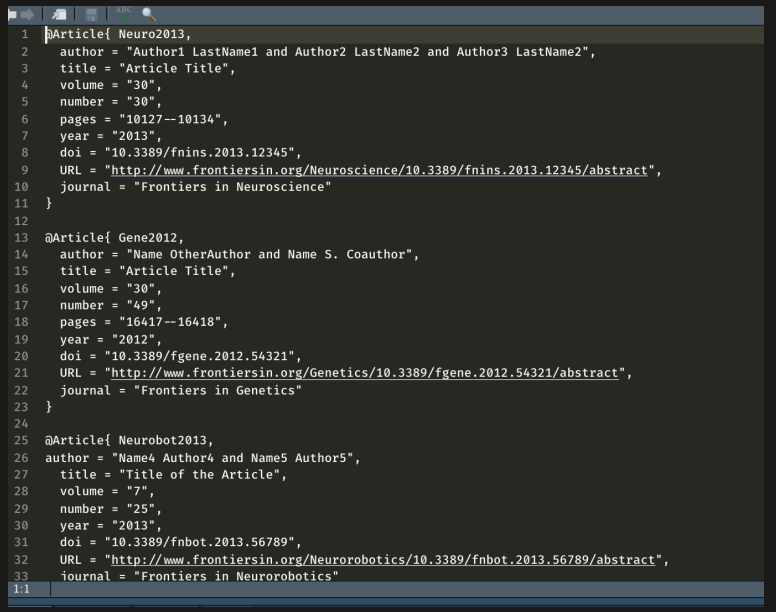
Reference management

References — Bibliography Database

- References are defined in .bib files
  - they follow the BibTeX format
- pandoc looks for a .bib file, and for the definitions therein, to process citations

```
---
...
bibliography: refs.bib
...
---
```

- .bib files are specified with the `bibliography` variable in YAML



```
@Article{ Neuro2013,
  author = "Author1 LastName1 and Author2 LastName2 and Author3 LastName2",
  title = "Article Title",
  volume = "30",
  number = "30",
  pages = "10127--10134",
  year = "2013",
  doi = "10.3389/fnins.2013.12345",
  URL = "http://www.frontiersin.org/Neuroscience/10.3389/fnins.2013.12345/abstract",
  journal = "Frontiers in Neuroscience"
}

@Article{ Gene2012,
  author = "Name OtherAuthor and Name S. Coauthor",
  title = "Article Title",
  volume = "30",
  number = "49",
  pages = "16417--16418",
  year = "2012",
  doi = "10.3389/fgene.2012.54321",
  URL = "http://www.frontiersin.org/Genetics/10.3389/fgene.2012.54321/abstract",
  journal = "Frontiers in Genetics"
}

@Article{ Neurobot2013,
author = "Name4 Author4 and Name5 Author5",
  title = "Title of the Article",
  volume = "7",
  number = "25",
  year = "2013",
  doi = "10.3389/fnbot.2013.56789",
  URL = "http://www.frontiersin.org/Neurorobotics/10.3389/fnbot.2013.56789/abstract",
  journal = "Frontiers in Neurorobotics"
```

- A BibTeX entry consists of three elements
  - a type (e.g., `@Article`)
  - a citation-key (e.g., `Neuro2013`)
  - a number of tags (e.g., `title`, `volume`, ...)

References — Bibliography Database — Entries

Get information via `Google Scholar`:

follow `cite -> BibTex` and copy

paste into .bib, edit if necessary, and save

References — Bibliography Database — Entries

Get information via `Google Scholar`:

follow `cite -> BibTex` and copy

paste into .bib, edit if necessary, and save

Better options: Zotero, Mendeley

⭐ Useful article on how to use Zotero with R Markdown ⭐

References — Style

- Reference styles are defined in `.csl` files
- files for different styles (e.g., APA) are available at https://www.zotero.org/styles

```
---
...
csl: "../apa.csl"
...
---
```

- .csl files are specified with the `csl` variable in YAML

References — In-text Citation Syntax — Author-Date Styles (APA 7)

`[@bennett2015]` becomes (Bennett, 2015)

`@bennett2015` becomes Bennett (2015)

`[-@bennett2015]` becomes (2015)

`-@bennett2015` becomes 2015

`[@bennett2015 35]` becomes (Bennett, 2015, p. 35)

`[@bennett2015 33-35]` becomes (Bennett, 2015, pp. 33–35)

`[@bennett2015, ch. 1]` becomes (Bennett, 2015, ch. 1)

`[@bennett2015; @gilbert2019]` becomes (Bennett, 2015; Gilbert, 2019)

`[see @bennett2015, for details]` becomes (see Bennett, 2015, for details)

`@bennett2015 [33-35]` becomes Bennett (2015, pp. 33–35)

Citations — Reference List

```
This is the last sentence of an APA style manuscript.
## References
```

This is the last sentence of an APA style manuscript.

References

Bennett, S. (2015). Peanut butter and jelly. *Journal of Bone, 1*(12), 3–35.

Gilbert, T. (2019). Turning wine into water. In M. Albert (Ed.), *The book of ground* (pp. 124–142). Antman.

Writing code

Data: Superhero Dataset

Code, in and outside chunks

Code chunks:

```
library(readxl)
library(dplyr)
df <- read_excel("data/heroes_information.xlsx") %>%
  na_if(., -99) %>%
  mutate(bmi = round(Weight/((Height/100)**2)), 2) %>%
  rename(., Water_allergy = Gender) %>%
  mutate(Water_allergy = recode(Water_allergy, Male = "Yes", Female = "No")) %>%
  select(., -...1, -`2`)
```

Inline:

`The average height of a superhero is` `r round(mean(df$Height, na.rm = T), 2)` `centimeter`
becomes:

The average height of a superhero is 186.73 centimeter.

Code Chunks

- Code chunks are delimited spaces between a pair of three backticks ```
- their output, if there is any, appears in the output document

```
```{r, setup, echo=FALSE}
```
```

- in curly brackets `{}`, code chunks take
  - a language engine (e.g., `r`)
  - a label (e.g., `setup`)
  - options (e.g., `echo = FALSE`)
    - The complete list of options is available at https://yihui.org/knitr/options

Chunk options

`echo = FALSE` to exclude code

````
```{r, show_df, echo = FALSE}
head(df)
```
````

```
## # A tibble: 6 × 11
##   name          Water_allergy `Eye color` Race             `Hair color` Height Publisher       `Skin color` Align
##   <chr>         <chr>         <chr>       <chr>            <chr>         <dbl> <chr>           <chr>        <chr>
## 1 A-Bomb        Yes           yellow      Human            No Hair         203 Marvel Comics   -            good
## 2 Abe Sapien    Yes           blue        Icthyo Sapien    No Hair         191 Dark Horse Com… blue         good
## 3 Abin Sur      Yes           blue        Ungaran          No Hair         185 DC Comics       red          good
## 4 Abomination   Yes           green       Human / Radiation No Hair        203 Marvel Comics   -            bad
## 5 Abraxas       Yes           blue        Cosmic Entity    Black            NA Marvel Comics   -            bad
## 6 Absorbing Man Yes           blue        Human            No Hair         193 Marvel Comics   -            bad
```

Chunk options

`echo = TRUE` to include code and output

````
```{r, show_df_1, echo = TRUE}
head(df)
```
````

```
head(df)
```

```
## # A tibble: 6 × 11
##   name          Water_allergy `Eye color` Race            `Hair color` Height Publisher       `Skin color` Align
##   <chr>         <chr>         <chr>       <chr>           <chr>         <dbl> <chr>           <chr>        <chr>
## 1 A-Bomb        Yes           yellow      Human           No Hair         203 Marvel Comics   -            good
## 2 Abe Sapien    Yes           blue        Icthyo Sapien   No Hair         191 Dark Horse Com… blue         good
## 3 Abin Sur      Yes           blue        Ungaran         No Hair         185 DC Comics       red          good
## 4 Abomination   Yes           green       Human / Radiation No Hair       203 Marvel Comics   -            bad
## 5 Abraxas       Yes           blue        Cosmic Entity   Black            NA Marvel Comics   -            bad
## 6 Absorbing Man Yes           blue        Human           No Hair         193 Marvel Comics   -            bad
```

Chunk options

`include = FALSE` to exclude entire chunk content (code and output)

````
```{r, show_df_2, include = FALSE}
head(df)
```
````

Chunk options

`results = "hide"` to include only code (no output)

````
```{r, show_df_3, results = "hide"}
head(df)
```
````

```
head(df)
```

Chunk options

Cache results, useful for complex analyses:

```
```{r ... cache=TRUE}
```

Prevent evaluation:

```
```{r ... eval=FALSE}
```

Show results as produced by code (no transformation with pandoc, necessary for some packages (e.g., stargazer)):

```
```{r ... results="asis"}
```

Prevent showing warnings, messages, errors:

```
```{r ...error=FALSE, message=FALSE, warning=FALSE}
```

The Setup Chunk

It is recommended to use the first code chunk for general setup, where you can

- define your own defaults for chunk options, with `knitr::opts_chunk$set()`
- load the necessary packages
- import raw data

````
```{r, setup, include=FALSE}
## chunk option defaults
knitr::opts_chunk$set(echo=FALSE, message=FALSE)
## packages
library(dplyr)
library(ggplot2)
library(stargazer)
library(readxl)
```
````

The Data Chunk

Second chunk for the main operations on raw data

- cleaning, transformations

````
```{r, data, ...}
df <- read_excel("data/heroes_information.xlsx") %>%
  na_if(., -99) %>%
  mutate(bmi = round(Weight/((Height/100)**2)), 2) %>%
  rename(., Water_allergy = Gender) %>%
  mutate(Water_allergy = recode(Water_allergy, Male = "Yes", Female = "No")) %>%
  select(., -...1, -`2`)

```
````

Inline Code

```
If we multiply _pi_ by 5, we get `r pi * 5`.
```

If we multiply *pi* by 5, we get 15.7079633.

```
The average height of a superhero in the dataset is `r mean(df$Height, na.rm = T)`, which would
round to `r round(mean(df$Height, na.rm = T), digits = 1)`.
```

The average height of a superhero in the dataset is 186.7263056, which would round to 186.7.

```
__Only `r nrow(subset(df, Height < 100))` superheros__ in the dataset are Smallings.
```

Only 9 superheros in the dataset are Smallings.

Figures

Figures — Markdown Syntax

The syntax `![Figure Caption](figure.extension)` embeds images, and/or figures into .Rmd documents

```
![A superhero's best friend](img/superdog.jpg)
```

Figure 1: A superhero's best friend.

Figures — knitr

The knitr package offers a capable alternative with the include_graphics() function

````
```{r, superdog, echo=FALSE, fig.cap="A superhero's best friend."}
knitr::include_graphics("img/superdog.jpg")
```
````

Figure 1: A superhero's best friend.

Figures — knitr

Size is defined with the chunk options `out.width` or `out.height`

````
```{r ... out.width="15%"}
knitr::include_graphics("img/superdog.jpg")
```
````

````
```{r ... out.width="40%"}
knitr::include_graphics("img/superdog.jpg")
```
````

Plots - ggplot2

ggplot2

1) The `ggplot` function and the `data` argument

```
ggplot(data = df)
```

ggplot2

1) The `ggplot` function and the `data` argument

```
ggplot(data = df)
```

2) The mapping aesthetics, or aes; most importantly, the variable(s) that we want to plot

```
ggplot(data = df,
       mapping = aes(x = Height, y = Weight, color = Alignment))
```

ggplot2

1) The `ggplot` function and the `data` argument

```
ggplot(data = df)
```

2) The mapping aesthetics, or aes; most importantly, the variable(s) that we want to plot

```
ggplot(data = df,
       mapping = aes(x = Height, y = Weight, color = Alignment))
```

3) The geometric objects, or geom; the visual representations

```
ggplot(data = df,
       mapping = aes(x = Height, y = Weight, color = Alignment)) +
       geom_point()
```

ggplot2

Put the code in a chunk, and give it a caption

````
```{r, scatterplot, `fig.cap = "Superheros size"`}
ggplot(data = df,
       mapping = aes(x = Height, y = Weight, color = Alignment)) +
       geom_point()
```
````

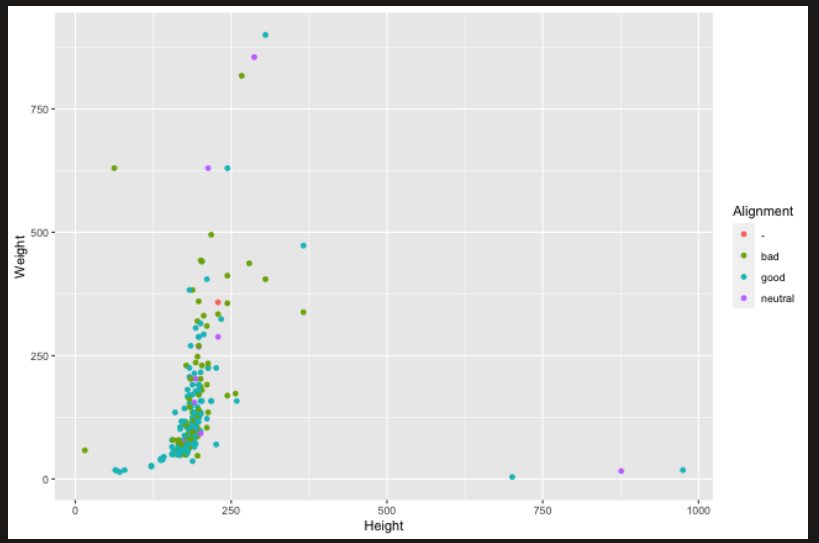
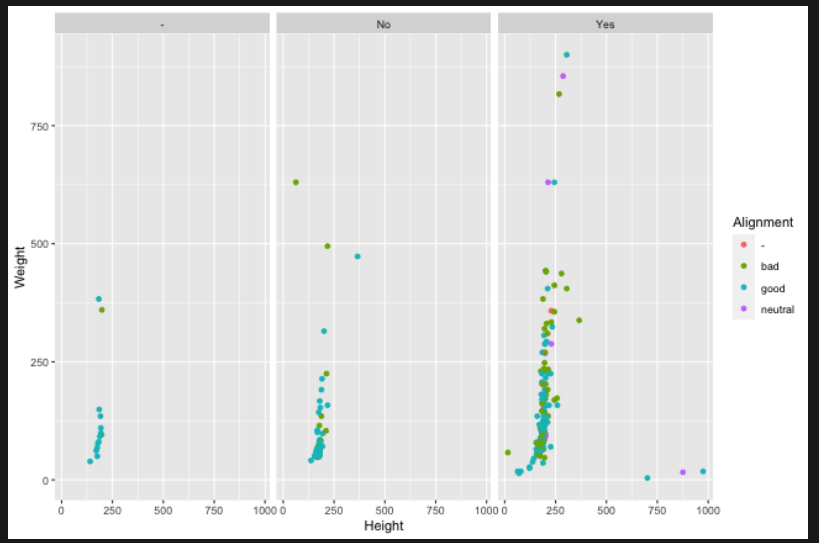
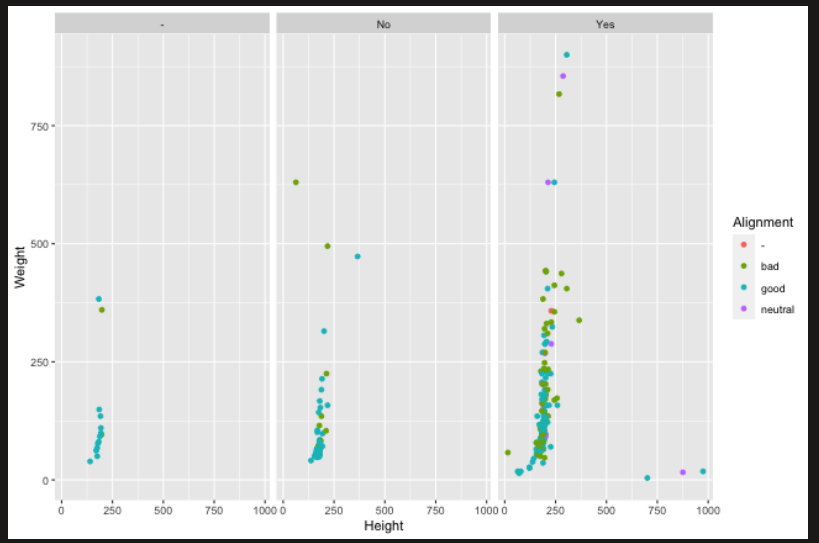
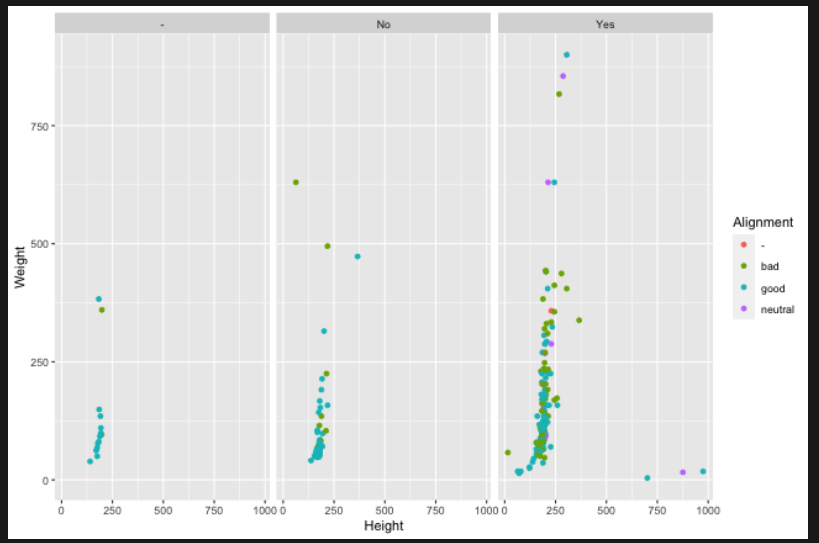


Figure 1. A scatterplot of superheros' height and weight.

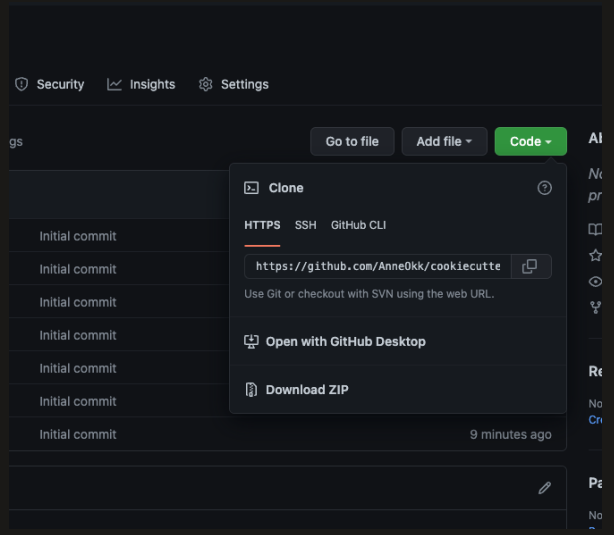
ggplot2

Add facets for subgroups, e.g., `branch`

```
```{r, scatterplot, fig.cap = "A scatterplot of superheros' height
ggplot(data = df,
       mapping = aes(x = Height, y = Weight, color = Alignment))  +
       geom_point() +
       facet_wrap(. ~ Water_allergy)
```
```



Figure 1. A scatterplot of superheros' height and weight.

ggplot2

Add facets for subgroups, e.g., `branch`

```
```{r, scatterplot, fig.cap = "A scatterplot of superheros' height
ggplot(data = df,
       mapping = aes(x = Height, y = Weight, color = Alignment))  +
       geom_point() +
       facet_wrap(. ~ Water_allergy)
```
```

Every plot you can think of - you can create it in R! See R Graph Gallery!



Figure 1. A scatterplot of superheros' height and weight.

ggplot2

Add facets for subgroups, e.g., `branch`

```
```{r, scatterplot, fig.cap = "A scatterplot of superheros' height
ggplot(data = df,
       mapping = aes(x = Height, y = Weight, color = Alignment))  +
       geom_point() +
       facet_wrap(. ~ Water_allergy)
```
```

Every plot you can think of - you can create it in R! See R Graph Gallery!
Also have a look at the plotly graphical library.



Figure 1. A scatterplot of superheros' height and weight.

Tables

Tables with stargazer

- A capable package for creating at least three kinds of tables
    - raw data, in columns and rows
    - descriptive/summary statistics
    - regression models

Tables with `stargazer`

- The `stargazer` package requires specific settings
  - in the chunk options
  - and, in the `type` argument of the `stargazer()` function

- These settings depend on the desired output format,* as shown below

Output	Chunk Option	Type Argument
LaTex / PDF	results="asis"	latex
HTML	results="asis"	html
Word	comment=""	text

* The following slides use the setting for LaTex and PDF outputs.

Tables with `stargazer`

- `stargazer` tables look slightly different in different output formats
- workarounds for Word:
  - `knit` to HTML as well as Word, copy the tables from HTML to Word
  - `knit` to PDF, open the PDF in Word
  - use a different package to create tables, such as `huxtable`

Example data table with stargazer

Table of first three rows of the dataset

```
```{r, data_table, echo=FALSE, results="asis"}
stargazer(data = head(df, n = 3), type = "latex", summary = FALSE)
```
```

Example *data* table with stargazer

Table of first three rows of the dataset

````
```{r, data_table, echo=FALSE, results="asis"}
stargazer(data = head(df, n = 3), type = "latex", summary = FALSE)
```
````

Notice the options of the chunk and the arguments of the function

Example data table with `stargazer`

Table of first three rows of the dataset

````
```{r, data_table, echo=FALSE, results="asis"}
stargazer(data = head(df, n = 3), type = "latex", summary = FALSE)
```
````

Notice the options of the chunk and the arguments of the function

- with `results="asis"`, `knitr` will pass through results without reformatting them
- they should remain LaTeX (`type = "latex"`) because our outcome document is PDF, converted from LaTeX
- with `summary = FALSE`, the table will present the data, not its descriptive statistics

Example data table with stargazer

Table of the first three rows of the dataset

```
```{r, data_table, echo=FALSE, results="asis"}
stargazer(data = head(df, n = 3), type = "latex", summary = FALSE)
```
```

Table 1:

	name	Water_allergy	Eye color	Race	Hair color	Height	Publisher	Skin color	Alignment	Weight	bmi
1	A-Bomb	Yes	yellow	Human	No Hair	203	Marvel Comics	-	good	441	107
2	Abe Sapien	Yes	blue	Icthyo Sapien	No Hair	191	Dark Horse Comics	blue	good	65	18
3	Abin Sur	Yes	blue	Ungaran	No Hair	185	DC Comics	red	good	90	26

Example regression table with stargazer

Create a table of regression models instead

```
```{r, regression_table, echo=FALSE, results="asis"}
stargazer(data = lm(Height ~ Weight, data = df),
          type = "html", header = FALSE,
          title = "Regression Results")
```
```

Regression Results

	Dependent variable:
	Height
Weight	0.109***
	(0.025)
Constant	174.865***
	(3.843)
Observations	490
R^2	0.037
Adjusted R^2	0.035
Residual Std. Error	57.935 (df = 488)
F Statistic	18.971*** (df = 1; 488)
Note:	*p<0.1; **p<0.05; ***p<0.01

Summary statistics with vtable

```
library(vtable)
df  %>% select(Alignment, bmi) %>% st(.)
```

Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Alignment	734						
... -	7	1%					
... bad	207	28.2%					
... good	496	67.6%					
... neutral	24	3.3%					
bmi	490	38.557	133.965	0	21	30	2510

There are many alternatives for creating tables in R Markdown, see `knitr`, `kableExtra`, and `huxtable` packages.

Structure your project

Structure your project

Minimal example

```
.
└── example_project
    ├── R
    ├── README.md
    ├── analysis
    |    └── template.Rmd
    ├── data
    ├── docs
    ├── example_project.Rproj
    ├── output
    ├── .gitignore
```

- R - Resuable R code (functions etc.)
- analysis - R Markdown analysis files
- docs - Rendered analysis reports
- data - (Raw) data used for analysis
- output - Output files (e.g., figures, plots...)
- README.md - project description (will be displayed if added to GitHub)
- .gitignore - The purpose of gitignore files is to ensure that certain files remain untracked (not added when your code is pushed to GitHub); e.g., data that is not anonymized, irrelevant files

Easy project structuring with cookiecutter

1. Install cookiecutter

2. Generate a new analysis directory: type `cookiecutter gh:lazappi/cookiecutter-r-analysis` (in 📱 terminal 📱)

Useful folders to *add*

Subfolders for `/data`

```
├── data
    ├── raw
    └── processed
```

Subfolders for`/output`

```
├── output
    ├── figs
    └── ...
```

`/config` folder for csl and other custom styling

```
├── config
```

The `R` and `analysis` folders

Your .rmd file lives in the `/analysis` folder. This is where the final script is executed and the final output (e.g., a pdf) is created.

The `/R` folder is used for custom functions that you load in the main .rmd file via `source("remote-functions.R")` at the top of the file.

The `R` and `analysis` folders

Your .rmd file lives in the `/analysis` folder. This is where the final script is executed and the final output (e.g., a pdf) is created.

The `/R` folder is used for custom functions that you load in the main .rmd file via `source("remote-functions.R")` at the top of the file.

The setup chunk becomes:

````
```{r, setup, include=FALSE}
## chunk option defaults
knitr::opts_chunk$set(echo=FALSE, message=FALSE)
## packages
library(dplyr)
library(ggplot2)
library(stargazer)
library(readxl)
## custom functions
source("remote-functions.R")
```
````

Alternative: Easy project structuring with templates from GitHub

- Search for an example project structure that serves the purpose, e.g., here.
- Download the directory
- save at desired location



Next level stuff: R packages

Building R packages is beyond the scope of the workshop but definitely useful to get into. See R Packages Book for more information.

R Markdown templates

Setup for APA7 articles:

- install and load the papaja library

```
remotes::install_github("crsh/papaja@devel")
```

- Install the APA 7 (or another style) document class here
- Move csl into the correct folder (e.g., `/config`)
- Open an R Markdown APA template in your `/analysis` folder

Add this to the YAML header (for APA7 style, see also here):

```
header-includes:
  - |
    \makeatletter
    \renewcommand{\paragraph}{\@startsection{paragraph}{4}{\parindent}%
      {0\baselineskip \@plus 0.2ex \@minus 0.2ex}%
      {-1em}%
      {\normalfont\normalsize\bfseries\typesectitle}}

    \renewcommand{\subparagraph}[1]{\@startsection{subparagraph}{5}{1em}%
      {0\baselineskip \@plus 0.2ex \@minus 0.2ex}%
      {-\z@\relax}%
      {\normalfont\normalsize\bfseries\itshape\hspace{\parindent}{#1}\textit{\addperi}}{\relax}}
    \makeatother

csl               : "../config/apa.csl" #< path to csl file
documentclass     : "apa7"
```

Add this to the YAML header (to knit output to docs folder):

```
knit: (function(inputFile, encoding) {
  rmarkdown::render(inputFile, encoding = encoding, output_dir = "../docs") })
```

 Yey, you're all set! 

Your turn! 

Create an R project for your analysis (~ 45 min.)

1. Use `cookiecutter` to create a project from scratch or use a project template that serves the purpose, e.g., The Example Project
2. Create an R Markdown template that you would like to use (e.g., APA7 paper with `papaja` package)
3. If you already have an analysis script, move the files into the correct folders (e.g., raw data into `data/raw` subfolder, custom R functions into `R` folder) ❗ ❗ **Do not forget to refer to the files using relative paths** ❗ ❗
4. 🧶 KNIT to html or pdf