Writing Reproducible Research Papers with R Markdown

Resul Umit

November 2020

Skip intro — To the contents slide.

I can teach this workshop at your institution — Email me.

Who am I?

Resul Umit

- post-doctoral researcher in political science at the University of Oslo
- teaching and studying representation, elections, and parliaments
    - a recent publication: Parliamentary communication allowances do not increase electoral turnout or incumbents' vote share
- teaching workshops, also on
    - version control and collaboration
    - working with Twitter data
    - creating academic websites
    - automated web scraping
- more information available at resulumit.com

How did I use to write?

First, with Stata + Word, I was ...

- frustrated with Word
  - formatting tables, figures, citations, and equations
  - managing references
- tired of switching between programmes/screens
  - and, worried about making mistakes in between
- paying for programme licences

How did I use to write?

Then, with Stata + R + LaTeX, I was ...

- ~~frustrated with Word~~
  - ~~formatting tables, figures, citations, and equations~~
  - ~~managing references~~
- tired of switching between programmes/screens
  - and, worried about making mistakes in between
- paying for the Stata licence
- converting PDF documents to Word manually
  - coordinating work with co-authors who don't use LaTeX/PDF
  - submitting to journals which don't accept LaTeX/PDF

How do I write now?

Now, with R Markdown, I am ... happy!

- ~~frustrated with Word~~
  - ~~formatting tables, figures, citations, and equations~~
  - ~~managing references~~
- ~~tired of switching between programmes/screens~~
  - ~~and, worried about making mistakes in between~~
- ~~paying for the Stata licence~~
- ~~converting PDF documents to Word, manually~~
  - ~~coordinating work with co-authors who don't use LaTeX/PDF~~
  - ~~submitting to journals which don't accept LaTeX/PDF~~

R Markdown

- Efficient
  - write text, cite sources, tidy data, analyse, table, and plot it in one programme/screen
  - re-do one, more, or all of these with ease
    - decrease the possibility of making mistakes in the process
- Flexible
  - output to various formats
    - e.g., HTML, LaTeX, PDF, Word
- Open access/source
  - use for free
  - create documents accessible to anyone with a computer and internet connection
  - benefit from the work of a great community of users/developers

Reproducibilty — Before Publication

- Having written a complete draft
  - with data including re-coded variables, tables, figures, and text with references to specific results (e.g., numbers from summary and/or regression statistics)
- If you and/or your co-authors decide
  - to reverse a re-coded variable to its previous/original measure
  - and/or, to exclude a subgroup of observations from analysis
- How resource intensive would this revision be?
  - how long would this revision take?
  - how many programmes would be needed for this revision, and how much would they cost?
  - there is an inverse relationship between this resource intensity and reproducibilty

Reproducibilty — After Publication

- After your paper is published, if others, including your future self, would like to test how robust the results are
    - to reversing a re-coded variable to its previous/original measure
    - and/or, to excluding a subgroup of observations from analysis
- How resource intensive would this test be?
    - how accessible is the data, documentation (how was the variable re-coded in the first place?), and the code?
    - how long would the test take?
    - how many programmes would be needed for this revision, and how much would they cost?
    - there is an inverse relationship between this resource intensity and reproducibilty

The Workshop — Overview

- Two days, on how to write reproducible research papers with R Markdown
  - 200+ slides, 40+ exercises, and time for converting a real project
- Based on converting a mock manuscript written in Word to R Markdown
  - plus, improving its reproducibility and version-controlling it
  - with a PDF output in mind
- Designed for researchers with basic knowledge of R programming language
  - does not cover programming with R
    - e.g., writing functions
  - ability to regress, plot, and table in R will be very helpful
    - but not absolutely necessary — these skills can be developed after learning R Markdown as well

The Workshop — Contents

Part 1. Getting the Tools Ready

- e.g., downloading course material

Part 2. Introducing R Markdown

- e.g., creating a new document

Part 3. Setting Metadata

- e.g., defining output format

Part 4. Writing Text

- e.g., adding emphasis to text

Part 5. Managing References

- e.g., citing sources

Part 6. Adding Code, Figures, and Tables

- e.g., plotting data

Part 7. Addressing Functionality Gaps

- e.g., adjusting line spacing

Part 8. Using Version Control

- e.g., integrating Git and GitHub

Part 9. Collaborating with Others

- e.g., working simultaneously with co-authors

Part 10. Working on a Real Project

- e.g., converting a work-in-progress of yours

The Workshop — Organisation

- Sit in groups of two
  - participants learn as much from their partner as from instructors
  - switch partners after every second part
- Type, rather than copy and paste, the code that you will find on these slides
  - typing is a part of the learning process
- When you have a question
  - ask your partner
  - google together
  - ask me

The Workshop — Organisation — Slides

03:00

Slides with this background colour indicate that your action is required, for

- setting the workshop up
    - e.g., downloading course material
- completing the exercises
    - e.g., managing references in R Markdown
    - there are 40+ exercises
    - these slides have countdown timers

The Workshop — Organisation — Slides

- Codes and texts that go in R Markdown documents `appear as such — in a different font, on gray background`
  - long codes and texts will have their own line(s)

````
```{r, scatterplot, fig.cap = "A scatterplot of journal metrics."}

ggplot(data = df, mapping = aes(x = h5_median, y = h5_index, color = subfield)) +
       geom_point() +
       facet_wrap(. ~ branch) +
       scale_colour_discrete(name = "Journal Type", breaks = c(0, 1), labels = c("Generalist", "S

```
````

The Workshop — Organisation — Slides

- Codes and texts that go in R Markdown documents `appear as such — in a different font, on gray background`
  - long codes and texts will have their own line(s)
- Results that come out in output files appear as such — in the same font, on green background
  - except very obvious results, such as figures and tables
- Specific sections are highlighted yellow as such for emphasis
  - these could be for anything — codes and texts in input, results in output, and/or texts on slides
- The slides are designed for self-study as much as for the workshop
  - *accessible*, in substance and form, to go through on your own

The Workshop — Aims

- To make you aware what is possible with R Markdown
  - we will cover a large breath of issues, not all of it is for long-term memory
    - one reason why the slides are designed for self study as well
  - awareness of what is possible, `Google`, and perseverance are all we need
- To encourage you to convert into R Markdown
  - practice with a mock manuscript (Parts 3–9)
  - start converting a real one (Part 10)

Part 1. Getting the Tools Ready

Back to the contents slide.

Course Materials — Overview

Materials, available at https://github.com/resulumit/rmd_workshop/tree/materials, have the following structure

```
rmd_workshop-materials
   |
   |- manuscript
   |   |
   |   |- reproduce_this.pdf
   |   |- journals.Rmd
   |   |- references.bib
   |   |- apa_7th.csl
   |
   |- data
   |   |
   |   |- journals.csv
   |
   |- image
   |   |
   |   |- google_scholar.png
```

Course Materials — Download from the Internet

- Download the materials from https://github.com/resulumit/rmd_workshop/tree/materials
  - on the webpage, follow

```
Code -> Download ZIP
```

- Unzip and rename the folder
  - unzip to a location that is not synced
    - e.g., perhaps Documents, but not Dropbox
  - rename the folder as `YOURNAME-rmd`
    - e.g., `resul-rmd`
    - this will come handy in Part 9

Course Materials — Contents

- `manuscript\reproduce_this.pdf`
    - the document, formatted in Word but saved as PDF, that we will re-create with R Markdown
    - randomly generated sentences, with figures and tables from randomly a generated dataset*
    - key sections in need of attention are highlighted yellow

* The text, *Lorem ipsum*, is generated with the `stringi` package (Gagolewski, 2020) while the dataset is created with the `fabricatr` package (Blair et al., 2019).

Course Materials — Contents

- `manuscript\reproduce_this.pdf`
  - the document, formatted in Word but saved as PDF, that we will re-create with R Markdown
  - randomly generated sentences, with figures and tables from randomly generated dataset
  - key sections in-need of attention are highlighted
- `manuscript\journals.Rmd`
  - the R Markdown document that we will work on
  - includes unformatted text from `reproduce_this.pdf` to save time
  - major components, such as paragraphs and tables, are numbered and marked in comments to facilitate navigation
- `manuscript\references.bib`
  - a BibTeX document with three fabricated references
- `manuscript\apa_7th.csl`
  - a Citation Style Language document, with APA (7th Edition) referencing style (Wiernik, 2020)

Course Materials — Contents

`data\journals.csv`

- a dataset created with the `fabricatr` package (Blair et al., 2019), imagined to explore the `Google Scholar` rankings of fictitious journals
- includes the following variables
  - **name**: journals (1090 random titles)
  - **origin**: geographic origins (five continents)
  - **branch**: major discipline of journals (four branches)
  - **since**: time of first publication (years)
  - **h5_index**: H5 Index (integers)
  - **h5_median**: H5 Median (integers)
  - **english**: English (1) *vs.* other-language (0) journals
  - **subfield**: subfield (1) *vs.* generalist (0) journals
  - **issues**: number of issues published per year (integers)

Course Materials — Contents

- `image\google_scholar.png`
  - a screeenshot image of the Google Scholar homapage

Git — Download from the Internet and Install

- For Windows, install 'Git for Windows', downloading from https://gitforwindows.org
  - select 'Git from the command line and also from 3rd-party software'
- For Mac, install 'Git', downloading from https://git-scm.com/downloads

GitHub — Open an Account

Sign up for GitHub at https://github.com

- registering an account is free
- usernames are public
  - either choose an anonymous username (e.g., `asdf029348`)
  - or choose one carefully — it becomes a part of users' online presence
- usernames can be changed later

R and RStudio — Download from the Internet and Install

- Download R from https://cloud.r-project.org
  - choose the version for your operating system
- Download RStudio from https://rstudio.com/products/rstudio/download
  - choose the free version

R Packages — Install from within RStudio

```
install.packages(c("rmarkdown", "tinytex", "dplyr", "stargazer", "ggplot2"))

tinytex::install_tinytex()
```

- `rmarkdown` (Allaire et al., 2020), for automating the process of converting R Markdown documents into other formats

R Packages — Install from within RStudio

```
install.packages(c("rmarkdown", "tinytex", "dplyr", "stargazer", "ggplot2"))

tinytex::install_tinytex()
```

- `rmarkdown` (Allaire et al., 2020), for automating the process of converting R Markdown documents into other formats
- `tinytex` (Xie, 2020), for PDF outputs
  - requires an additional step to install
  - alternative: a TeX/LaTeX system installed on your computer

R Packages — Install from within RStudio

```
install.packages(c("rmarkdown", "tinytex", "dplyr", "stargazer", "ggplot2"))

tinytex::install_tinytex()
```

- `dplyr` (Wickham, 2020a), for data manipulation
  - alternatives: e.g., `base`, `data.table` (Dowle & Srinivasan, 2019)
- `stargazer` (Hlavac, 2018), for tables
  - alternatives: e.g., `knitr` (Xie, 2020b), `kableExtra` (Zhu, 2019)
- `ggplot2` (Wickham, 2020b), for figures
  - alternatives: e.g., `base`, `plotly` (Sievert et al., 2020)

RStudio Project — Create from within RStudio

- RStudio allows for dividing your work with R into separate projects, each with own history etc.
  - this page has more information on why projects are recommended

- Create a new RStudio project for the existing* workshop directory `...\YOURNAME-rmd` from the RStudio menu:

```
File -> New Project -> Existing Directory -> Browse -> ...\YOURNAME-rmd ->
Open
```

* Recall that we have downloaded this earlier from GitHub. Back to the relevant slide.

R Markdown Cheat Sheet — Download from the Internet

Downloading process can be initiated from within RStudio

- follow from the RStudio menu

```
Help -> Cheatsheets -> R Markdown Cheat Sheet
```

Other Resources*

- Pandoc User's Guide
  - available at https://pandoc.org/MANUAL.html
- R Markdown: The Definitive Guide (Xie et al., 2019)
  - open access at https://bookdown.org/yihui/rmarkdown
- R for Data Science (Wickham and Grolemund, 2019)
  - open access at https://r4ds.had.co.nz

* During the workshop, R Markdown Cheat Sheet is likely to be more helpful than these resources, which I recommend to be consulted after the workshop.

Part 2. Introducing R Markdown

Back to the contents slide.

R Markdown Document — Create from within RStudio

- Create a new R Markdown document from the RStudio menu:*

  ```
  File -> New File -> R Markdown -> OK
  ```

- Save your new document:**

  ```
  File -> Save
  ```

- Observe that
  - the document has been saved to your working directory, and
  - it has the .Rmd extension

* This is for demonstration purposes only. Otherwise, we will work with `journals.Rmd`, which you have already downloaded, to save time.

** Alternatively, use the `Save` button or the keyboard shortcut (e.g., `Ctrl + S` on Windows). For shortcuts, follow `Tools -> Keyboard Shortcuts Help` or `Tools -> Modify Keyboard Shortcuts...`.

R Markdown Document — Components

Observe also that the document has three components

- YAML

```yaml
---
title: "Untitled"
output: html_document
---
```

R Markdown Document — Components

Observe also that the document has three components

- YAML
- text

```
---
title: "Untitled"
output: html_document
---
```

```
## R Markdown

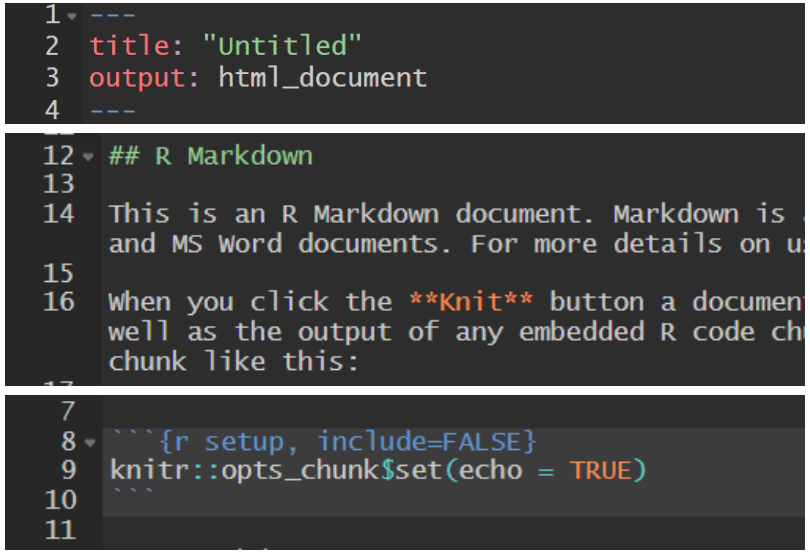
This is an R Markdown document. Markdown is
and MS Word documents. For more details on u

When you click the **Knit** button a documen
well as the output of any embedded R code ch
chunk like this:
```

R Markdown Document — Components

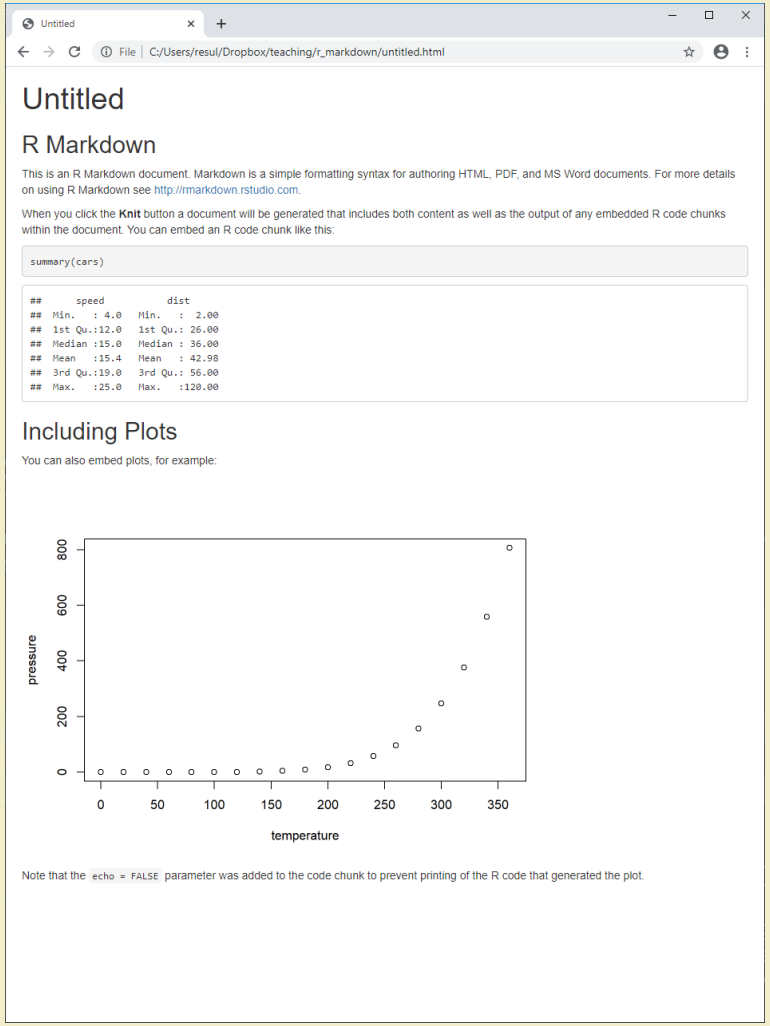
Observe also that the document has three components

- YAML
- text
- code chunks



R Markdown Document — Compile

- Click the `Knit` button to compile it, and observe that
  - the output document has the same name as your .Rmd document
- You may want to delete these newly created files, as we will work with `journals.Rmd` instead to save time.



R Markdown Document — Compilation Process

- When you `Knit`, the following happens:

  ```
  .Rmd --knitr--> .md --pandoc--> output
  ```

  - `knitr`* 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., HTML, LaTeX, PDF, Word
- This process is automated by the `rmarkdown` package

* If you had not already have the `knitr` package, it would have been installed together with the `rmarkdown` package.

** RStudio comes with a copy of `pandoc` (http://pandoc.org), which is not an R package, so that you do not have to install it separately.

R Markdown Document — Notes

- Behind the scenes, each .Rmd file is compiled in its own session, and therefore
  - the code needs to stand alone, for reproducibility reasons
  - you might have already loaded a package and/or imported data elsewhere (e.g., in a separate .R and/or .Rmd file), but even in the same session, these won't be available to a given .Rmd file
- R Markdown can produce more than documents,* including
  - presentations, again with `rmarkdown`
  - books, with `bookdown` (Xie, 2020c)
  - websites, with `blogdown` (Xie, 2020d)

* Here we will focus on research papers only. In a separate workshop, I teach how to create professional websites with R Blogdown.

Part 3. Setting Metadata

Back to the contents slide.

YAML — Overview

.Rmd documents start* with YAML

- it includes the metadata variables
  - e.g., title, output format
- it is written between a pair of three hyphens -

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

* Technically, we can place YAML anywhere in a .Rmd document. However, it is a good practice to start with YAML so that the metadata is easly accessbile.

YAML — Variables

- `title` and `output` are the basic variables of YAML
  - variable names are typed in lower case, followed by a colon :
  - the list of available variables, as well as options and sub-options for these variables, depends on the output format
    - Pandoc User's Guide provides a comprehensive documentation
    - R Markdown Cheat Sheet provides a helpful list
- Typical YAML variables for an research paper:

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

YAML — Variables

Variables can take strings

```
---
title: "Journals: Random Words With Random Data"
output:
---
```

YAML — Variables

Variables can take strings, options

```
---
title: "Journals: Random Words With Random Data"
output: pdf_document
---
```

YAML — Variables

Variables can take strings, options, sub-options

```
---
title: "Journals: Random Words With Random Data"
output:
    pdf_document:
        keep_tex: true
---
```

YAML — Variables

Variables can take strings, options, sub-options, and code

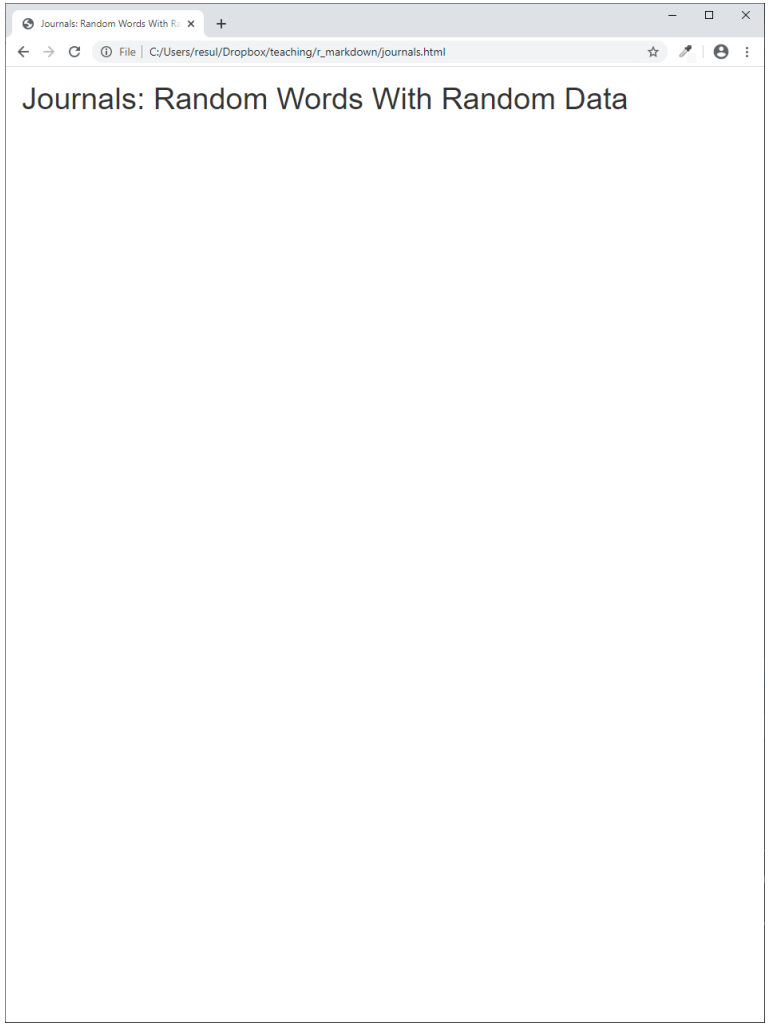
```
---
title: "Journals: Random Words With Random Data"
date: "r format(Sys.Date(), '%d %B %Y')"
output:
    pdf_document:
        keep_tex: true
---
```

YAML — Variables — Output Formats

Documents as output formats include

- HTML

```
---
title: "Journals: Random Words With Random Data"
output: html_document
---
```

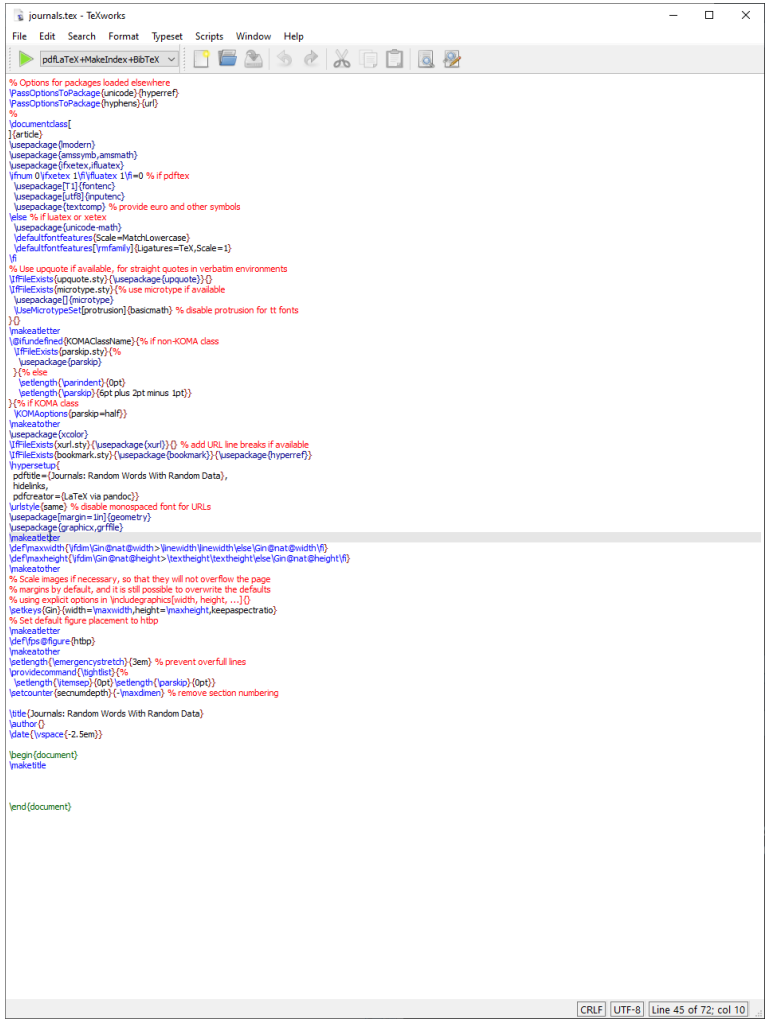


YAML — Variables — Output Formats

Documents as output formats include

- HTML
- LaTeX

```
---
title: "Journals: Random Words With Random Data"
output: latex_document
---
```

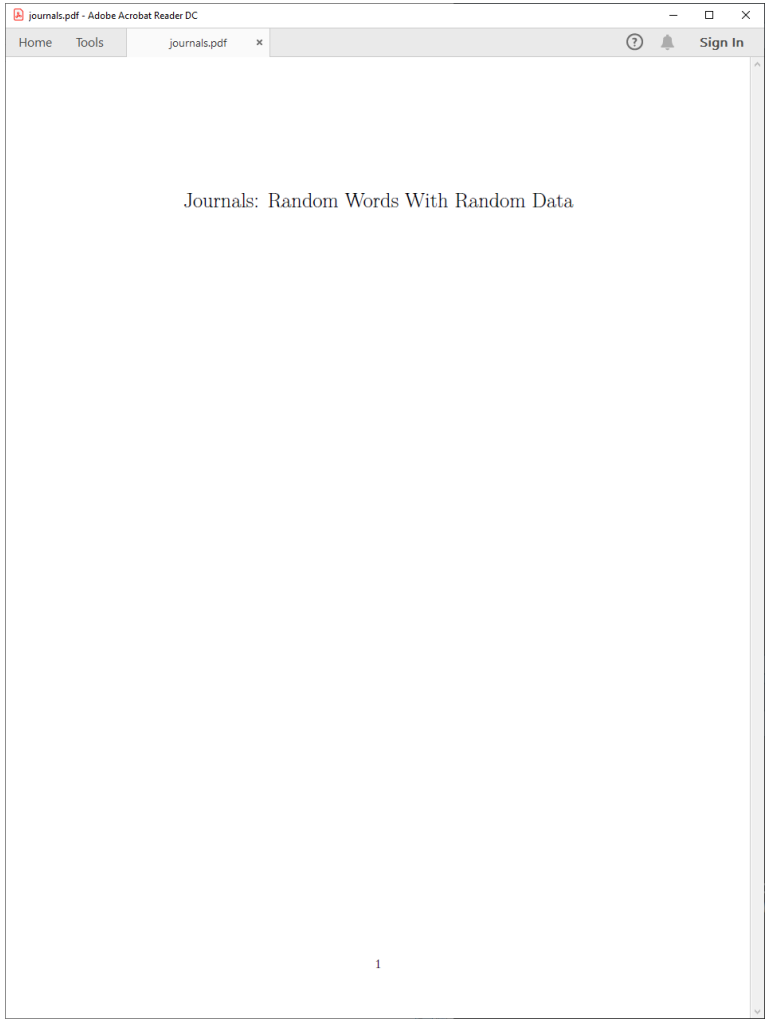


YAML — Variables — Output Formats

Documents as output formats include

- HTML
- LaTeX
- PDF

```
---
title: "Journals: Random Words With Random Da
output: pdf_document
---
```

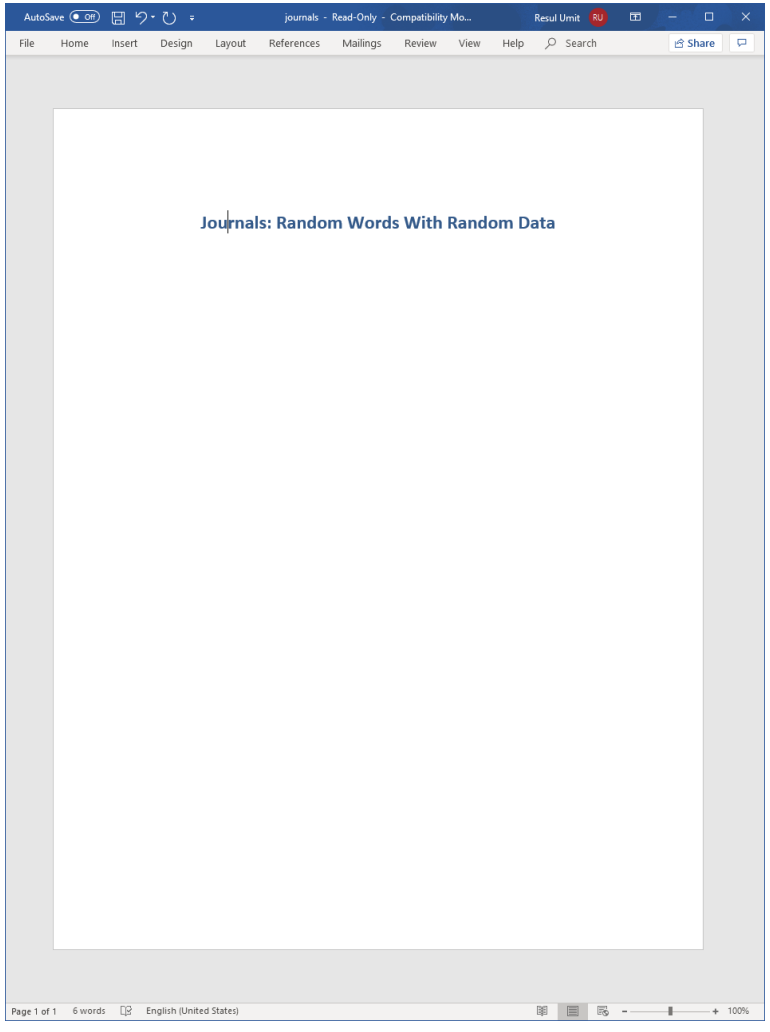


YAML — Variables — Output Formats

Documents as output formats include

- HTML
- LaTeX
- PDF
- Word

```
---
title: "Journals: Random Words With Random Data"
output: word_document
---
```



YAML — Variables — Output Formats

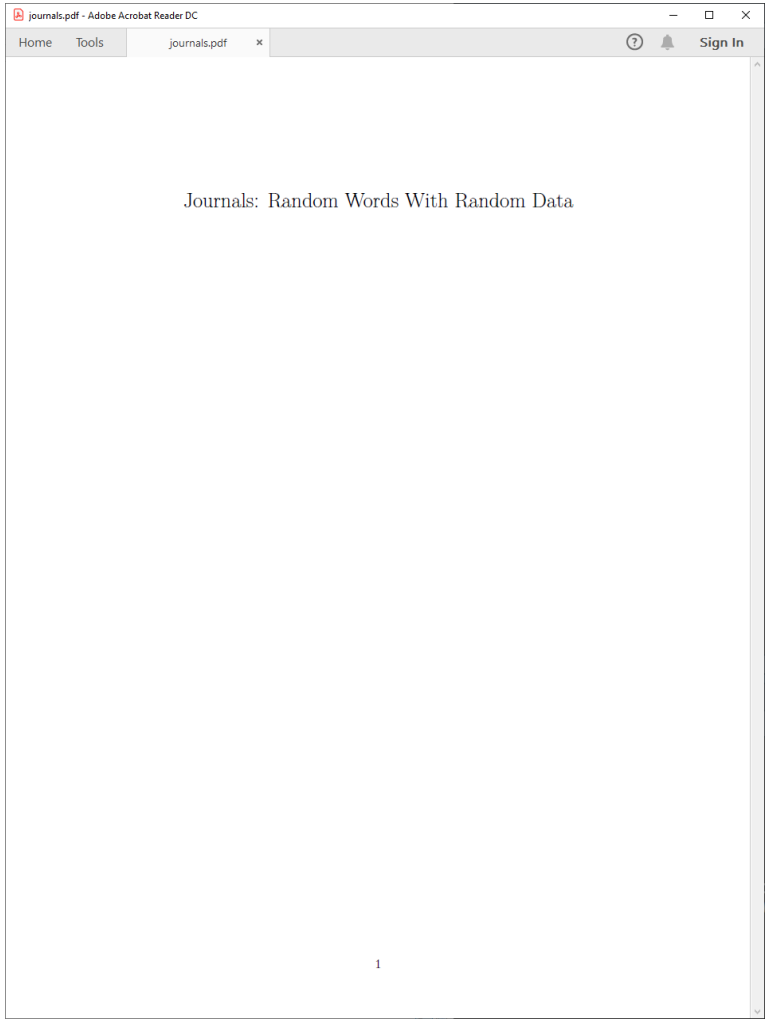
- Documents as output formats
  - `html_document`
  - `latex_document`
  - `pdf_document`*
  - `word_document`
  - `github_document`
  - `md_document`
  - `odt_document`
  - `rtf_document`
- Presentations as output formats
  - `beamer_presentation`
  - `iosslides_presentation`
  - `powerpoint_presentation`
  - `slidy_presentation`

* For reasons of simplicity, this workshop focuses on LaTex and/or PDF outputs. Different output formats have slightly different customisations. See Pandoc User's Guide and/or R Markdown Cheat Sheet.

YAML — Strings

Strings with special characters, such as colon, require quotation marks " or '

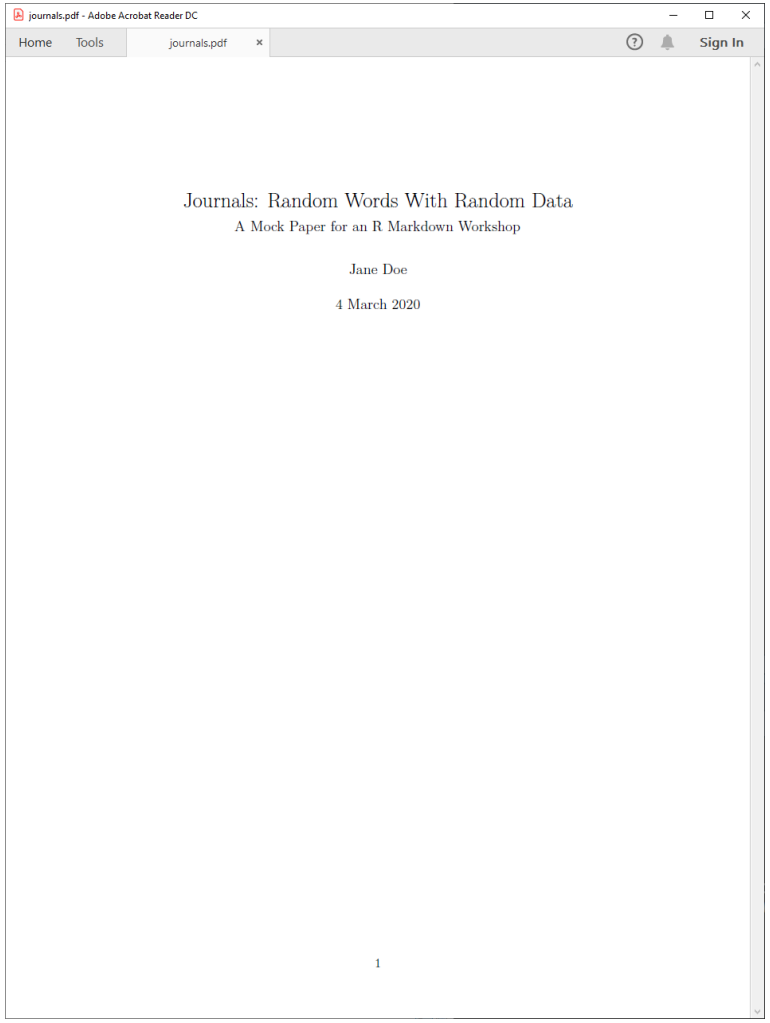
```
---
title: "Journals: Random Words With Random Data"
output: pdf_document
---
```



YAML — Strings

Quotation marks are optional for strings without special characters

```
---
title: "Journals: Random Words With Random Data"
subtitle: A Mock Paper for an R Markdown Workshop
author: Jane Doe
date: 4 March 2020
output: pdf_document
---
```



YAML — Strings

The syntax `^[footnotes_go_here]` adds footnotes to strings

```
---
title: "Journals: Random Words With Random Data^[Prelimi
subtitle: A Mock Paper for an R Markdown Workshop
author: "Jane Doe^[Department of Science, University of
date: 4 March 2020
output: pdf_document
---
```

YAML — Strings

The `bibliography` and `csl` variables take strings as well

```
---
title: "Journals: Random Words With Random Data^[Preliminary draft. Please do not cite or circula
subtitle: A Mock Paper for an R Markdown Workshop
author: "Jane Doe^[Department of Science, University of Random. Email: jane.doe@random.edu. Webs
date: 4 May 2020
bibliography: references.bib
csl: apa_7th.csl
output: pdf_document
---
```

YAML — Strings

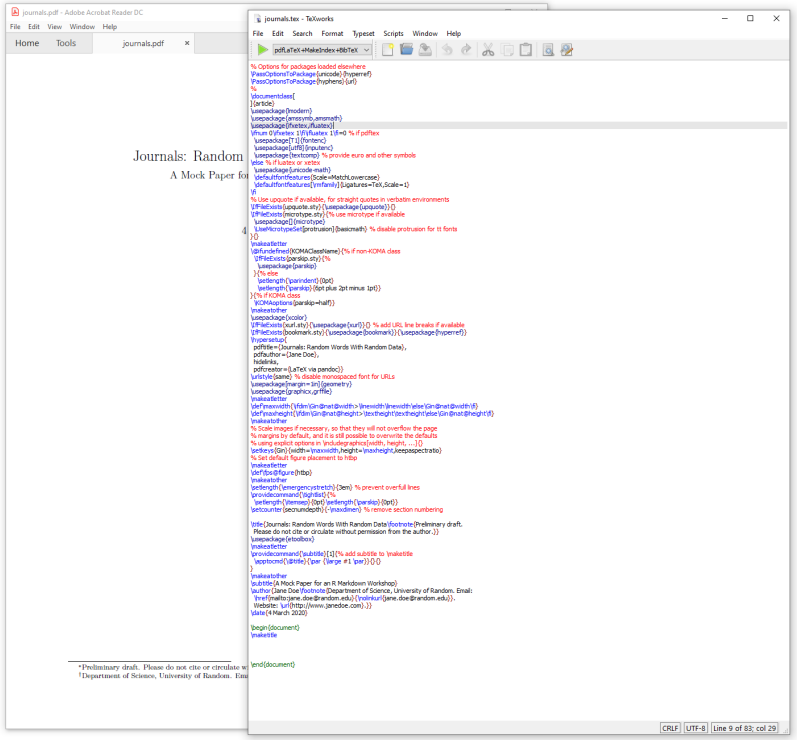
These strings depend on (a) where the files are located and (b) how they are named

```
---
...
bibliography: references/ref_library.bib
csl: "C:/Users/resul/Dropbox/teaching/r_markdown/references/chicago_manual_17.csl"
...
---
```

YAML — Options and Sub-Options

Options can have sub-options

```
---
title: "Journals: Random Words With Random Data^[Prelimi
subtitle: A Mock Paper for an R Markdown Workshop
author: "Jane Doe^[Department of Science, University of
date: 4 March 2020
bibliography: references.bib
csl: apa_7th.csl
output:
    pdf_document:
        keep_tex: true
---
```



YAML — Options and Sub-Options

Options can have sub-options

```
---
title: "Journals: Random Words With Random Da
subtitle: A Mock Paper for an R Markdown Work
author: "Jane Doe^[Department of Science, Uni
date: 4 March 2020
bibliography: references.bib
csl: apa_7th.csl
output:
    pdf_document:
        keep_tex: true
---
```

Notice that

- this specific setting, highlighted, will create multiple outputs
    - a LaTeX and a PDF document
- all but the last option (i.e., `true`) takes a colon
- options and sub-options (except the last option, again) are stepwise indented
    - exactly with four spaces
    - the alignment between the colons for `pdf_document` and `keep_tex` is coincidental

YAML — R Code

Variables can take code as well

```
---
title: "Journals: Random Words With Random Data^[Prelimi
subtitle: A Mock Paper for an R Markdown Workshop
author: "Jane Doe^[Department of Science, University of
date: "r format(Sys.Date(), '%d %B %Y')"
bibliography: references.bib
csl: apa_7th.csl
output: pdf_document
---
```

YAML — R Code

Variables can take code as well

```
---
title: "Journals: Random Words With Random Da
subtitle: A Mock Paper for an R Markdown Work
author: "Jane Doe^[Department of Science, Uni
date: "r format(Sys.Date(), '%d %B %Y')"
bibliography: references.bib
csl: apa_7th.csl
output: pdf_document
---
```

Notice that

- such codes can be particularly useful for variables
  - that need frequent updates
  - and that can be automatically updated
    - e.g., `date`
- there are quotation marks around the code
- we'll cover codes in .Rmd documents later on in the workshop

YAML — R Code

Code and text can be combined in a string

```
---
title: "Journals: Random Words With Random Data^[Prelimi
subtitle: A Mock Paper for an R Markdown Workshop
author: "Jane Doe^[Department of Science, University of
date: "First version: 4 March 2020. This version: r forr
bibliography: references.bib
csl: apa_7th.csl
output: pdf_document
---
```

YAML — Some Further Settings for PDF Outputs

- `fontsize`
  - the default is `10pt`
  - the other options are `11pt` and `12pt`
- `linkcolor`, `urlcolor`, `citecolor`
  - the default is the colour of the text
  - the other options are white, red, green, blue, cyan, magenta, yellow
- `link-citations`
  - the default is `no`
  - the other option is `yes` — a click on an citation will take the screen to the relevant entry in the list of references

Exercises

10:00

1) Open `journals.Rmd` and fill in the YAML variables for the mock paper

- take cues from `reproduce_this.pdf` and/or the slides

2) Add and set one of the variables mentioned as further settings for PDF outputs above

- i.e., `fontsize`, `linkcolor`, `urlcolor`, `citecolor`, `link-citations`

3) Add and set a completely new variable not covered so far

- see, for example, the R Markdown Cheat Sheet

4) `Knit` your `journals.Rmd`

- observe the outcome

Part 4. Writing Text

Back to the contents slide.

Syntax — Overview

- R Markdown follows the syntax in Pandoc's Markdown
  - for the complete rules of the syntax, see Pandoc User's Guide
  - for a useful summary of the syntax, see the R Markdown Cheat Sheet

Syntax — 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.

Syntax — Hard Brakes

Two or more spaces at the end of lines introduce hard brakes, 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.

Syntax — Line Blocks

Spaces in lines starting 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

Syntax — 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

Syntax — 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 second sentence of that paragraph, which is followed 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. This is the second sentence of that paragraph, which is followed by a blank line.

* Multiple blank lines between paragraphs reduce to one.

Syntax — 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 second
sentence of that paragraph, which is followed 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 second sentence of that paragraph, which is followed 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. This is the second sentence of that paragraph, which is followed by a blank line.

Exercises

05:00

5) Hard Brakes

- see `reproduce_this.pdf`: page 1
- apply in `journals.Rmd`: paragraph 1

6) Line Blocks / Block Quotes

- see `reproduce_this.pdf`: page 1
- apply in `journals.Rmd`: block quote, between paragraphs 1 and 2

- see `reproduce_this.pdf`: page 5
- apply in `journals.Rmd`: hypothesis 1, between paragraphs 14 and 15; hypothesis 2, between paragraphs 16 and 17

Syntax — 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

Syntax — 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

Syntax — 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

Exercises

03:00

7) Headers

- see `reproduce_this.pdf`: pages 1 to 11
  - 11 headers, Abstract to References
- apply in `journals.Rmd`

8) Emphases

- see `reproduce_this.pdf`: pages 1 and 2
  - bold and italics
- apply in `journals.Rmd`: paragraph 2

Syntax — Links — Internal*

You can link text to section headers in the same document

`[Conclusion](#conclusion)` becomes Conclusion, and a click takes the screen to that section

Multi-word headers need hyphenation

`[Literature Review](#literature-review)` becomes Literature Review, and it works only if the second part is hyphenated

* The links to references, figures, and tables are covered later on.

Syntax — Links — External

You can link text to URLs

`[visit my website](https://resulumit.com/)` becomes visit my website

`[https://resulumit.com](https://resulumit.com/)` becomes https://resulumit.com

`<https://resulumit.com>` becomes https://resulumit.com as well

You can also link text to an email address

`[email me](mailto:resuluy@uio.no)`* becomes email me

`<resuluy@uio.no>` becomes resuluy@uio.no

* Notice the prefix mailto: in the syntax

Exercises

03:00

9) Links — Internal

- see `reproduce_this.pdf`: page 2
  - the link to the Literature Review section
- apply in `journals.Rmd`: paragraph 4

10) Links — External

- see `reproduce_this.pdf`: page 1
  - email and website links in one of the footnotes
- apply in `journals.Rmd`: title page items

Syntax — 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 — with or without spaces, it works

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

$$E = mc^2$$

`$$E = mc_{2} $$` becomes

$$E = mc_2$$

Syntax — Footnotes — Inline Notes

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

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

Notice that

- the caret sign ^ comes before the left square bracket [
- this syntax works in YAML as well as in text
  - footnotes in YAML get symbols, in text they get numbers

[1] Corresponding author.

Syntax — Footnotes — Notes with Identifiers

An alternative is to use the `[^identifier]` syntax, with identifiers defined elsewhere in the same document

```
Dr Doe holds a PhD in rock science.[^defence_date]

[^defence_date]: She defended her thesis in 2017.
```

Dr Doe holds a PhD in rock science.[1]

Notice that

- the caret sign comes after the left square bracket
- this syntax works in text, but not in YAML

[1] She defended her thesis in 2017.

Exercises

03:00

11) Equations

- see `reproduce_this.pdf`: page 7
- apply in `journals.Rmd`: paragraph 22; block equation, between paragraphs 22 and 23

12) Footnotes

- see `reproduce_this.pdf`: page 2
- apply in `journals.Rmd`: paragraph 3

Syntax — Lists

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

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

- books
- articles
- reports

Syntax — Lists — Nesting

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

Syntax — Lists — Numbering

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

Syntax — Dashes

Two hyphens grouped together introduce an en-dash

`--` becomes –

Three hyphens grouped together introduce an em-dash

`---` becomes —

Syntax — Subscript and Superscript

A pair of tildes introduces subscript

`CO~2~` becomes CO_2

A pair of carets introduces superscript

`R^2^` becomes R^2

Syntax — Sub- and Super-scripts

A pair of tildes introduces subscript

`CO~2~` becomes CO_2

A pair of carets introduces subscript

`R^2^` becomes R^2

Notice that

- the syntax here (Markdown-based) is different than the one for equations (LaTeX-based)
  - e.g., `R^2^` versus `mc^{2}`

Exercises

03:00

13) Lists

- see `reproduce_this.pdf`: page 3
- apply in `journals.Rmd`: list, between paragraphs 10 and 11

14) Dashes

- see `reproduce_this.pdf`: page 2
- apply in `journals.Rmd`: paragraph 6

15) Sub- and Super-scripts

- see `reproduce_this.pdf`: page 2
- apply in `journals.Rmd`: paragraph 5

Part 5. Managing References

Back to the contents slide.

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

- `pandoc` can process a citation only if there is a linked entry in the .bib file
  - but not all entries have to be cited

References — Bibliography Database — Entries

- A BibTeX entry consists of three elements
  - a type
    - e.g., `@article`
  - a citation-key
    - e.g., `bennett2015`
  - a number of tags
    - e.g., `title`, `author`
- Different tags are available for different reference types
  - some tags are required, others are optional

```
@article{bennett2015,
  title={Peanut butter and jelly},
  author={Bennett, Stephanie},
  journal={Journal of Bone},
  year={2015}
  volume={1},
  number={12},
  pages={3--35},

}
```

References — Bibliography Database — Entries

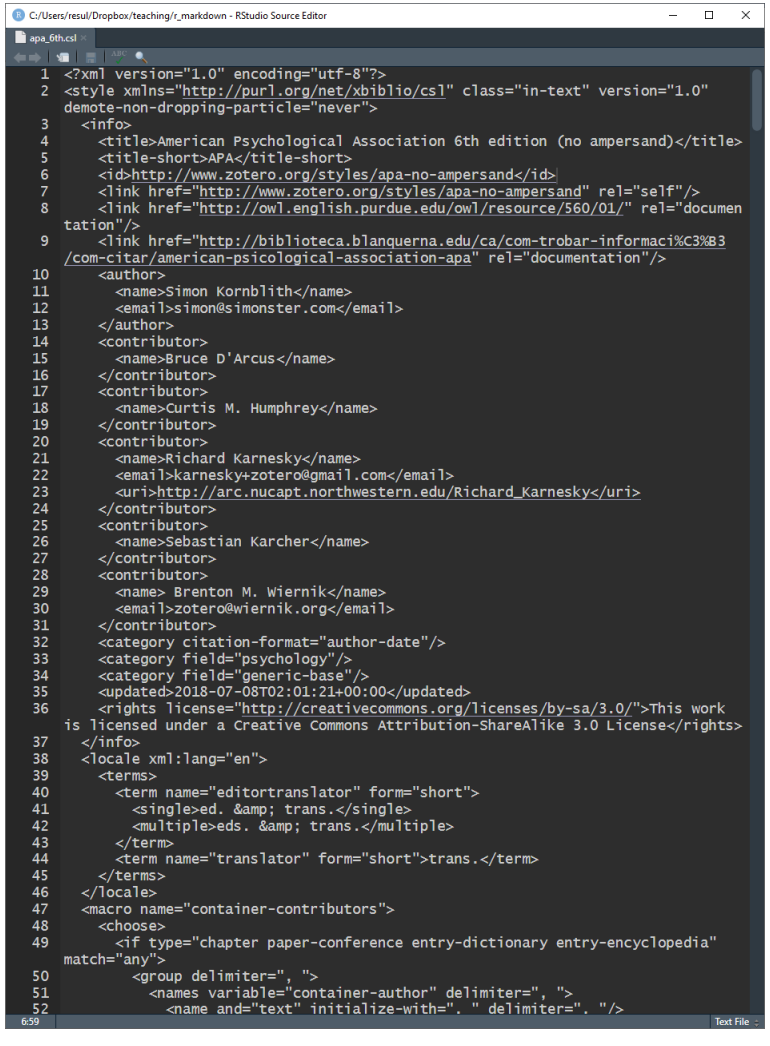
- One could create entries by hand
  - this requires knowing the BibTeX format, entry types, tags, and related information about references to be cited
    - this is not efficient
- A good alternative is to use `Google Scholar`, which provides BibTeX entries
  - follow `cite -> BibTex` and copy
  - paste into .bib, edit if necessary, and save
- Some publishers and journals provide BibTeX entries on their website as well

References — Style

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

- `pandoc` looks for a .csl file, and for the styles therein, to style citations and references
  - .csl files are specified with the `csl` variable in YAML
  - if unspecified, it uses a Chicago author-date format

- .csl files affect the style only in outputs
  - no matter which the style is used, the citation syntax in .Rmd documents remains the same



References — In-text Citation Syntax — Author-Date

All citations keys take the 'at' sign @ while square brackets and/or minus signs introduce variation

`[@bennett2015]` becomes (Bennett, 2015) in APA 7th edition

`@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)

References — In-text Citation Syntax — Numerical

All citations keys take the 'at' sign @

`A clever sentence.[@bennett2015]` becomes A clever sentence.[1] in certain numerical sytles

`A clever sentence.[@bennett2015; @gilbert2019]` becomes A clever sentence.[1,2]

Individual styles may or may not use additional information, such as page numbers

`A clever sentence.[@bennett2015 35]` might become A clever sentence.[1] as well

Individual styles may or may not be sensitive to variation, such as square brackets

`A clever sentence. @bennett2015` might become A clever sentence.[1] as well

Citations — Reference List

The list of references appears after the last line of the document, with no section header

- so that you can choose the header yourself, by ending .Rmd documents with a header of your choice

```
This is the last sentence of a manuscript with an APA style.

## References
```

This is the last sentence of a manuscript with an APA style.

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.

References — Internal Links

For internal links from in-text citations to the reference list, set `link-citations: yes` in YAML

- a click on these links takes the screen to the relevant entry in the list
- the `linkcolor` variable make these links explicit
  - setting this is not necessary for the links to work — the default is black

```
---
...
bibliography: references.bib
csl: apa_7th.csl
link-citations: yes
linkcolor: blue
...
---
```

Exercises

07:30

16) Add an entry to `references.bib` for the following book

- *R Markdown: The Definitive Guide* by Xie and co-authors

17) Reproduce the citations and reference list in the mock paper

- see `reproduce_this.pdf`: pages 3 and 11
- apply in `journals.Rmd`: paragraph 7 to 9

18) Change the reference style

- download the .csl file for your favourite style from https://www.zotero.org/styles
- put it into your working directory
- update the YAML variable

19) Link the citations to the reference list

Part 6. Adding Code, Figures, and Tables

Back to the contents slide.

Code, in and outside Chunks

Code — Overview

Most codes go inside code chunks

- e.g., code that imports and cleans data, and/or produces tables and/or figures

````
```{r}

df <- read.csv("rmd_workshop_files/images_data/journals.csv") %>%
      mutate(age = 2020 - since,
             english = factor(english),
             subfield = factor(subfield))

```
````

Codes can also go in line with text

- e.g., code that results in a single statistic

````
The average H5 Index for the journals in the dataset is `r mean(df$h5_index)`.
````

Code Chunks — Overview

- Code chunks are delimited spaces between a pair of three backticks
  - below is an empty chunk for R code

````
```{r}

```
````

- `r` is an option of the chunk, indicating that the code in the chunk above should be run by R
  - it could have been `python`, which we will not cover in this workshop
  - options go in a pair of curly brackets, on the same line with the first delimiter
- Chunks can be placed anywhere in .Rmd documents
  - their output, if there is any, might float around text to avoid breaking across pages

Code Chunks — Labels

It is recommended to label the code chunks, which are otherwise automatically numbered

- informative labels can be helpful for navigating through error messages as well as filenames of plots and cache
  - in the example below, the chunk is labelled as `data_import`
- but note that duplicate labels lead to an error during compilation

````
```{r, data_import}

df <- read_csv("data/journals.csv")

```
````

Code Chunks — Options

- Code chunks take options, listed on the same line with the first delimiter, in curly brackets
  - avoid spaces around the equal sign = between option tags and values
    - such spaces might lead to errors
  - in the example below, the chunk is labelled as `setup`, and and the `include` option is set to `FALSE`
    - with this option and value, nothing from this chunk will be included in the output document

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

- The list of options is available at https://yihui.org/knitr/options
  - R Markdown Cheat Sheet provides a helpful list as well

Code Chunks — Options — Defaults

Options have default values

- e.g., for `echo`, the default is `TRUE`
  - `echo`: should the source code printed in the output?
  - `TRUE`: yes it should

- therefore the following two chunks have the same function

````
```{r}
```
````

````
```{r, echo=TRUE}
```
````

Code Chunks — Options — Defaults

This chunk prints two things in the output document — (a) the code and (b) the head of the data frame

````
```{r}
head(df)
```
````

```
head(df)
```

```
##                name   origin   branch h5_index h5_median english subfield issues age
## 1  Journal of Bears Americas Physical       73        97       1        1      7  61
## 2   Journal of Moon     Asia   Social       72       106       1        0      6  64
## 3 Journal of Lumber Americas Physical       72       100       1        1      8  30
## 4 Journal of Houses   Europe   Social       72       102       1        0      8  38
## 5  Journal of Water   Europe   Social       70       100       1        0      5  33
## 6  Journal of Jeans Americas Physical       69       101       1        1      5  64
```

Code Chunks — Options

Setting `echo=FALSE` prevents the code from being displayed in the output document

````
```{r ... echo=FALSE}

head(df)

```
````

This chunk therefore prints one thing in the output document — the head of the data frame

```
##                name   origin   branch h5_index h5_median english subfield issues age
## 1  Journal of Bears Americas Physical       73        97       1        1      7  61
## 2   Journal of Moon     Asia   Social       72       106       1        0      6  64
## 3 Journal of Lumber Americas Physical       72       100       1        1      8  30
## 4 Journal of Houses   Europe   Social       72       102       1        0      8  38
## 5  Journal of Water   Europe   Social       70       100       1        0      5  33
## 6  Journal of Jeans Americas Physical       69       101       1        1      5  64
```

Code Chunks — Other Useful Options

Prevent the result(s) of the source code from being displayed in the output document

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

This chunk therefore prints one thing in the output document — the source code

```
head(df)
```

Setting `results="asis"` passes the results as they are produced by the code — `pandoc` does not transform these. In creating tables for PDF output with the `stargazer` package, this option is a must.

Code Chunks — Other Useful Options

Cache results for future compilations

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

Prevent R from running the code in the chunk altogether

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

Prevent messages and/or warnings from being displayed in the output

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

Code Chunks — Other Useful Options

Define the actual dimensions of figures, in inches

```
```{r ... fig.height=6, fig.width=9}
```
```

Define the size of figures as they appear in the output document, with `out.width` and/or `out.height`

```
```{r ... out.width="50%"}
```
```

Define the alignment of figures — `left`, `right`, or `center`

```
```{r ... fig.align="center"}
```
```

Code Chunks — Other Useful Options

Define captions for figures

```
```{r ... fig.caption="A Scatter Plot"}
```
```

Set the resolution for figures

```
```{r ... dpi=300}
```
```

Set extra options, such as angle, that output format would accept for figures

```
```{r ... out.extra="angle=45"}
```
```

Code Chunks — 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()`
  - avoids repeating chunk options
- 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)

# data
df_raw <- read.csv("journals.csv")

```
````

Code Chunks — The Data Chunk

I recommend using the second chunk for the main operations* on raw data

- e.g., for data cleaning and other transformations
- some minor transformations could be left to lower chunks
  - e.g., capitalizing variable names for figures

````
```{r, data, echo=FALSE ...}

df <- df_raw %>%
      ...

```
````

* I will be using the pipe operator `%>%` and other functions from the `dplyr` package for such operations in the following slides.

Code Chunks — The Data Chunk

Transform `subfield` and `english` into factor variables

- despite being numeric — i.e., 0s and 1s — they are meant to be categorical variables

````
```{r, data, echo=FALSE ...}

df <- df_raw %>%
      mutate(subfield = as.factor(subfield),
             english = as.factor(english))

```
````

Code Chunks — The Data Chunk

Create a new variable `age`, based on the existing variable `since`, to be used in regression models later on

````
```{r, data, echo=FALSE ...}

df <- df_raw %>%
      mutate(subfield = as.factor(subfield),
             english = as.factor(english),
             age = 2020 - since)

```
````

Drop the `since` variable, so that it won't appear in our summary statistics table later on

````
```{r, data, echo=FALSE ...}

df <- df_raw %>%
      mutate(subfield = as.factor(subfield),
             english = as.factor(english),
             age = 2020 - since) %>%
      select(-since)
```
````

Inline Code — Overview

Code can also be incorporated in text, with the `` `r ` `` syntax

- unlike chunks, these do not take options
- the output document will display the result of the code
  - in the exact place of the source code
- the result of the code will have the same formatting with the text

Inline Code — Examples

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

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

```
The average H5 Index for the journals in the dataset is `r mean(df$h5_index)`, which would
round to `r round(mean(df$h5_index), digits = 1)`.
```

The average H5 Index for the journals in the dataset is 26.3611366, which would round to 26.4.

```
__Only `r nrow(subset(df, english == 0))` journals__ in the dataset are published in a language
other than English.
```

Only 113 journals in the dataset are published in a language other than English.

Exercises

07:30

20) Setup Chunk

- introduce a setup chunk with one or more defaults chunk options, with `knitr::opts_chunk$set()`
- load the packages that we will need — `dplyr`, `ggplot2`, and `stargazer`
- import raw data

21) Data Chunk

- introduce a data chunk to transform `subfield` and `english` into factors
- create a new variable `age`, based on `since`
- drop `since` from the data frame

22) Inline code

- see `reproduce_this.pdf`: page 6
  - i.e., 1091 observations

- apply in `journals.Rmd`: paragraph 21
  - hint: use the `nrow` function

Figures

Figures — Images — Markdown Syntax

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

- similar to the link syntax, only this time it is preceded by an exclamation mark !
- goes outside code chunks, on a new line
- simple, but not very customisable

* Ideally, reproducible papers should produce their own images with data and code. However, there might be situations where this is not possible.

Figures — Images — Markdown Syntax

```
![A screenshot of the Google Scholar homepage](../image/google_scholar.png)
```

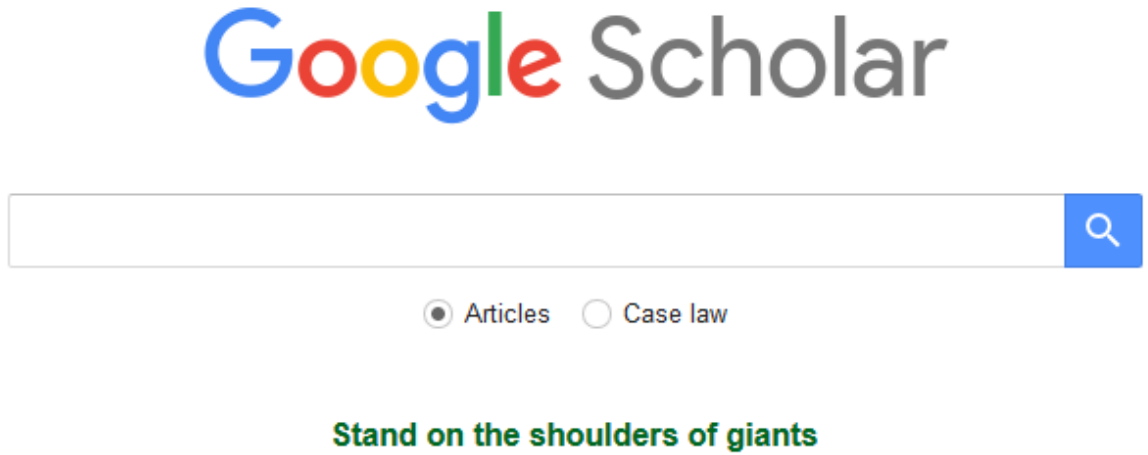


Figure 1: A screenshot of the Google Scholar homepage.

Figures — Images — Markdown Syntax

Figures are numbered automatically

```
![A screenshot of the Google Scholar homepage](../image/google_scholar.png)
```

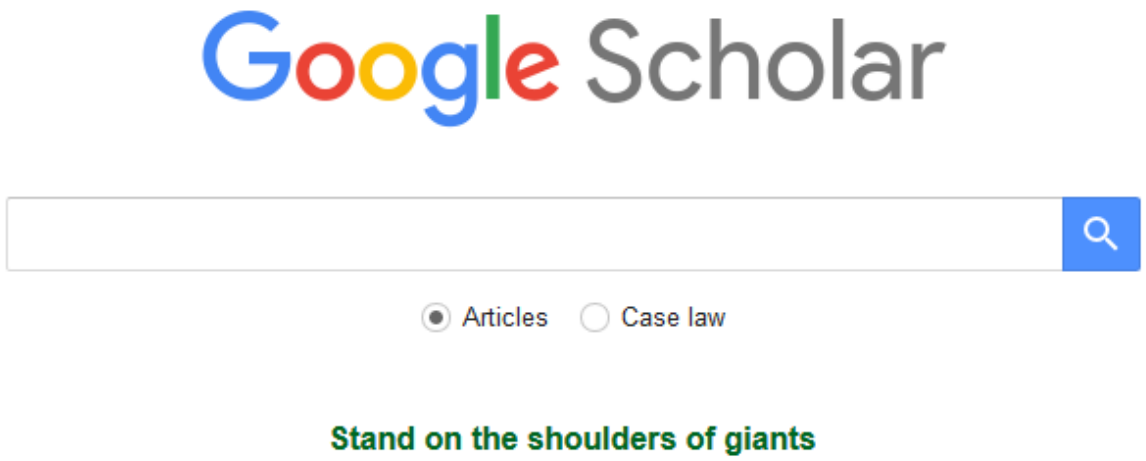


Figure 1: A screenshot of the Google Scholar homepage.

Figures — Images — Markdown Syntax

The syntax can accept `width` or `height` attributes as follows

```
![A screenshot of the Google Scholar homepage](../image/google_scholar.png){ width=40% }
```

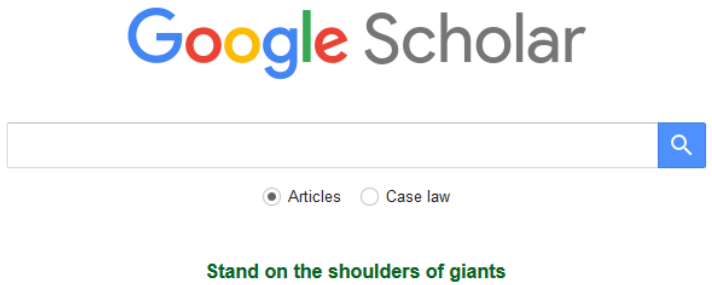


Figure 1: A screenshot of the Google Scholar homepage.

Figures — Images — `knitr`

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

- this goes inside code chunks
  - use the function with the double-colon operator `::`
    - e.g., `knitr::include_graphics("figure.extension")`

- this is more customisable, through the use of code chunks
  - size is defined with the `out.width` or `out.hight` options
    - rather than `fig.height` and/or `fig.width`

Figures — Images — `knitr`

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

````
```{r, screenshot, echo=FALSE, fig.cap="A screenshot of the Google Scholar homepage."}
knitr::include_graphics("../image/google_scholar.png")
```
````

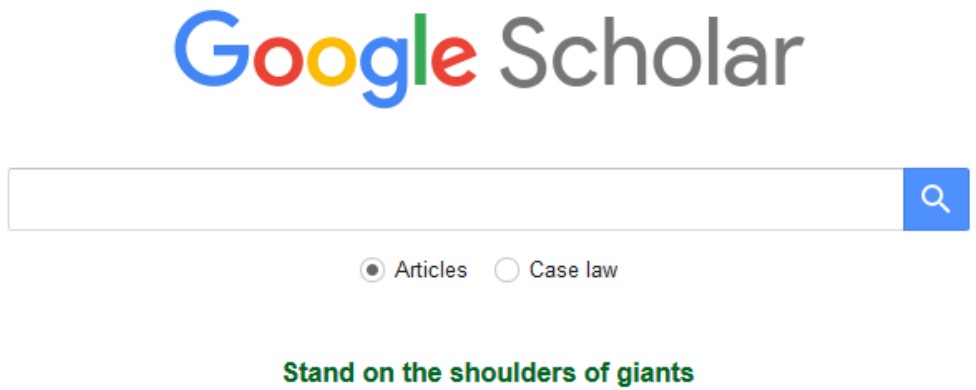


Figure 1: A screenshot of the Google Scholar homepage.

Figures — Images — `knitr`

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

````
```{r ... out.width="40%"}
knitr::include_graphics("../image/google_scholar.png")
```
````

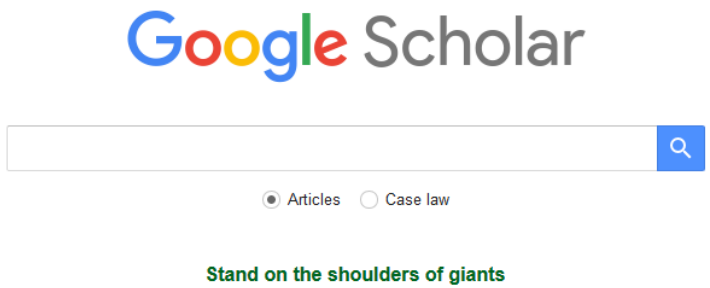


Figure 1: A screenshot of the Google Scholar homepage.

Figures — Images — `knitr`

Most other chunk options are common with figures plotted within R Markdown, such as `fig.align`

```
```{r ... fig.align="center"}
knitr::include_graphics("../image/google_scholar.png")
```
```

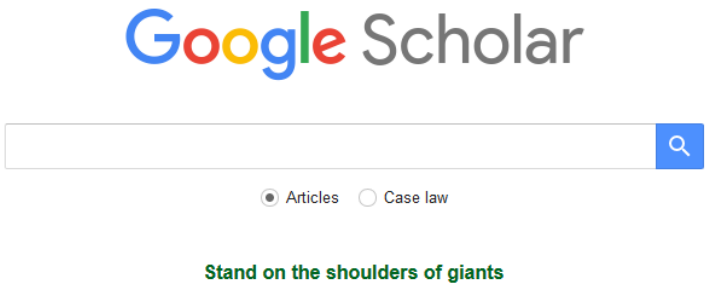


Figure 1: A screenshot of the Google Scholar homepage.

Exercise

03:00

23) Images

- see `reproduce_this.pdf`: figure 1 on page 10
- apply in `journals.Rmd`: figure 1, between paragraphs 19 and 20

Figures — `ggplot2` — Overview

- A powerful package for visualising data
- Used widely, not only by academics, but also by large corporations such as the New York Times
- A huge amount is written on this package. See, for example,
  - the package documentation
  - this book by its creator Hadley Wickham
  - this reference page
  - this webinar by one of its authors, Thomas Lin Pedersen
  - these extensions, maintained by the `ggplot2` community
- Among its alternatives are the `base` and `plotly` packages

Figures — `ggplot2` — Basics

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

- specify a data frame in the main `ggplot` function

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

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

- specify as an additional argument in the same `ggplot` function

```
ggplot(data = df, mapping = aes(x = h5_median, y = h5_index, color = subfield))
```

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

- specify, after a plus sign `+`, as an additional function

```
ggplot(data = df, mapping = aes(x = h5_median, y = h5_index, color = subfield)) +
       geom_point()
```

Figures — ggplot2

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

```
```{r, scatterplot, fig.cap = "A scatterplot of journal

ggplot(data = df, mapping = aes(x = h5_median, y = h5_ir
       geom_point()

```
```

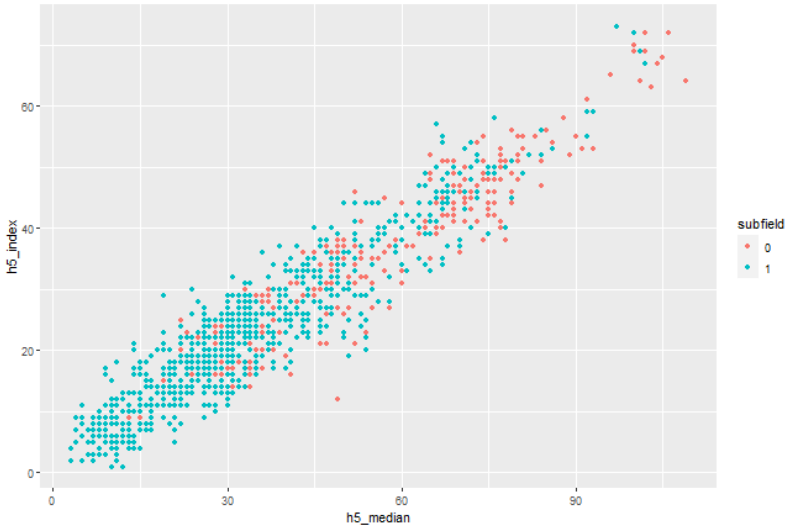


Figure 1. A scatterplot of journal metrics.

Figures — ggplot2

Add facets for subgroups, e.g., branch

```
```{r, scatterplot, fig.cap = "A scatterplot of journal

ggplot(data = df, mapping = aes(x = h5_median, y = h5_ir
       geom_point() +
       facet_wrap(. ~ branch)

```
```

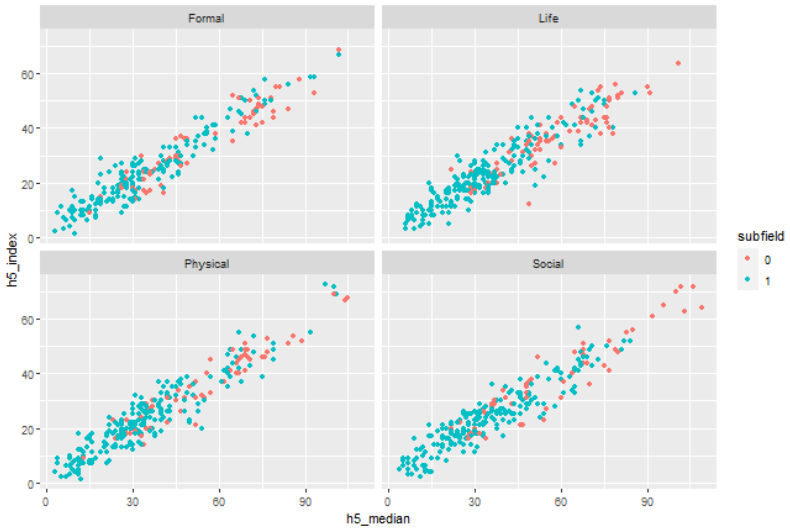


Figure 1. A scatterplot of journal metrics.

Figures — ggplot2

Scale the colour to improve the legend

```
```{r, scatterplot, fig.cap = "A scatterplot of journal

ggplot(data = df, mapping = aes(x = h5_median, y = h5_ir
       geom_point() +
       facet_wrap(. ~ branch) +
       scale_colour_discrete(name = "Journal Type", brea

```
```

Figure 1. A scatterplot of journal metrics.

Figures — ggplot2

Change the theme

```
```{r, scatterplot, fig.cap = "A scatterplot of journal

ggplot(data = df, mapping = aes(x = h5_median, y = h5_ir
       geom_point() +
       facet_wrap(. ~ branch) +
       scale_colour_discrete(name = "Journal Type", brea
       theme_bw()


```
```

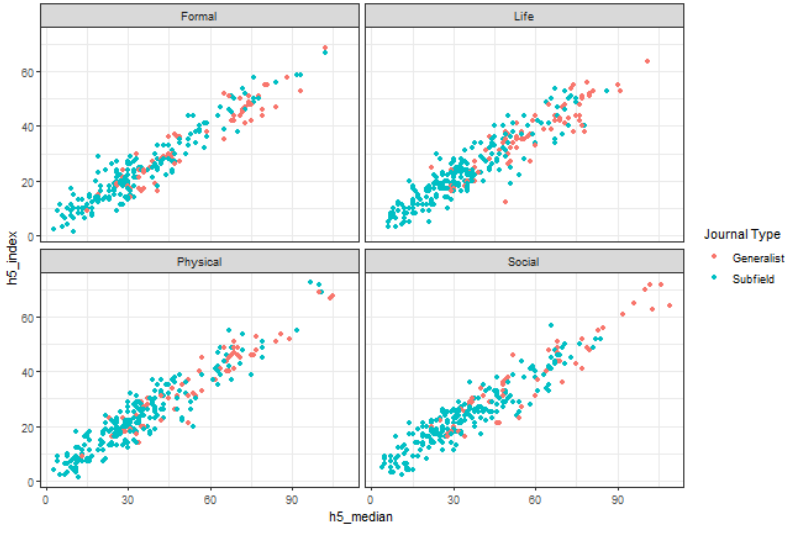


Figure 1. A scatterplot of journal metrics.

Figures — ggplot2

Improve the axis labels, e.g., with capital first letters

```
```{r, scatterplot, fig.cap = "A scatterplot of journal

ggplot(data = df, mapping = aes(x = h5_median, y = h5_ir
       geom_point() +
       facet_wrap(. ~ branch) +
       scale_colour_discrete(name = "Journal Type", brea
       theme_bw() +
       labs(x = "H5 Median", y = "H5 Index")

```
```

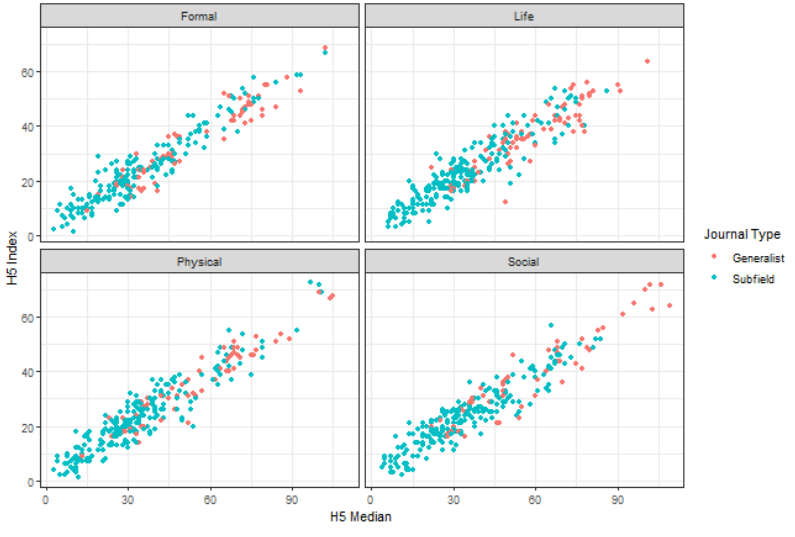


Figure 1. A scatterplot of journal metrics.

Figures — `ggplot2` — Notes

`geom_point` is one of many geoms avilable

- see this https://ggplot2.tidyverse.org/reference for other options, including
  - `geom_bar` for bar charts
  - `geom_boxplot` for box and whiskers plots

Exercises

10:00

24) Barplot

- see `reproduce_this.pdf`: figure 2 on page 7
- apply in `journals.Rmd`: figure 2, between paragraphs 21 and 22

25) Scatterplot

- see `reproduce_this.pdf`: figure 3 on page 9
- apply in `journals.Rmd`: figure 3, between paragraphs 27 and 28

Tables

Tables — Markdown Syntax

The following syntax, outside code chunks, introduces tables that pandoc can recognise

```
First Column  Second Column
------------  -------------
First cell    First cell
Second cell   Second cell
Third cell    Third cell
```

First Column	Second Column
First cell	First cell
Second cell	Second cell
Third cell	Third cell

Tables — Markdown Syntax

The position of headers, relative to their line underneath, defines column alignments

```
Left-Aligned          Centered
---------------  ----------------
First cell        First cell
Second cell       Second cell
Third cell        Third cell
```

Left-Aligned	Centered
First cell	First cell
Second cell	Second cell
Third cell	Third cell

Tables — Markdown Syntax

A line starting with a colon, placed before or after tables, introduces captions

```
    Centered          Right-Aligned
----------------  ----------------
First cell        First cell
Second cell       Second cell
Third cell        Third cell

: A hand-made table with R Markdown
```

Table 1: A hand-made table with R Markdown

Centered	Right-Aligned
First cell	First cell
Second cell	Second cell
Third cell	Third cell

Tables — Markdown Syntax

The caption line itself needs to be surrounded by empty lines

```
    Centered          Right-Aligned
----------------  ----------------
First cell        First cell
Second cell       Second cell
Third cell        Third cell

: A hand-made table with R Markdown

```

Table 1: A hand-made table with R Markdown

Centered	Right-Aligned
First cell	First cell
Second cell	Second cell
Third cell	Third cell

Tables — Markdown Syntax

Tables are numbered automatically

```
: A hand-made table with R Markdown

    Centered         Right-Aligned
----------------  ----------------
First cell        First cell
Second cell       Second cell
Third cell        Third cell
```

Table 1: A hand-made table with R Markdown

Centered	Right-Aligned
First cell	First cell
Second cell	Second cell
Third cell	Third cell

Tables — Markdown Syntax

Grid tables, with the following syntax, can handle complex cells with multiple lines and/or lists

```
+--------------------+--------------------+
| First Column       | Second Column      |
+====================+====================+
| - First item       | First cell         |
| - Second item      |                    |
| - Third item       |                    |
+--------------------+--------------------+
|Second cell         | Second cell with a |
|                    | long text          |
+--------------------+--------------------+
| Third cell         | Third cell         |
|                    |                    |
+--------------------+--------------------+

: A grid table with multi-line cells
```

Table 1: A grid table with multi-line cells

First Column	Second Column
- First item - Second item - Third item	First cell
Second cell	Second cell with a long text
Third cell	Third cell

Tables — Markdown Syntax

Grid tables can be aligned as well, with colons at the boundaries of the header separator*

```
+--------------------+--------------------+
| Left-Aligned       | Centered           |
+:===================+:==================:+
| - First item       | First cell         |
| - Second item      |                    |
| - Third item       |                    |
+--------------------+--------------------+
|Second cell         | Second cell with a |
|                    | long text          |
+--------------------+--------------------+
| Third cell         | Third cell         |
|                    |                    |
+--------------------+--------------------+

: A grid table with multi-line cells
```

Table 1: A grid table with multi-line cells

Left-Aligned	Centered
- First item - Second item - Third item	First cell
Second cell	Second cell with a long text
Third cell	Third cell

* Use `:=` for left-aligned, `:=:` for centered, `=:` for right-aligned columns.

Exercise

05:00

26) Markdown Tables

- see `reproduce_this.pdf`: table 1 on page 4
- apply in `journals.Rmd`: table 1, between paragraphs 11 and 12

Tables — `stargazer` — Overview

- A capable package for creating tables to present
  - data in columns and rows
  - descriptive/summary statistics
  - regression models
- Used widely by academics
- Creates LaTeX code, HTML/CSS code, and ASCII text to be knitted
- A lot is written on this package. See, for example,
  - the package documentation
  - this vignette by its author Marek Hlavac
  - this tutorial by Jake Russ

Tables — `stargazer` — Notes

- We must
  - set the chunk option `results="asis"` for chunks with stargazer tables
  - change the argument `type` in the `stargazer()` function for different output formats accordingly
    - e.g., the default `type = "latex"` is for LaTeX and PDF, `type = "html"` for HTML
- It is currently not quite possible to `knit` `stargazer` code into tables in Word documents
  - `stargazer` tables will not appear in Word documents automatically
  - workarounds available
    - `knit` to HTML as well as Word, copy the tables from HTML to Word
    - `knit` to PDF, open the PDF in Word
      - in Word, follow `File -> Open`
- `stargazer` tables might look slightly different in different output formats
  - on the following slides, they will have the HTML look

Tables — `stargazer` — Basics

- The `stargazer()` function
  - this is probably the only fuction you will ever use from this package
    - but it accepts many, many arguments to customise tables
- The `data` argument of that function, with two main options
  1. a data frame for data or summary statistics tables
     - e.g., `df`, here coming from `df <- read_csv(journals.csv)`
  2. one or more regression models for regression tables
     - e.g., `lm1`, here coming from `lm1 <- lm(h5_index ~ issues, data = df)`

Tables — `stargazer` — Data Tables

Table the first four rows of the dataset

````
```{r, data_table, echo=FALSE, results="asis"}
stargazer(data = head(df, n = 4), type = "latex", summary = FALSE)
```
````

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

- with `echo=FALSE`, the code will not be displayed in the output document
- with `results="asis"`, `knitr` will pass through results without reformatting them
  - these results are produced in LaTeX, due to `type = "latex"`
  - they should remain LaTeX because our outcome document is PDF, converted from LaTeX
- with `summary = FALSE`, the table will present the data, not its descriptive statistics

Tables — `stargazer` — Data Tables

Table the first four rows of the dataset

```
```{r, data_table, echo=FALSE, results="asis"}

stargazer(data = head(df, n = 4), type = "latex", summary = FALSE)

```
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Apr 10, 2020 - 12:31:21

Table 1:

	name	origin	branch	h5_index	h5_median	english	subfield	issues	age
1	Journal of Bears	Americas	Physical	73	97	1	1	7	61
2	Journal of Moon	Asia	Social	72	106	1	0	6	64
3	Journal of Lumber	Americas	Physical	72	100	1	1	8	30
4	Journal of Houses	Europe	Social	72	102	1	0	8	38

Tables — `stargazer` — Data Tables

Set `header = FALSE` to remove the note preceding tables

````
```{r, data_table, echo=FALSE, results="asis"}
stargazer(data = head(df, n = 4), type = "latex", summary = FALSE, header = FALSE)
```
````

Table 1:

	name	origin	branch	h5_index	h5_median	english	subfield	issues	age
1	Journal of Bears	Americas	Physical	73	97	1	1	7	61
2	Journal of Moon	Asia	Social	72	106	1	0	6	64
3	Journal of Lumber	Americas	Physical	72	100	1	1	8	30
4	Journal of Houses	Europe	Social	72	102	1	0	8	38

Tables — `stargazer` — Data Tables

Define a caption with the `title` argument

```
```{r, data_table, echo=FALSE, results="asis"}

stargazer(data = head(df, n = 4), type = "latex", summary = FALSE, header = FALSE,
          title = "First four rows of the dataset")

```
```

Table 1: First four rows of the dataset

	name	origin	branch	h5_index	h5_median	english	subfield	issues	age
1	Journal of Bears	Americas	Physical	73	97	1	1	7	61
2	Journal of Moon	Asia	Social	72	106	1	0	6	64
3	Journal of Lumber	Americas	Physical	72	100	1	1	8	30
4	Journal of Houses	Europe	Social	72	102	1	0	8	38

Tables — `stargazer` — Summary Statistics Tables

Create a table of summary statistics instead, for the complete dataset

```
```{r, summary_table, echo=FALSE, results="asis"}

stargazer(data = df, type = "latex", summary = TRUE, header = FALSE,
          title = "Descriptive statistics")

```
```

Table 1: Descriptive statistics

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
h5_index	1,091	26.361	13.814	1	17	35	73
h5_median	1,091	39.400	21.272	3	25	52	109
issues	1,091	4.676	1.786	1	3	6	12
age	1,091	42.902	26.370	1	23	56	158

Tables — `stargazer` — Summary Statistics Tables

Keep only a selection of statistics

```
```{r, summary_table, echo=FALSE, results="asis"}

stargazer(data = df, type = "latex", summary = TRUE, header = FALSE,
          title = "Descriptive statistics", summary.stat = c("n", "mean", "sd", "min", "max"))

```
```

Table 1: Descriptive statistics

Statistic	N	Mean	St. Dev.	Min	Max
h5_index	1,091	26.361	13.814	1	73
h5_median	1,091	39.400	21.272	3	109
issues	1,091	4.676	1.786	1	12
age	1,091	42.902	26.370	1	158

Tables — `stargazer` — Summary Statistics Tables

Omit a selection of statistics for the same effect

```
```{r, summary_table, echo=FALSE, results="asis"}

stargazer(data = df, type = "latex", summary = TRUE, header = FALSE,
          title = "Descriptive statistics", omit.summary.stat = c("p25", "p75"))

```
```

Table 1: Descriptive statistics

Statistic	N	Mean	St. Dev.	Min	Max
h5_index	1,091	26.361	13.814	1	73
h5_median	1,091	39.400	21.272	3	109
issues	1,091	4.676	1.786	1	12
age	1,091	42.902	26.370	1	158

Tables — `stargazer` — Summary Statistics Tables

Flip the table

```
```{r, summary_table, echo=FALSE, results="asis"}

stargazer(data = df, type = "latex", summary = TRUE, header = FALSE, flip = TRUE,
          title = "Descriptive statistics", omit.summary.stat = c("p25", "p75"))

```
```

Table 1: Descriptive statistics

Statistic	h5_index	h5_median	issues	age
N	1,091	1,091	1,091	1,091
Mean	26.361	39.400	4.676	42.902
St. Dev.	13.814	21.272	1.786	26.370
Min	1	3	1	1
Max	73	109	12	158

Exercise

05:00

27) Summary Statistics Tables

- see `reproduce_this.pdf`: table 2 on page 8
- apply in `journals.Rmd`: table 2, between paragraphs 23 and 24

Tables — `stargazer` — Regression Tables

Create a table of regression models instead

```
```{r, regression_table, echo=FALSE, results="asis"}

stargazer(data = lm(h5_index ~ subfield, data = df),
          type = "latex", header = FALSE,
          title = "Regression Results")

```
```

Table 1: Regression Results

	Dependent variable:
	h5_index
subfield1	-12.926***
	(0.896)
Constant	36.171***
	(0.781)
Observations	1,091
R^2	0.160
Adjusted R^2	0.160
Residual Std. Error	12.665 (df = 1089)
F Statistic	207.926*** (df = 1; 1089)
Note:	*p<0.1; **p<0.05; ***p<0.01

Tables — `stargazer` — Regression Tables

Models can also be estimated outside the function first

```
```{r, regression_table, echo=FALSE, results="asis"}

 lm1 <- lm(h5_index ~ subfield, data = df)

 stargazer(data = lm1, type = "latex", header = FALSE,
           title = "Regression Results")

```
```

Table 1: Regression Results

	Dependent variable:
	h5_index
issues	1.913***
	(0.227)
Constant	17.415***
	(1.137)
Observations	1,091
R^2	0.061
Adjusted R^2	0.060
Residual Std. Error	13.391 (df = 1089)
F Statistic	70.959*** (df = 1; 1089)
Note:	*p<0.1; **p<0.05; ***p<0.01

Tables — `stargazer` — Regression Tables

Keep only a selection of statistics

```
```{r, regression_table, echo=FALSE, results="asis"}

stargazer(data = lm1, type = "latex", header = FALSE,
          title = "Regression Results",
          keep.stat = c("n", "rsq"))

```
```

Table 1: Regression Results

	Dependent variable:
	h5_index
issues	1.913^{***}
	(0.227)
Constant	17.415^{***}
	(1.137)
Observations	1,091
R^2	0.061
Note:	*p<0.1; **p<0.05; ***p<0.01

Tables — `stargazer` — Regression Tables

Display multiple models in the same table

```
```{r, regression_table, echo=FALSE, results="asis"}

stargazer(data = list(lm1, lm2), type = "latex",
          header = FALSE, title = "Regression Results",
          keep.stat = c("n", "rsq"))

```
```

Table 1: Regression Results

	Dependent variable:	
	h5_index	
	(1)	(2)
issues	1.913***	1.424***
	(0.227)	(0.212)
english1		17.262***
		(1.244)
Constant	17.415***	4.226***
	(1.137)	(1.415)
Observations	1,091	1,091
R^2	0.061	0.202
Note:	*p<0.1; **p<0.05; ***p<0.01	

Tables — `stargazer` — Regression Tables

Change variable labels

```
```{r, regression_table, echo=FALSE, results="asis"}

stargazer(data = list(lm1, lm2), type = "latex",
          header = FALSE, title = "Regression Results",
          keep.stat = c("n", "rsq"),
          dep.var.labels = "H5 Index",
          covariate.labels = c("Issues", "English"))

```
```

Table 1: Regression Results

	Dependent variable:	
	H5 Index	
	(1)	(2)
Issues	1.913***	1.424***
	(0.227)	(0.212)
English		17.262***
		(1.244)
Constant	17.415***	4.226***
	(1.137)	(1.415)
Observations	1,091	1,091
R^2	0.061	0.202
Note:	*p<0.1; **p<0.05; ***p<0.01	

Tables — `stargazer` — Regression Tables

Change significance levels

```
```{r, regression_table, echo=FALSE, results="asis"}

stargazer(data = list(lm1, lm2), type = "latex",
          header = FALSE, title = "Regression Results",
          keep.stat = c("n", "rsq"),
          dep.var.labels = "H5 Index",
          covariate.labels = c("Issues", "English"),
          star.cutoffs = c(0.05, 0.01, 0.001))

```
```

Table 1: Regression Results

	Dependent variable:	
	H5 Index	
	(1)	(2)
Issues	1.913^{***}	1.424^{***}
	(0.227)	(0.212)
English		17.262^{***}
		(1.244)
Constant	17.415^{***}	4.226^{**}
	(1.137)	(1.415)
Observations	1,091	1,091
R^2	0.061	0.202
Note:	*p<0.05; **p<0.01; ***p<0.001	

Exercise

07:30

28) Regression Tables

- see `reproduce_this.pdf`: table 3 on page 10
- apply in `journals.Rmd`: table 3, between paragraphs 30 and 31

Part 7. Addressing Functionality Gaps

Back to the contents slide.

Functionality Gaps

- Not everything is possible to achieve with R Markdown syntax, code chunks, and/or code
  - e.g., centering text
- Workarounds available through inclusion of other languages and/or syntaxes in .Rmd documents
  - e.g., incorporating HTML or LaTeX code into R Markdown
  - workarounds might be output specific
    - e.g., LaTeX-based workarounds may work only for LaTeX and PDF outputs
- There are no exclusive list of gaps or workarounds
  - these are specific to the output you want to achieve, problems you encounter
  - after writing a few manuscripts with R Markdown, you will have addressed most typical gaps in your workflow

Functionality Gaps — Examples

Problem:

How can we cross-reference figures, tables, and equations in R Markdown?

Solution:

Insert a LaTeX label into the targets (figures, tables, and equations), and then use the `\autoref{figure_caption}` syntax in text

Functionality Gaps — Examples — Cross-references

For figures, insert a LaTeX label into the `fig.caption` option, and use the `\autoref{latex_label}` syntax in text

````
\autoref{scatter_plot} visualises the relationship between the two journal metrics.

```{r ... fig.caption = "A Scatter Plot \\label{scatter_plot}"}

ggplot(data = df) +
       geom_point(...

```
````

Figure 1 visualises the relationship between the two journal metrics.

Functionality Gaps — Examples — Cross-references

For Markdown tables, insert a LaTeX label after the table caption, and use the `\autoref{latex_label}` syntax in text

```
See \autoref{handmade_table} for further details.

: A hand-made table with R Markdown \label{handmade_table}

+--------------------+--------------------+
| Left-Aligned       | Centered           |
...
```

See Table 1 for further details.

Functionality Gaps — Examples — Cross-references — Note

Note that there is a difference in the label syntax for figures and R Markdown tables

- we use a double backslash \\ to label figures
  - e.i., `\\label{scatter_plot}` because the label goes into a string
  - the first is an escape operator for the second, LaTeX backslash
- we use single backslash \ to label R Markdown tables
  - e.i., `\label{handmade_table}` because the label is not in any string
  - there is no need for the escape operator

Exercises

05:00

29) Referring to Figures

- see `reproduce_this.pdf`: pages 6 and 9
- apply in `journals.Rmd`: paragraphs 19, 21, and 27

30) Referring to Markdown Tables

- see `reproduce_this.pdf`: page 4
- apply in `journals.Rmd`: paragraph 11

Functionality Gaps — Examples — Cross-references

For `stargazer` tables, define a label with the `label` argument, and use the `\autoref{latex_label}` syntax in text

```
```{r, regression_table, echo=FALSE, results="asis"}

stargazer(data = list(lm1, lm2), type = "latex",
          ...
          label = "regression_results")

```

\autoref{regression_results} provides results from two (
```

Table 1 provides results from two OLS models.

Table 1: Regression Results

	Dependent variable:	
	H5 Index	
	(1)	(2)
Issues	1.913^{***}	1.424^{***}
	(0.227)	(0.212)
English		17.262^{***}
		(1.244)
Constant	17.415^{***}	4.226^{**}
	(1.137)	(1.415)
Observations	1,091	1,091
R^2	0.061	0.202

Functionality Gaps — Examples — Cross-references — Note

Note that we can cross-reference specific results in tables as well

- there is no gap here — this possible with inline code

```
In Model 1, the coefficient for _Issues_ is
`r round(coef(summary(lm1))["issues", "Estimate"], digits = 2)`.
```

In Model 1, the coefficient for *Issues* is 1.91.

Functionality Gaps — Examples — Cross-references

For equations, insert a LaTeX label in an equation environment, and use the `\autoref{latex_label}` syntax in text

```
\begin{equation}
\label{special_relativity}
E = mc_{2}
\end{equation}

According to \autoref{special_relativity}, space and time are linked.
```

According to Equation 1, space and time are linked.

Exercises

07:30

31) Referring to Tables

- see `reproduce_this.pdf`: pages 7 and 9
- apply in `journals.Rmd`: paragraph 23 and 29

32) Referring to Results in Regression Tables

- see `reproduce_this.pdf`: page 9
- apply in `journals.Rmd`: paragraph 29
  - hint: to extract the standard error from the model, use the column `Std. Error`

33) Referring to Equations

- see `reproduce_this.pdf`: page 7
- apply in `journals.Rmd`: paragraph 22
  - hint: transform the existing equation from R Markdown to LaTeX syntax, to be able to insert the label

Functionality Gaps — Examples

Problem:

R Markdown adds the list of references to the end of documents. This might be undesirable for some manuscripts, for example those with an appendix. Similarly, some journals require tables and figures to be added after references.

Solution:

Define where exactly the list of references should appear with the HMTL code `<div id ="refs">`

```
# References
<div id = "refs"></div>

# Appendix
```

Functionality Gaps — Examples

Problem:

R Markdown produces outputs with single-line-spaced text while we might prefer or be required (e.g., by journal submission rules) to double-space our manuscripts.

Solution:

Use the `doublespacing` command from the LaTeX package `setspace` (Carlisle et al., 2011)

- because the command comes from a package, we need to add it to YAML with `header-includes`
- including commands in YAML ensures they are applied throught the output*

```
---
...
header-includes:
    - \usepackage{setspace}\doublespacing
---
```

* This can be reversed anywhere in text, with the `singlespacing` command.

Exercise

02:00

34) Line Spacing

- introduce 1.5 spacing to the manuscript
  - hint: the command is called `onehalfspacing`
- except for the Abstract, which should be single spaced

Functionality Gaps — Examples

Problem:

Pages, tables, figures etc. are numbered continuously across an output. We might prefer or be required (e.g., by journal submission rules) to change this behaviour, for example for appendices.

Solution:

Use the `setcounter` in combination with the `renewcommand` command, outside code chunks

```
\setcounter{page}{1}
\renewcommand*{\thepage}{A\arabic{page}}

\setcounter{table}{0}
\renewcommand*{\thetable}{A\arabic{table}}

\setcounter{figure}{0}
\renewcommand*{\thefigure}{A\arabic{figure}}
```

Part 8. Using Version Control

Back to the contents slide.

Version Control

- Research papers have many versions before publication
  - typically written over a long period of time, in numerous sittings
  - at the end of every sitting, essentially a different version of the same manuscript is created*

* They also often written by multiple authors and/or on different computers, increasing the number of versions created. Here I assume projects are single-authored on a single computer, leaving the topic of collaboration (including, with oneself) to the next section — Part 9.

Version Control

- Research papers have many versions before publication
  - typically written over a long period of time, in numerous sittings
  - at the end of every sitting, essentially a different version of the same manuscript is created
- With many versions created over time, there emerge at least two challenges
  - keeping track of changes and versions
  - reverting to a previous version when necessary
- We all version control, in different ways, such as
  - edit, rename, save
  - use applications or websites such as Dropbox, Google Docs, Overleaf
  - use distributed version control systems such as Git and GitHub

Version Control — Manual Attempts

Typically, hand-made attemps to version control lead to cluttered folders

```
manuscript
    |
    |- journals_FINAL_19May.Rmd
    |- journals_FINAL.Rmd
    |- journals_26APRIL_newliterature.Rmd

    ...

    |- journals.Rproj
    |- references.bib
    |- apa_7th.csl
```

Version Control — Git and GitHub — Definitions

- Git
  - a software that keeps track of versions of a set of files
  - it is *local* to you, the records are kept on your computer

- GitHub
  - a hosting service, or a website, that can keep the records
  - it is *remote* to you, like the Dropbox website
  - but unlike Dropbox, GitHub is specifically structured to keep records with Git

- Repository, or repo
  - a set of files whose records are kept together, by Git and/or on GitHub
  - it is like a folder, which can keep files and other folders containing files

Version Control — Git and GitHub — Definitions

- To commit
  - to take a snaphot of, or to version, a repository
  - it is like saving a new version of all files and sub-folders in your project folder with a new name
  - it is local, the records are kept on your computer unless you `push`

- To push
  - to move the records from Git to GitHub, from your computer to online server
  - it is like uploading (the new versions of) your files and sub-folders to a website,
  - it also involves merging, if this not the first push

* For projects that are single-authored on a single computer, merging is typically automatic. It becomes an issue for collaborated projects, which we will cover in the next section — Part 9.

Version Control — Git and GitHub

Version control with Git and GitHub requires

1. initial setup, done once*
    - unless for a new computer or, if ever, a new GitHub account
    - a bit technical, but worth the hassle
2. project setup, repeated for every paper
    - shorter, less complicated

* We have started this process already, in Part 1 of the workshop, by downloading and installing Git and signing up for GitHub. Back to the relevant slide.

Version Control — Git — Inital Setup

1) Enable version control with RStudio

- from the RStudio menu, follow:

  ```
  Tools -> Global Options -> Git/SNV -> Enable version control interface for
  RStudio projects
  ```

- RStudio will likely find Git automatically. In case it cannot, Git is likely to be at
  - `c:/Program Files/Git/bin/git.exe` on Windows
  - `/usr/local/git/bin/git` on Mac

Version Control — Git — Inital Setup

2) Set Git Bash as your shell (Windows-only step)

- from the RStudio menu, follow:

```
Tools -> Global Options -> Terminal -> New terminals open with: Git Bash
```

Version Control — Git — Inital Setup

3) Introduce yourself to Git

- from the RStudio menu, follow:

  ```
  Tools -> Terminal -> New Terminal
  ```

- enter the following lines in the Terminal, with the email address that you have used to sign up for GitHub

```
git config --global user.name "YOUR-NAME"
git config --global user.email "YOUR-EMAIL-ADDRESS"
```

- enter the following line in the Terminal, to observe whether the previous step was sucessful

```
git config --global --list
```

Version Control — Git and Github — Project Setup*

1) Initiate local version control with Git

- from the RStudio menu, follow:

```
Tools -> Version Control -> Project Setup... -> Version Control System -> Git
```

- after confirming your new repository, and restarting the session, observe that
  - now there is now a Git tab in RStudio, documenting the differences between you local repository and the one on GitHub. When you change a file, it will appear here.
  - your project now includes a `.gitignore` file
    - this is where you can list files and/or folders to be excluded from being tracked

* These instructions presume there is an exiting RStudio project to be set up for version control. If not, or to start a new project, follow from this slide first.

Version Control — Git and Github — Project Setup

2) Create a new GitHub repository

- on GitHub, follow:

  ```
  Repositories -> New -> Repository name (e.g., "rwd_workshop") -> Public ->
  Create repository
  ```

- observe that
  - repository URLs have the following structure: https://github.com/USER_NAME/REPOSITORY_NAME
    - this is the address to view the repository online
    - for use in the `Terminal`, the address gets the `.git` extension
      - e.g., https://github.com/USER_NAME/REPOSITORY_NAME.git

Version Control — Git and Github — Project Setup

3) Push an existing repository

- from the RStudio menu, follow:

  > `Tools -> Terminal -> New Terminal`

- enter the following lines in the `Terminal`, with your username and repository name

```
git remote add origin https://github.com/USER_NAME/REPOSITORY_NAME.git
git add .
git commit -m "first commit"
git push -u origin master
```

- if this is your first time using GitHub with RStudio, you will be prompted to authenticate
  - follow the instructions on your screen and in your email
- observe that your project files are now online, listed on the GitHub repository

Version Control — Git and Github — Workflow

1) Edit and Save

- work on one or more files under version control
  - e.g., delete the first sentence of the abstract in `journals.Rmd`, and save it
  - under the Git tab in RStudio, find the list of files that you edited since the last push
  - these will have `M`, for modified, as `Status`

2) Commit and Push

- tick `Staged`* for one or more files that you would like to commit
  - enter a `Commit message` that summarises the edits
  - click `Commit` to create a record of the new version locally to your computer
  - click `Close -> Push` to push the version to GitHub

* To stage is to add files to be comitted. It allows us to commit files individually or together with other files.

Version Control — Git and Github — Workflow

1) Edit and Save

- work on one or more files under version control
  - e.g., delete the first sentence of the abstract in `journals.Rmd`, and save it
  - under the Git tab in RStudio, find the list of files that you edited since the last push
  - these will have `M`, for modified, as `Status`

2) Commit and Push

- tick `Staged` for one or more files that you would like to commit
  - enter a `Commit message` that summarises the edits
  - click `Commit` to create a record of the new version locally to your computer
  - click `Close -> Push` to push the version to GitHub
- observe the changes in the Git tab in RStudio and on the GitHub repository

Version Control — Git and Github — `.gitignore`

- `.gitignore` specifies which file(s) and/or folder(s) should be excluded from version control
  - a set of project-specific files are ignored by default
    - see your `.gitignore` file
- `.gitignore` lists one item per line
  - each line has a pattern, which determines whether one or more files or folders are to be ignored
- See the documentation at https://git-scm.com/docs/gitignore
  - for pattern formats and further details

Version Control — Git and Github — `.gitignore`

- There might be good reasons to ignore some others, including files
  - that contain information that we do not want others to see
    - e.g., personal API keys
  - that we do not have the right to share with others
    - e.g., secondary data with user agreements otherwise
  - that we (re-)create automatically as outputs
    - e.g., `journals.pdf`, as opposed to journals.Rmd

Version Control — Git and Github — `.gitignore`

- Observe that, by default, `.gitignore` has a list of project-specific files
  - you can delete, or comment out, any or all to start including them in version control

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
```

Version Control — Git and Github — `.gitignore`

- Observe that, by default, `.gitignore` has a list of project-specific files
- In addition, you can ignore, for example,
    - a specific folder, relative to the root directory

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
/manuscript/
```

Version Control — Git and Github — `.gitignore`

- Observe that, by default, `.gitignore` has a list of project-specific files
- In addition, you can ignore, for example,
  - a specific folder, relative to the root directory
  - a specific file in a specific folder, relative to the root directory

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
/manuscript/
/manuscript/journals.pdf
```

Version Control — Git and Github — `.gitignore`

- Observe that, by default, `.gitignore` has a list of project-specific files
- In addition, you can ignore, for example,
  - a specific folder, relative to the root directory
  - a specific file in a specific folder, relative to the root directory
  - a specific file in any folder

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
/manuscript/
/manuscript/journals.pdf
journals.pdf
```

Version Control — Git and Github — `.gitignore`

- Observe that, by default, `.gitignore` has a list of project-specific files
- In addition, you can ignore, for example,
  - a specific folder, relative to the root directory
  - a specific file in a specific folder, relative to the root directory
  - a specific file in any folder
  - all files with a specific extension, anywhere in the project

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
/manuscript/
/manuscript/journals.pdf
journals.pdf
*.pdf
```

Version Control — Git and Github — `.gitignore` — Notes

- There are many other pattern formats
  - see the documentation at https://git-scm.com/docs/gitignore
- Starting to ignore a file or folder that is already being tracked requires clearing the cache
  - after changing and saving `.gitignore`, enter the following line in the `Terminal`
  - with your speficic `/path/to/file`

```
git rm --cached /path/to/file
```

- The following command clears *all* cache
  - might be useful after changes to `.gitignore` that involves several files or folders
  - but should be used with care, on an otherwise up-to-date repository

```
git rm -r --cached .
```

Exercises

05:00

35) Reproducibility and Version Control

- imagine that, after producing all these tables and figures, and writing up your results, you have decided to exclude journals from Oceania from analysis
  - hint: use the `filter` function in the data chunk
  - create a new version of the manuscript
  - comit and push to GitHub

36) Gitignore

- stop tracking `journals.pdf`
  - change `.gitignore`
  - remove `journals.pdf` from cache
  - comit and push to GitHub

Part 9. Collaborating with Others

Back to the contents slide.

Collaboration

- Many research papers are written by multiple authors and/or on multiple computers
    - yourself on a different computer (e.g., laptop at home, desktop at office), poses similar challenges as collaboration
- With multiple authors and/or computers, there emerges at least two additional challenges beyond version control
    - communicating the versions to other authors and/or computers
    - working on the same project with co-authors at the same time
- We all manage collaboration, in different ways, such as
    - edit, rename, save, e-mail
    - use applications or websites such as Dropbox, Google Docs, Overleaf
    - use distributed version control systems such as Git and GitHub

Collaboration — Git and GitHub — Definitions

- To pull
  - to move the (presumably) up-to-date records from GitHub to your computer
  - it is like downloading a zipped folder of files
- To merge
  - to integrate different versions into a single version
    - e.g., the old version on your laptop, with (the changes in) the new version from GitHub
  - except the first push or pull, pushing and pulling necessiate merging
- Merge conflict
  - emerges when versions to be merged include edits *on the same line of the same file*
    - edits on different lines are not a problem as changes are tracked line by line
  - less likely to occur in one-author-multiple-computer setting
    - more likely while collaborating with others
  - requires human intervention, to decide which edit to keep and which one to discharge

Collaboration — Git and GitHub — Definitions

- Branch
  - a line of development in a repository; a copy of the repository, with all its versions, at a given time
  - by default, repositories have one branch, called *master*
- Pull request
  - a proposal to pull and merge
    - e.g., a proposal from one co-author to another, -e.g., tp merge a branch into master
  - it allows a review of changes on GitHub before merge, to deal with potential merge conflicts

Collaboration — Git and GitHub — Project Setup

- The setup depends on the users' role, on whether they are
  - the *owner* who creates the GitHub repository, or
  - the *collaborator* who is then added to that repository

- Once the project is setup
  - it continues to be associated with the owner's GitHub profile
  - at the same time, it is listed under the collaborator's profile as well
  - both the owner and the collaborator have the same rights, unless otherwise restricted

Collaboration — Git and GitHub — Project Setup — Owner

1) The setup for the owner is largely the same as in any single-author, single-computer scenario

- following the instructions on this slide forward
  - to introduce version control to a local project with Git,
  - to create a remote repository for that project on GitHub, and
  - to associate the local project with the remote repository

2) As an additional step, the owner needs to invite their collaborator(s) to the project

- following, from the relevant GitHub repository,

```
Settings -> Manage access -> Invite a collaborator
```

Collaboration — Git and GitHub — Project Setup — Collaborator

1) Notice that the remote part of the setup is done by the owner for the collaborator

- subject to acceptance of the invitation
  - invitations are available directly at https://github.com/notifications, but also sent via email
  - with an option to "Accept invitation"
  - on acceptance, projects appear among the repositories of the collaborator

2) The local part of the setup still needs to be done

- by creating a new RStudio project with version control
- following, from the Rstudio menu,*

```
File -> New Project -> Version Control -> Git
```

- the `Repository URL`, required for the above process, is the version without the `.git` extension
  - in the form of https://github.com/OWNER_USER_NAME/REPOSITORY_NAME

Exercises

10:00

37) Owner Setup

- create a new version-controlled RStudio project, with Git and GitHub
- add the default R Markdown template to your project
  - hint: click `File -> New File -> R Markdown -> OK` to create the template
  - another hint: name the project and the template in a way that they are easily distinguishable from your partner's project and template

38) Invitation to Collaborate

- invite the partner in your current group as a collaborator to your new project
  - hint: you will need their username, full name, or email address to do so

39) Collaborator Setup

- accepting the invitation from your partner, do the necessary arrangements so that you can collaborate on your partner's project

Colloboration — Git and Github — Workflow

1) Pull

- on the Git tab in RStudio, click `Pull` to move the up-to-date records from GitHub to your computer
  - if your collaborator has not pushed anything since your last pull, you will be noticed that `Already up-to-date`.
  - collaborative projects require pulling as well as pushing because your collaborator(s) might have pushed their commits to GitHub
  - pulling frequently minimises the risk of merge conflicts

2) Edit and save; commit and push

- the same procedure as in any single-author, single-computer scenario
  - as described on this slide forward
- pushing frequently minimises the risk of merge conflicts

Exercise

05:00

40) Non-simultaneous Collaboration

- take in turns with your partner to work on the same document (of the same project)
  - *owner*: edit the first header in the document (i.e., "R Markdown"), save, commit, and push
  - *owner and collaborator*: observe the changes, if any, on your own `.Rmd` file, and/or on your GitHub repository
    - click on the relevant commit message on GitHub and observe the commit
  - *collaborator*: pull, revert the header back to original, save, commit, and push

- notice that you have not encountered any errors and/or merge conflicts
  - because everyone edited and merged with an up-to-date document
  - this is the default scenario in single-author, multiple computer scenario

Exercise

10:00

41) Simultaneous Collaboration — Different Lines

- work on the same document at the same time
  - *owner*: edit the first header in the document (i.e., "R Markdown"), save, commit, and push
  - *collaborator*: edit the second header in the document (i.e., "Including Plots"), save, commit, and push
    - observe the error message that the last pusher will receive, follow the instructions on RStudio to solve the problem

- notice that you have encountered an error but not a merge conflict
  - pulling before pushing solves the problem because the edits are not on the same line
  - the merge takes place automatically, on the *local* repository of the last pusher

Exercise

10:00

42) Simultaneous Collaboration — Same Line

- work on the same document at the same time
  - *owner*: edit the first header in the document again, save, commit, and push
  - *collaborator*: edit the first header in the document as well, save, commit, and push
    - observe the error message that the last pusher will receive, follow the instructions on RStudio to solve the problem

- notice that you have encountered not only an error but also a merge conflict
  - pulling before pushing alone does not solve the problem because the edits are on the same line
    - the conflict cannot be solved automatically — it needs human intervention
    - by pulling first, you can view the conflict directly on the file
      - marked between less than < and greater than > signs, divided by the equal signs
      - solution is to accept the remote version, by deleting your edit and or moving that edit to a different line
  - merging takes place on the *local* repository of the last pusher

Colloboration — Git and Github — Workflow — Alternative

- The workflow above is rather simple, but has some disadvantages, including
  - not easy, albeit still possible, to see the edits of the collaborators
  - not clear who is in charge of the overall progress
  - not possible to discuss edits
  - not possible to compromise on conflicting edits

- An alternative workflow exits
  - work on different branches of the same project
  - version control to your own branch
  - create pull requests with comments
  - merge the branch into master

Colloboration — Git and Github — Workflow — Alternative

1) Branch

- click `New Branch` on the Git tab
  - name it, and leave everything else as default
  - notice that you are now working on a new branch

2) Edit and save; commit and push

- the same procedure as in any single-author, single-computer scenario
  - as described on this slide forward
- notice, on GitHub, that your commit is in the new branch, while *master* remains unchanged

3) Pull request

- On GitHub, click

  ```
  Pull requests -> New pull request
  ```

- choose what is to be pulled, and write a note to your collaborator who can accept or reject the merge
  - if there are merge conflicts, the collaborator solves them on GitHub before merging

Exercises

10:00

43) Pull request

- create a pull request for your collaboration project
  - create a branch for yourself
  - edit any line, save, commit, and push
  - request your branch to be merged

44) Merging

- view the pull request of your collaborator
- take the necessary steps to merge it to *master*

Colloboration — Git and Github — Workflow — Notes

- It is possible to edit `.Rmd` documents directly on GitHub
    - click on any editable file, and `Edit this file`
    - commit changes, either as a direct commit or a pull request
- A GitHub account is enough for collaboration with co-authors who do not work with Git, R, or RStudio
    - not possible to knit to see the outcome
    - would suit co-authors whose contributions are plain text

Exercises

05:00

45) GitHub edit

- create two edits on the `.Rmd` document in your collaboration project
- commit one of the edits as a direct commit
- commit the other as a pull request

Part 10. Working on a Real Project

Back to the contents slide.

Real Project

- Consider converting a real project to R Markdown
    - now, in the remainder of the workshop
- Choose an existing project, preferably
    - single-authored
    - at an early stage
    - but one that you are, will be, working on
- Ask me for help
    - with no more slides to go through, I will now focus on helping you start your first project in R Markdown

References

Back to the contents slide.

References

Allaire, J. J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W. and Iannone, R. (2020). rmarkdown: Dynamic documents for R. R package, version 2.1.

Blair, G., Cooper, J., Coppock, A., Humphreys, M., Rudkin, A. and Fultz, N. (2019). fabricatr: Imagine your data before you collect it. R package, version 0.10.0.

Carlisle, D., Fairbairns, R., Harris, E. and Tobin, G. (2011). setspace – Set space between lines. LaTeX package, version 6.7a.

Dowle, M. and Srinivasan, A. (2019). data.table: Extension of 'data.frame'. R package, version 1.12.8.

Gagolewski, M. (2020). stringi: Character String Processing Facilities. R package, version 1.4.6.

Hlavac, M. (2018). stargazer: Well-formatted regression and summary statistics tables. R package, version 5.2.2.

Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M., & Despouy, P. (2020). plotly: Create Interactive Web Graphics via 'plotly.js'. R package, version 4.9.3.

Wickham, H. and Grolemund, G. (2019). R for data science. O'Reilly. Open access at https://r4ds.had.co.nz.

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., Woo, K., Yutani, H. and Dunnington, D. (2020a). dplyr: A grammar of data manipulation. R package, version 0.8.5.

References

Wickham, H., François, W., Henry L. and Müller, K. (2020b). ggplot2: Create elegant data visualisations using the grammar of graphics. R package, version 0.8.5.

Wiernik, B. M. (2020). American Psychological Association 7th edition (no ampersand). Citation style language file, version 1.0.

Xie, Y. (2020a). tinytex: Helper functions to install and maintain TeX Live and compile LaTeX documents. R package, version 0.21.

Xie, Y. (2020b). knitr: A general-purpose package for dynamic report generation in R. R package, version 1.28.

Xie, Y. (2020c). bookdown: Authoring books and technical documents with R Markdown. R package, version 0.18.

Xie, Y. (2020d). blogdown: Create blogs and websites with R Markdown. R package, version 0.18.

Xie, Y., Allaire, J. J., and Grolemund, G. (2019). R markdown: The definitive guide. CRC Press. Open access at https://bookdown.org/yihui/rmarkdown.

Zhu, H. (2019). kableExtra: Construct Complex Table with 'kable' and Pipe Syntax. R package, version 1.1.0.

The workshop ends here.

Congratulations for making it this far, and

thank you for joining me!

Back to the contents slide.