Reproducible Documents with `{rmarkdown}`

Day 2

Instructor: Selina Baldauf

Freie Universität Berlin - Theoretical Ecology

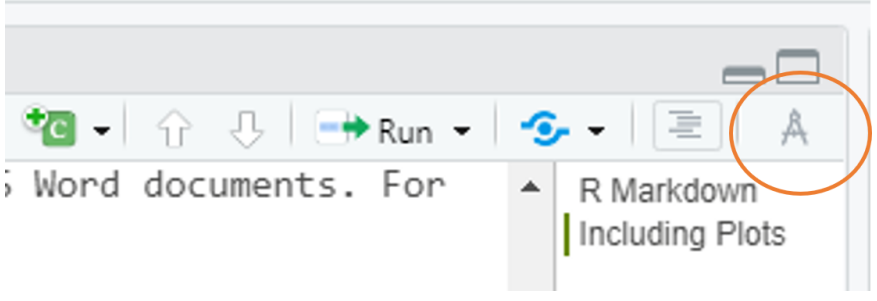
2022-22-03 (updated: 2022-03-29)

Topics today

- Citations
- A bit more output formats
  - `html_document`
  - `pdf_document`
  - `word_document`
- Make tables look nice
- Some more tips and good practice

The visual editor in RStudio

- WYSIWYG editor (*What you see is what you get*)
  - More similar to Word etc. but with less functionality
- Click on the button on the top right



- Very helpful in the beginning until you remember how everything works in markdown
- Especially helpful for markdown tables and citations
- But careful: Can reformat the `.Rmd` file a bit, so sometimes if you switch back it can look different than before.

Adding citations - The classic way

Bibliographies can be included via a BibTeX data base.

- Create a `.bib` file that consists of bibliography entries

```
@Book{cookbook,
  title = {R Markdown Cookbook},
  author = {Yihui Xie and Christophe Dervieux and Emily Riederer},
  publisher = {Chapman and Hall/CRC},
  address = {Boca Raton, Florida},
  year = {2020},
  note = {ISBN 9780367563837},
  url = {https://bookdown.org/yihui/rmarkdown-cookbook},
  }
```

Adding citations - The classic way

Bibliographies can be included via a BibTeX data base.

- Create a `.bib` file that consists of bibliography entries
- Add name and location of your `.bib` file as a medatada field in YAML header

```
---
output: html_document
bibliography: references.bib
---
```

- Cite an article from the database with `@bib_item_name` for in text citations or `[@bib_item_name]` for citation in brackets
  - Here, I cite `@cookbook` because it's a good book `[@cookbook]`
  - Here, I cite Xie, Dervieux, and Riederer (2020) because it’s a good book (Xie, Dervieux, and Riederer 2020)
- List of references used will be added to the end of the document
  - Just add a heading `# References` to end of the doc

Adding citations - The classic way

- Add a custom citation style file with:

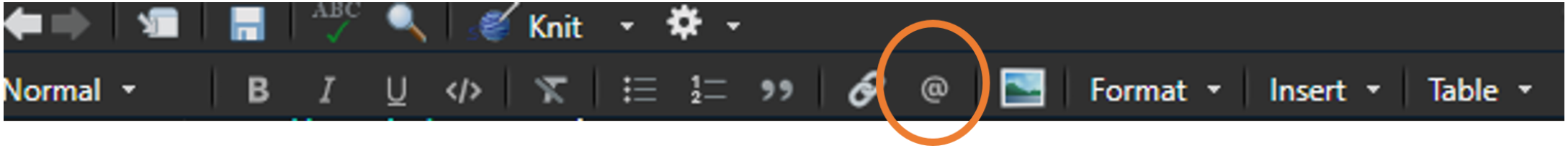
```
---
output: html_document
bibliography: references.bib
csl: myrefstyle.csl
---
```

- Most (all?) reference managers can export your citations as a `.bib` file
- Problem: RStudio does not auto-fill citations
    - You have to know the name of the citation in order to cite it

Adding citations - Visual editor

Citations can also be added using the **visual editor** in RStudio.

- Visual editor creates and extends `.bib` file automatically
- Search and add citations from
  - The bibliography file
  - Zotero
  - DOI
  - ...
- Just click on the `@` symbol in the visual editor to add a citation



- You can also start typing `@` and the editor will suggest you a list of citations that fit

Adding citations - Visual editor

Using Zotero

- If you use Zotero on your machine, RStudio should automatically detect the installation
- If not, go to `Tools->Global Options->R Markdown -> Citations` and enter the location of your Zotero data directory and the library that you would like to use
    - In General this should be recognized automatically

Now you

Task 1: Add some citations

Find the task description here

A bit more on the output formats

`html_document`, `pdf_document`, `word_document`

Specify multiple ouput types

- Specify multiple output types in the YAML header
  - Here just the default settings

```
---
title: "My first document"
author: "Selina Baldauf"
date: "3/22/2022"
output:
  html_document: default
  pdf_document: default
  word_document: default
bibliography: references.bib
---
```

- Decide which output you want before rendering
- Chose the output type with the little arrow next to the knit button
- If you knit without specifying the output type, the last rendered type is taken

Specify different ouput types

- Specify the options for different output types in the yaml header

```
---
title: "My first document"
author: "Selina Baldauf"
date: "3/22/2022"
output:
  html_document:
    toc: true
    toc_float: true
    highlight: "kate"
  pdf_document:
    toc: true
    highlight: "espresso"
bibliography: references.bib
---
```

- Keep in mind that multiple output types can become difficult if you use a lot of options and functionality specific only to one of the types
  - You will see this later e.g. when formatting tables

`html_document`

See here for more examples and options

html_document

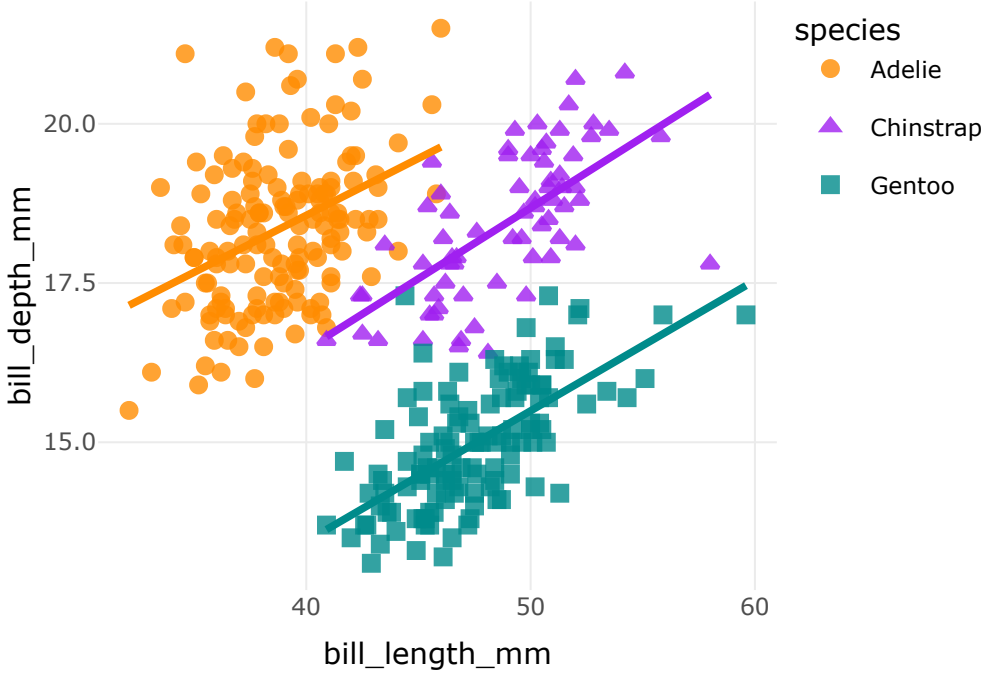
Interactive graphs e.g. with `plotly`

```
library(ggplot2)
scatter <- ggplot(
  data = penguins,
  aes(
    x = bill_length_mm,
    y = bill_depth_mm,
    color = species,
    shape = species
  )
) +
  geom_point(size = 2, alpha = 0.8) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_color_manual(values = c("darkorange", "purple", "cyan4")) +
  theme_minimal()
```

html_document

Interactive graphs e.g. with `plotly`

```
library(plotly)
ggplotly(scatter)
```



html_document

Tabbed sections

```
## This is the main section {.tabset}

And here I add some text to the main section.

### The first tab

some content

### The second tab

some other content
```

This is the main section

And here I add some text to the main section

The first tab | The second tab

some content

html_document

Customize with HTML and CSS (Advanced)

- Customize elements using HTML and/or CSS (advanced)
- Define custom elements
- Add the custom CSS to the YAML header

```
output:
  html_document:
    css: "my-style.css"
```

- Add HTML tags with CSS e.g. to change font color

`<span style="color: red;">red text</span>` becomes red text

`pdf_document`

See here for more examples and options

pdf_document

See all options and default values with `?rmarkdown::pdf_document`

- `fig_caption` will automatically number figures
- `citation_package` define the citation package to use (`natbib` or `biblatex`)
- `keep_tex`: Keep the intermediate `*.tex` file?
- `template`: Path to a template file

Example

```
---
title: "My first document"
author: "Selina Baldauf"
date: "3/22/2022"
output:
  pdf_document:
    fig_caption: true
    citation_package: "natbib"
    keep_tex: true
    template: "my_template.tex"
---
```

pdf_document

- You can use LaTeX syntax in the text
- Define some latex options as *top-level* YAML metadata, e.g.

```
---
title: "My first document"
author: "Selina Baldauf"
date: "3/22/2022"
output:
  pdf_document:
    fig_caption: true
fontsize: 11pt
geometry: "margin=1in"
documentclass: "article"
urlcolor: "blue"
---
```

word_document

See here for more examples and options

word_document

See all options and default values with `?rmarkdown::word_document`

- Not so many features directly available in R Markdown
- Use an office template for customization

```
---
title: "My first document"
author: "Selina Baldauf"
date: "3/22/2022"
output:
  word_document:
    reference_docx: "my-styles.docx"
---
```

Read or watch how to create a custom office template

- Package `{officedown}` might be useful
  - It offers templates for advanced Word documents and Powerpoint presentations
  - `File` -> `New File` -> `R Markdown...` -> `From Template`

Now you

Task 2: Try different output formats

Find the task description here

Nice looking tables in R Markdown

Nice looking tables with R Markdown

- The default for printing tables looks the same as printing it in the console:

```
iris_sum
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

- This is not really nice for documents

`knitr::kable()`

Simple to use table generator from the `knitr` package.

```
knitr::kable(iris_sum) # or iris_sum %>% knitr::kable()
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

- Chose `kable` as default table printing in YAML header:

```
df_print: "kable"
```

knitr::kable()

Add arguments for additional formatting:

```
kable(x,
  format,
  digits = getOption("digits"),
  row.names = NA,
  col.names = NA,
  align, caption = NULL,
  label = NULL,
  format.args = list(),
  escape = TRUE, ...
)
```

- See here for many examples many different use cases

knitr::kable()

Example:

```
knitr::kable(
  iris_sum,
  digits = 1,
  col.names = c("Species", "Sepal Length", "Sepal Width", "Petal Length", "Petal Width"),
  caption = "Summary of the Iris data",
  align = "l"
)
```

Table: Summary of the Iris data

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
setosa	5.0	3.4	1.5	0.2
versicolor	5.9	2.8	4.3	1.3
virginica	6.6	3.0	5.6	2.0

The `{kableExtra}` package

- Provides options for table styling
- Most of the features work for both HTML and PDF tables
- Find the full documentation here
    - If you use tables a lot, I recommend looking through the documentation to see all possibilities
- Load the packages in the setup chunk before using them

```
library(knitr)
library(kableExtra)
```

The {kableExtra} package

`kable_styling()` is the basic styling function

- Use the pipe operator (`%>%`) to pipe `kable()` output to styling function `kable_styling()`

```
iris_sum %>%
  kable() %>%
  kable_styling(
  full_width = FALSE, # display table on full page width?
  position = "center",  # if not full width -> where
  font_size = 15
)
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

The {kableExtra} package

`kable_styling()` provides styling options

- Additional styling options for HTML output are passed via `bootstrap_options`

```
iris_sum %>%
  kable() %>%
  kable_styling(
  full_width = FALSE, # display table on full page width?
  position = "center",  # if not full width -> where
  font_size = 15,
  bootstrap_options = c("striped", "hover")
)
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

The {kableExtra} package

`kable_styling()` provides styling options

- Additional styling options for PDF output are passed via `latex_options`

```
iris_sum %>%
  kable(booktabs = TRUE) %>%
  kable_styling(
    full_width = FALSE, # display table on full page width?
    position = "center", # if not full width -> where
    font_size = 15,
    bootstrap_options = c("striped", "hover"),
    latex_options = c("striped", "hold_position", "scale_down")
  )
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

- Depending on the output format you chose, `bootstrap_options` or `latex_options` will be ignored

The {kableExtra} package

`kable_styling()` provides styling options

- Additional styling options for PDF output are passed via `latex_options`

```
iris_sum %>%
  kable(booktabs = TRUE) %>%
  kable_styling(
    full_width = FALSE, # display table on full page width?
    position = "center", # if not full width -> where
    font_size = 15,
    bootstrap_options = c("striped", "hover"),
    latex_options = c("striped", "hold_position", "scale_down")
  )
```

- `booktabs = TRUE` will use the `booktabs` LaTeX package to create nice horizontal lines and removes vertical lines
- `hold_position` places the table where it is created in the document (no floating)
- `striped` creates striped tables

The {kableExtra} package

Packing rows and columns

```
iris_sum %>%
  kable() %>%
  kable_styling(font_size = 15) %>%
  add_header_above(c("", "Sepals" = 2, "Petals" = 2)) %>%
  pack_rows("Group 1", 1, 1) %>%
  pack_rows("Group 2", 2, 3)
```

	Sepals		Petals	
Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Group 1				
setosa	5.006	3.428	1.462	0.246
Group 2				
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

The {kableExtra} package

Adding footnotes

```
iris_sum %>%
  kable() %>%
  footnote(general = "Here is a general comments of the table. ",
           number = c("Footnote 1; ", "Footnote 2; "),
           alphabet = c("Footnote A; ", "Footnote B; "),
           symbol = c("Footnote Symbol 1; ", "Footnote Symbol 2")
           )
```

virginica	6.588	2.974	5.552	2.026

Note:

Here is a general comments of the table.

[1] Footnote 1;

[2] Footnote 2;

[a] Footnote A;

[b] Footnote B;

[*] Footnote Symbol 1;

[†] Footnote Symbol 2

The `{kableExtra}` package

Some predefined html themes

- `kableExtra` offers some themes for HTML tables
  - `kable_paper`, `kable_classic`, `kable_classic_2`, `kable_minimal`, `kable_material` and `kable_material_dark`
  - Use them alternative to `kable_styling()`

```
iris_sum %>%
  kable() %>%
  kable_classic(
    font_size = 15
  )
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

- This only works with HTML output! It will give you an error for PDF output.

The `{flextable}` package

- Works with PDF, HTML and Word output
  - I recommend it for Word output
- Alternative to `kable` and `kableExtra`
- Set options for all tables in beginning (e.g. in setup chunk)

```
library(flextable)
set_flextable_defaults(
  font.size = 10,
  theme_fun = theme_booktabs,
  padding = 6,
  digits = 1
)
```

- See all options with `?flextable::set_flextable_defaults`

The {flextable} package

- An example table that looks decent in all 3 outputs

```
iris_sum %>%
  flextable() %>%
  set_caption("Summary of the iris data") %>%
  set_header_labels(
    Sepal.Length = "Sepal Length",
    Sepal.Width = "Sepal Width",
    Petal.Length = "Petal Length",
    Petal.Width = "Petal Width"
  ) %>%
  colformat_double()
```

Summary of the iris data

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
setosa	5.0	3.4	1.5	0.2
versicolor	5.9	2.8	4.3	1.3
virginica	6.6	3.0	5.6	2.0

Now you

Task 3: Create a nice table

Find the task description here