

R Markdown:

Reproducible Report Writing

Bianca Peterson (PhD)

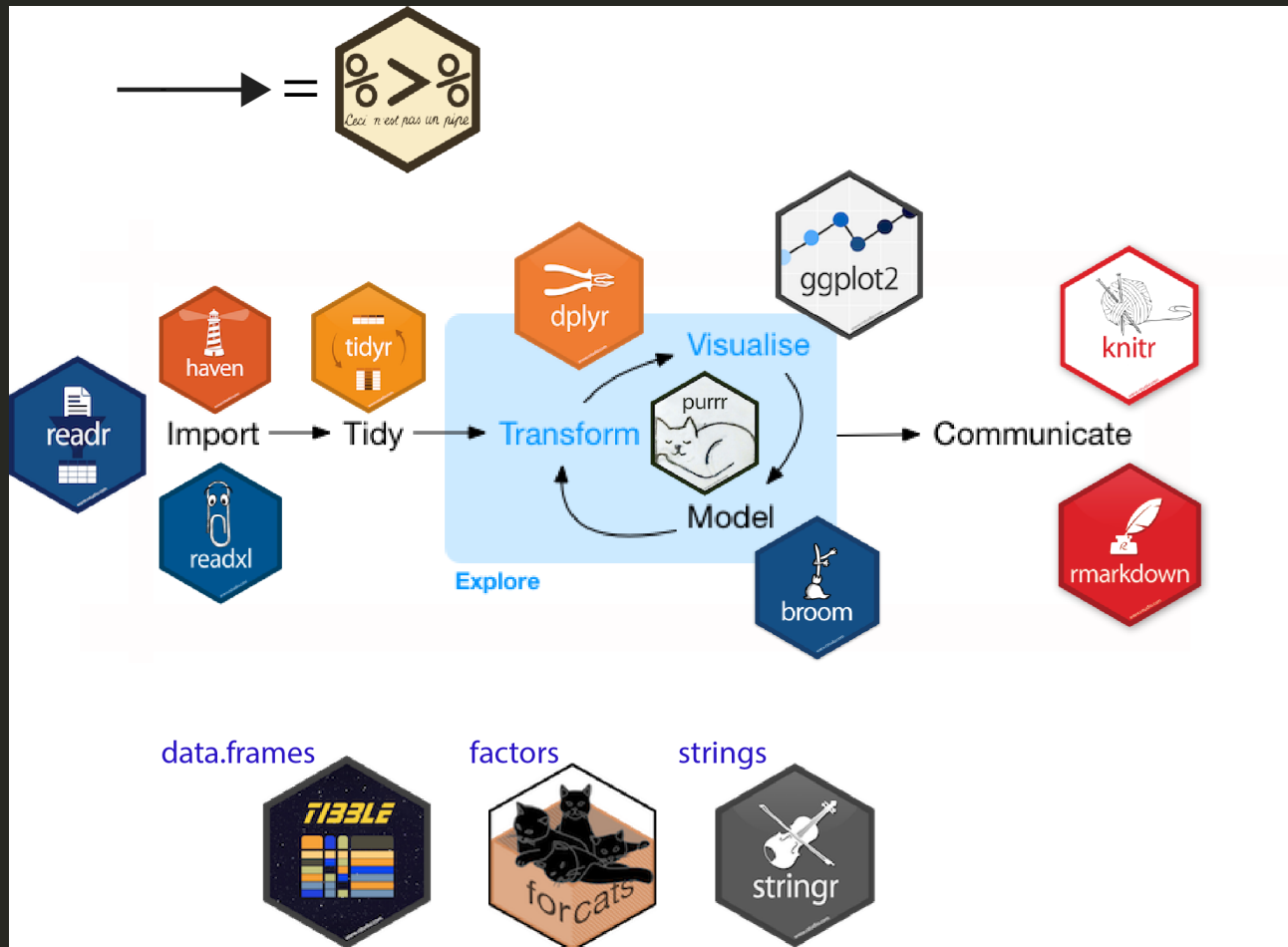
Conquest Analytics and Training

24 June 2021

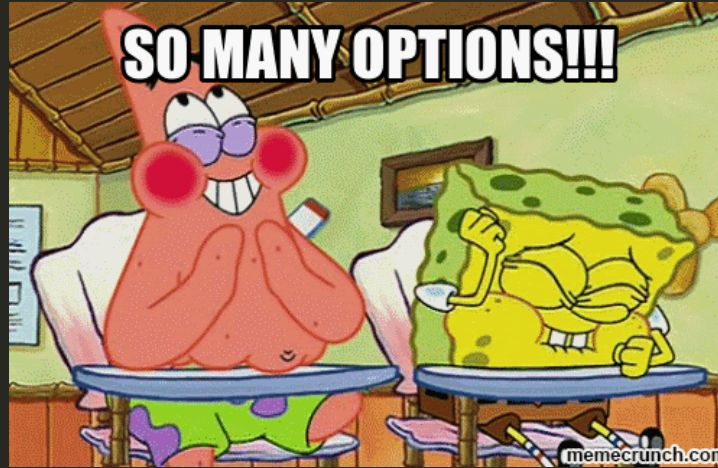
What is R Markdown?

- R Markdown = text + code + output (graphs, tables, etc.)
- R Markdown uses Markdown syntax
- Markdown = simple 'markup' language that uses plain text to create headers, images, links, etc.

Where does R Markdown fit in?



Why is this useful?



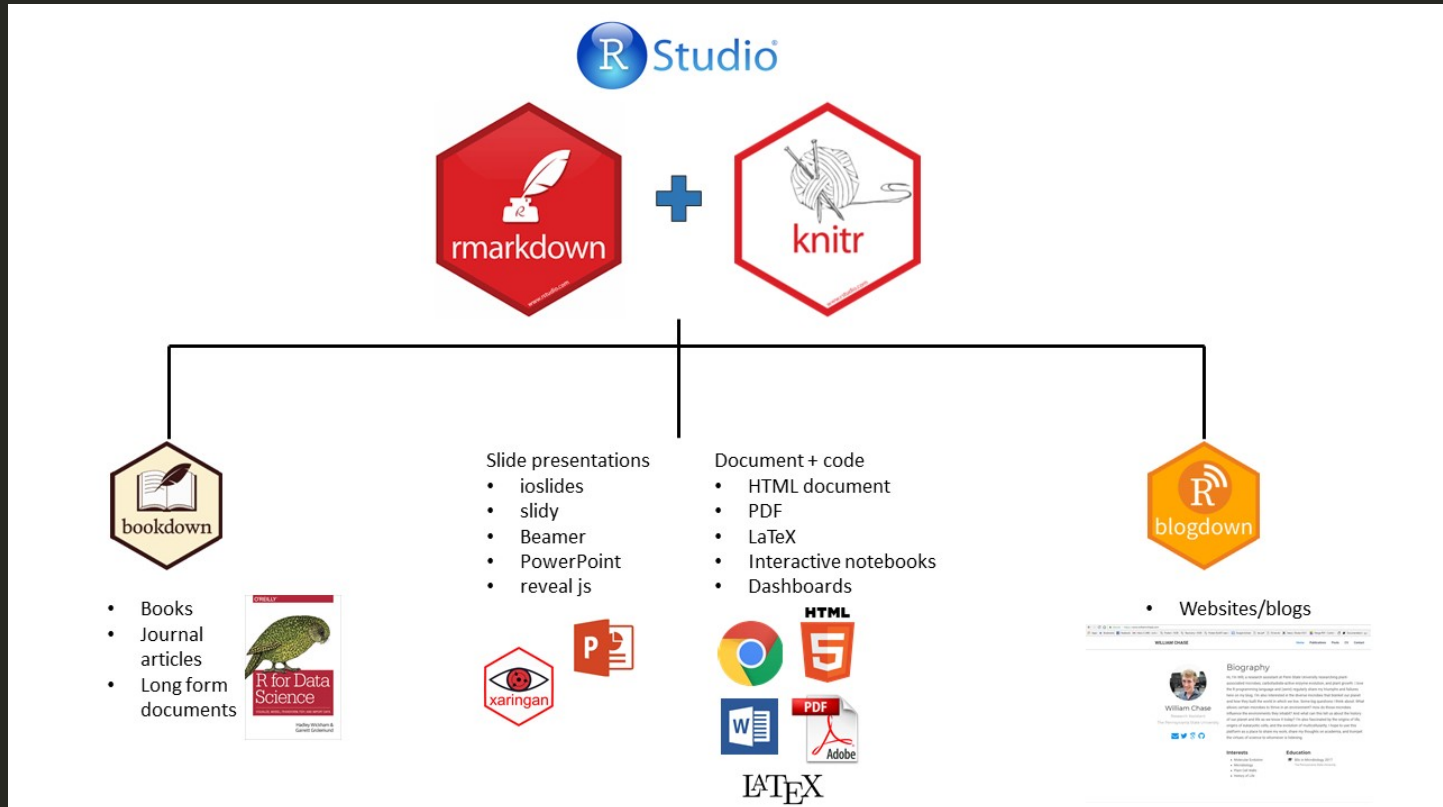
- Fully reproducible
- Narrative text + code
- Multiple languages (40+): R, Python, SQL, ...
- Static and dynamic output formats

Most importantly!

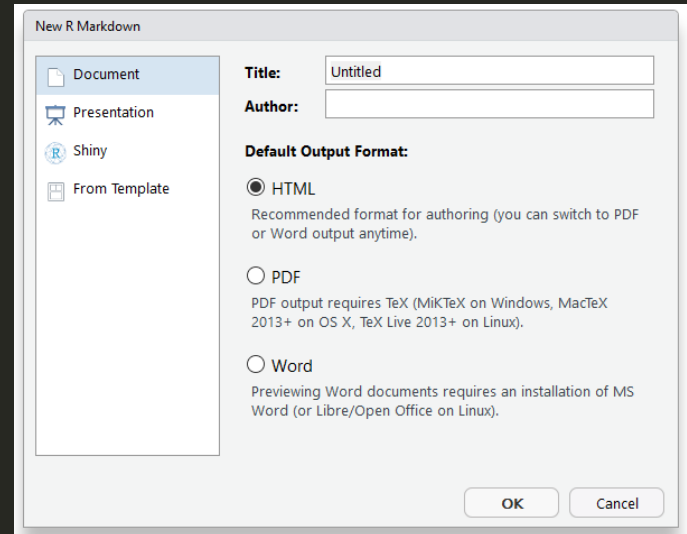
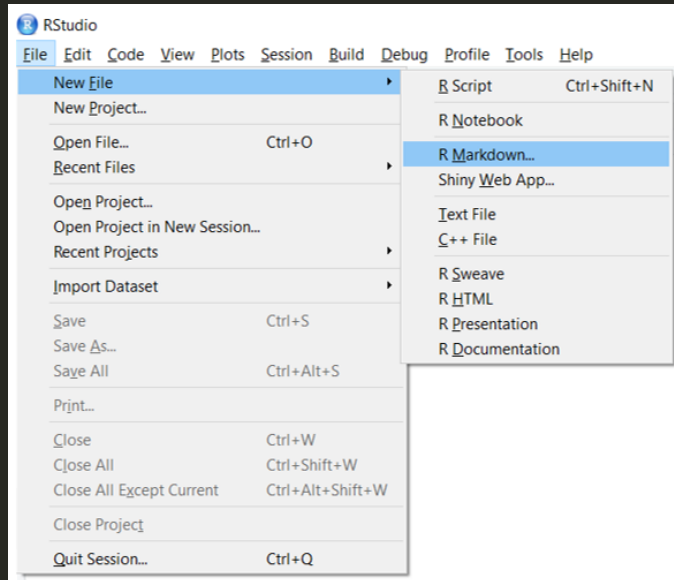


- Many journals are signing the joint DART (Data Access & Research Transparency) initiative
- Requires author(s) to hand over raw data and code that produced results
- Manuscript only published if results are replicated successfully

What else can R Markdown offer?



How-to...



Anatomy of an R Markdown report



YAML header: metadata that is needed to render the document (i.e. produce an output file).

Main body: text, which may include in-line code, written in R Markdown (plain text formatting) syntax.

Code chunks: R (or other) code that can be executed when the document is knit/rendered.

YAML header

Default YAML header

```
---  
title: "How to write a report using R Markdown"  
author: Bianca Peterson  
date: "2019/04/06"  
output: html_document  
---
```

How to write a report using R Markdown

Bianca Peterson

2019/04/06

output: since we chose html format, this will be specified here

There are several other optional and cosmetic items that can be added to the YAML header, such as the font, font size, spacing, margins, themes, styling sheets, etc.

Custom YAML header

```
---
title: "How to write a report using R Markdown"
author:
- Bianca Peterson, Conquest Analytics & Training
- Name Surname, University of Somewhere
date: "`r format(Sys.time(), '%d %B %Y')`"
output:
  pdf_document:
    latex_engine: pdflatex
    keep_tex: true
    toc: true
    number_sections: true
---
```

author: Specify more authors by listing names and affiliations

date: Retrieve date automatically from your system

output: PDF document

- Additional output-level options are specified underneath it, with a 2-space indent
- **latex_engine:** Document will be rendered with LaTeX
- **keep_tex:** The raw `.tex` file will be rendered along with the PDF

toc: A table of contents will be created.

How to write a report using R Markdown

Bianca Peterson, Conquest Analytics & Training
Name Surname, University of Somewhere

26 December 2019

Contents

1	Introduction	1
2	Material and Methods	1
2.1	Method 1	1
2.2	Method 2	1
3	Results	1
4	Discussion	2
5	Conclusion	2
5.1	References	2

1 Introduction

Some text here.

2 Material and Methods

2.1 Method 1

Some text here.

2.2 Method 2

Some more text here.

3 Results

Some text here.

Main body

Markdown

Let's learn some Markdown: click [here](#)

Summary of Markdown

This is a level 1 header

This is a level 1 header

This is a level 2 header

This is a level 2 header

This is a level 3 header

This is a level 3 header

This starts a paragraph block

This starts a paragraph block

> This will create a block quote

■ This will create a block quote

Format text

`*italics*` or `_italics_`

italics or *italics*

`**bold**` or `__bold__`

bold or **bold**

`<u>underline</u>` with the `<u>` HTML tag

underline with the `<u>` HTML tag

`X^superscript^`

$x^{\text{superscript}}$

`X~subscript~`

$x_{\text{subscript}}$

Some text here. `^[This is a footnote]`

Some text here.¹

¹ This is a footnote.

Mathematical expressions

You can write LaTeX math expressions inside a pair of dollar signs.

In-line math expression

`$A = \pi * r^{2}$` renders $A = \pi * r^2$

Display style

```
$$\bar{X}=\frac{1}{n}\sum_{i=1}^nX_i$$
```

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

Note: No spaces after the opening `$` or before the closing `$`.

See [this document](#) for mathematical notation.

Add lists - unordered

- An
- Unordered
- List
 - + Sub-item 1
 - + Sub-item 2

- * Another
- * Unordered
- * List
 - + Sub-item 1
 - + Sub-item 2

- An
- Unordered
- List
 - Sub-item 1
 - Sub-item 2

- Another
- Unordered
- List
 - Sub-item 1
 - Sub-item 2

Add lists - ordered/numerical

1. An
2. Ordered/numerical
3. List
 - + Sub-item 1
 - + Sub-item 2

- a. Another
- b. Ordered/numerical
- c. List
 - + Sub-item 1
 - + Sub-item 2

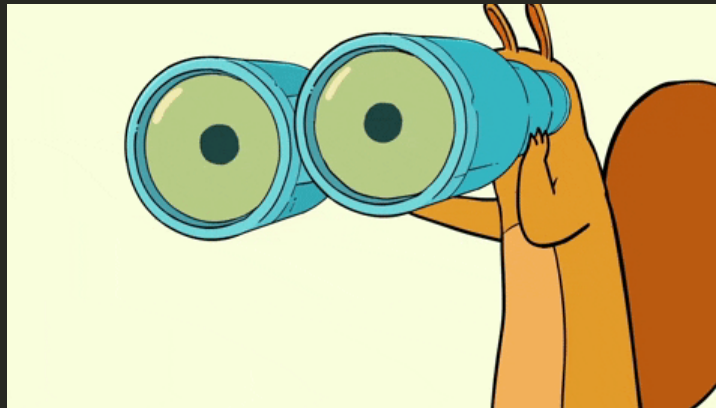
1. An
2. Ordered/numerical
3. List
 - Sub-item 1
 - Sub-item 2

1. Another
2. Ordered/numerical
3. List
 - Sub-item 1
 - Sub-item 2

Hyperlink text

Check out the [RStudio website]
(<https://rstudio.com>) for many great resources.

Check out the **RStudio website** for many great resources.



Inline code

Here is an example of how I am using inline code to specify that today's date is `r format(sys.time(), '%d %B %Y')`.

Here is an example of how I am using inline code to specify that today's date is 24 June 2021.

Code	Meaning	Code	Meaning
%a	Abbreviated weekday	%A	Full weekday
%b	Abbreviated month	%B	Full month
%c	Locale-specific date and time	%d	Decimal date
%H	Decimal hours (24 hour)	%I	Decimal hours (12 hour)
%j	Decimal day of the year	%m	Decimal month
%M	Decimal minute	%p	Locale-specific AM/PM
%S	Decimal second	%U	Decimal week of the year (starting on Sunday)
%w	Decimal Weekday (0=Sunday)	%W	Decimal week of the year (starting on Monday)
%x	Locale-specific Date	%X	Locale-specific Time
%y	2-digit year	%Y	4-digit year
%z	Offset from GMT	%Z	Time zone (character)

Code chunks

Code chunks

```
``` {r chunk_name, warning=FALSE, error=FALSE}

Within the code chunk, it is just like a normal R script

You will have to load your data

my_data <- read.csv("data.csv")

And install and load a library if your are going to use it in your analysis, of course!

install.packages("tidyverse", dependencies = TRUE)
library(tidyverse)

Let's display our dataframe

my_data

Now plot the data

plot(my_data)

```
```

Code chunks Rules/options/arguments

If you want to display the output without displaying the code in the final document, assign additional **rules/options/arguments** for that code chunk

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

Attendees' age differences

attendees <- c("Bianca", "Oliver", "Adele", "Anthony")
age <- c(28, 32, 35, 40)
dataframe <- data.frame(attendees, age)
plot(dataframe)

```
```

Code chunks Rules/options/arguments

| option | default | effect |
|------------|----------|---|
| eval | TRUE | Whether to evaluate the code and include its results |
| echo | TRUE | Whether to display code along with its results |
| warning | TRUE | Whether to display warnings |
| error | FALSE | Whether to display errors |
| message | TRUE | Whether to display messages |
| tidy | FALSE | Whether to reformat code in a tidy way when displaying it |
| results | "markup" | "markup", "asis", "hold", or "hide" |
| cache | FALSE | Whether to cache results for future renders |
| comment | "###" | Comment character to preface results with |
| fig.width | 7 | Width in inches for plots created in chunk |
| fig.height | 7 | Height in inches for plots created in chunk |

See the [R Markdown Reference Guide](#) and [this document](#) for a complete list of knitr chunk options.

Code chunks Language

You can even switch between different **languages** in the same R Markdown document

```
``` {python, include=TRUE, tidy=TRUE, python.reticulate = FALSE}  
print("Hello world!")
```
```

To see which languages (>40) are supported in *knitr*:

```
names(knitr::knit_engines$get())
```

Images & tables

Insert images

1 - Insert figures saved locally - R Markdown

```
![This is the caption for my image](path/to/image.jpg)
```

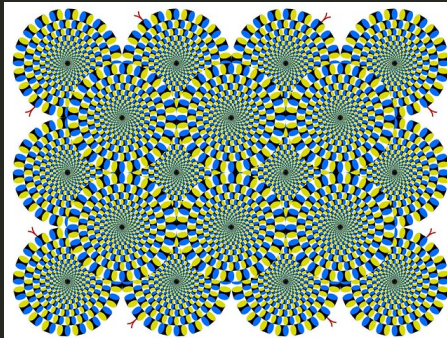
Insert images

2 - Insert figures saved locally - code chunk

```
```{r echo=FALSE, out.width='50%', fig.align='center',  
fig.cap='This is the caption for my image'}

knitr::include_graphics("figures/optical-illusion.jpg")

```
```



This is the caption for my image

Insert images

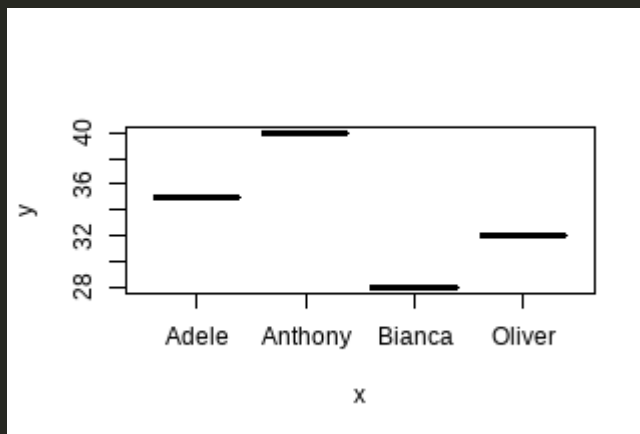
3 - Generate a figure from code and modify the dimensions

```
```{r my_plot, fig.width=6, fig.height=4, include=TRUE, tidy=TRUE}  

Attendees' age differences

attendees <- c("Bianca", "Oliver", "Adele", "Anthony")
age <- c(28, 32, 35, 40)
dataframe <- data.frame(attendees, age)
plot(dataframe)

```
```



Insert tables

1 - Standard R Markdown

```
```{r, include=TRUE}  
dataframe
```
```

| ## | | attendees | age |
|------|--|-----------|-----|
| ## 1 | | Bianca | 28 |
| ## 2 | | Oliver | 32 |
| ## 3 | | Adele | 35 |
| ## 4 | | Anthony | 40 |

Insert tables

2 - Create tables with markdown syntax

```
Table Header	Second Header	Third Header
Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6
```

The position of the colons (":") indicate text-alignment

| Table Header | Second Header | Third Header |
|--------------|---------------|--------------|
| Cell 1 | Cell 2 | Cell 3 |
| Cell 4 | Cell 5 | Cell 6 |

Insert tables

3 - *kable()* function from knitr package

```
```{r include=TRUE}

knitr::kable(dataframe,
 align = c("l", "r"),
 caption = "Ages of attendees",
 format = "html")

```
```

| Ages of attendees | |
|-------------------|-----|
| attendees | age |
| Bianca | 28 |
| Oliver | 32 |
| Adele | 35 |
| Anthony | 40 |

Insert tables

4 - *pander()* function from pander package

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

pander::pander(dataframe,
 caption = "Ages of attendees",
 justify = c("left", "right"))

```
```

Note: *Pandoc* should be pre-installed if you want to use *pander*

Table 1: Ages of attendees

| attendees | age |
|-----------|-----|
| Bianca | 28 |
| Oliver | 32 |
| Adele | 35 |
| Anthony | 40 |

Advanced features

Embed sub-documents

- You can build a document from sub-documents: *knitr* will integrate them into the output.
- Sub-documents (aka child-documents) don't need complete YAML headers.
- Add child documents with a code chunk, and include the document location in the label, e.g.

```
```{r sub-doc1, child="sub-document1.Rmd"}  
...`
```

- Using multiple child documents helps you manage multiple sections in a long report in a convenient and tidy way.

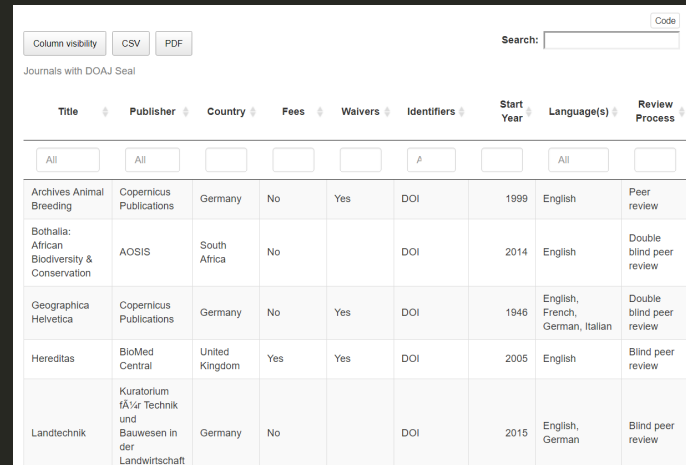
# Add dynamic features

## Code folding

- Add `code_folding: hide` to the YAML header below the `output:` and `html:` lines.
  - HTML: Code will be hidden - **Code tab** available
  - Word & PDF: Code not hidden

## Interactive data table

- Requires the **Datatable (DT)** package
- Only works for HTML output
- User is able to filter and sort data



The screenshot shows a web interface for a table of journals. At the top, there are buttons for 'Column visibility', 'CSV', and 'PDF', and a search bar. Below the buttons, the text 'Journals with DOAJ Seal' is displayed. The table has columns: Title, Publisher, Country, Fees, Waivers, Identifiers, Start Year, Language(s), and Review Process. Each column has a dropdown menu for filtering. The table contains five rows of data:

Title	Publisher	Country	Fees	Waivers	Identifiers	Start Year	Language(s)	Review Process
Archives Animal Breeding	Copernicus Publications	Germany	No	Yes	DOI	1999	English	Peer review
Bothalia: African Biodiversity & Conservation	AOSIS	South Africa	No		DOI	2014	English	Double blind peer review
Geographica Helvetica	Copernicus Publications	Germany	No	Yes	DOI	1946	English, French, German, Italian	Double blind peer review
Hereditas	BioMed Central	United Kingdom	Yes	Yes	DOI	2005	English	Blind peer review
Landtechnik	Kuratorium für Technik und Bauwesen in der Landwirtschaft	Germany	No		DOI	2015	English, German	Blind peer review

# Add parameters

Allow the analysis/report to be **customised** before knitting

- Examples:
  - Show results for a specific geographic location
  - Run a single analysis multiple times for different assumptions

Add the following to the **YAML header**:

```

title: My Document
output: html_document
params:
 year: 2018
 region: "Europe"
 printcode: TRUE
 data: "file1.csv"

```

Add the following to the **main body of text** wherever the year/region/etc. should be inserted:

```
`r params$region`
`r params$year`
```

```
This report is based on `r params$region` for `r params$year`
```

This report is based on Europe for 2018

# Render with parameters

- Parameters can do more than control text.
- You can also use parameters to make changes to graphs, e.g. only plot the number of HIV positive patients in a specific region for a specific timeframe.

```
```{r}

my_data <- read_csv(params$file)

subset <- my_data %>%
  filter(year == params$year,
         region == params$region)

```
```

# Autopopulate your Biosketch from ORCID

## Step 1: Authorise access to ORCID

```
library("rorcid")
library("httpuv")

token <- orcid_auth(scope = "/authenticate", reauth =
FALSE, redirect_uri = getOption("rorcid.redirect_uri"))

res <- orcid_bio(orcid = "0000-0001-6927-9159")
bio <- res$"0000-0001-6927-9159"$content
```

1. A web browser will open at the ORCID web site.
2. Log in to your account as usual.
3. The variable `token` will be added to the RStudio Environment - this is the authorisation code issued by ORCID that allows R and RStudio to read information from your ORCID account).

# Autopopulate your Biosketch from ORCID

## Step 2: Get token value

- In the R CONSOLE, type `token` and press Enter
- Copy and paste the token value (everything after `BEARER`) that was printed to the console



# Autopopulate your Biosketch from ORCID

## Step 3: Create .Renviron file

Run the following code in the **R CONSOLE**

```
if (!file.exists("~/Renviron")) {
 file.create("~/Renviron")
}

file.edit("~/Renviron")
```

1. A new text file (.Renviron) will open in RStudio.
2. Type the following: `ORCID_TOKEN="token value"` and paste your unique token value where `token value` is indicated.
3. Save this file - it will be saved in your computer's home directory.
4. This file will be read every time you start R, thus enabling RStudio to communicate with ORCID to refresh or add data from your profile.

# Autopopulate your Biosketch from ORCID

## Step 4: Insert your ORCID bio

Place the following code in your **.Rmd** file

```
`r bio`
```

When you knit the document, your biography from your ORCID profile will appear in the reproducible report.

Generating the  
output file

# Compile .Rmd file

1 - Using the interface: Press the Knit button in the taskbar

2 - Programmatically: `rmarkdown::render("Path/to/File.Rmd")`

R will:

- Execute each embedded code chunk and insert the results into your report
- Build a new version of your report in the output file type (in this case HTML)
- Open a preview of the output file in the viewer pane
- Save the output file in your working directory

Note: TeX is required to create PDF output:

- Windows: MiKTeX (Complete)
- Mac OS X: MacTeX.pkg
- Linux: Use system package manager

# Compile .Rmd file with parameters

## 1 - Specify parameters in code

```
rmarkdown::render("MyDocument.Rmd",
 params = list(year = 2017,
 region = "Asia",
 printcode = FALSE,
 file = "file1.csv"))
```

## 2 - Render interactively

```
rmarkdown::render("MyDocument.Rmd", params = "ask")
```

|                                               |
|-----------------------------------------------|
| year                                          |
| <input type="text" value="2018"/>             |
| region                                        |
| <input type="text" value="Europe"/>           |
| <input checked="" type="checkbox"/> printcode |
| data                                          |
| <input type="text" value="file1.csv"/>        |

# Publishing your reproducible report

# Publish via GitHub

- Create / Log in to your GitHub account.
- Create a new repository by initialising the repo with a README file.
- On your Desktop: Navigate to the `.html` file you've just knitted, and rename to `index.html`.
- On GitHub:
  - Click **Upload a file** - drag the `index.html` file to the box, then click the green button at the bottom.
  - Go to settings, scroll down to the **GHPages** section, click the dropdown and select **master**.
  - Scroll to the **GHPages** section and copy the github.io URL.
  - Go back to the code section of the GitHub repo and add the URL. If you click on this link, you'll see your HTML report.

■ Note: you can add the `.Rmd` file to the GitHub repo and do version control via RStudio!

# Publish via **R**Pubs

- Open your .Rmd document in RStudio and click **knit**.
- In the item view pane, click on **Publish**. You'll be redirected to the rpubs web site. You'll need to register if you don't have an rpubs account.
- Your document is now published online.



# Publish via binder

- Copy all files used by knitr into your GitHub repo, but make a copy of your .Rmd file that doesn't include the ORCID section (the authentication is complex).
- On GitHub:
  - Create a `runtime.txt` file that contains: `r-2018-07-09`. This chooses a version of R at the date indicated.
  - Create an `install.R` file that contains the following code:

```
install.packages("rmarkdown")
install.packages("bitops")
install.packages("caTools")
install.packages("tidyverse")
install.packages("DT")
```

# Publish via

- Edit the **README** file in Github and add: [<https://mybinder.org/v2/gh/github-username/github-repo/master?urlpath=rstudio>]  
Replace **github-username** with your GitHub account name and **github-repo** with your GitHub repository name.
- Click the Binder button and your environment will load (it might take up to 20 mins the first time).

# Writing manuscripts

# Journal styles/templates

```
Install from CRAN
install.packages("rticles")

Or install development version from GitHub
devtools::install_github("rstudio/rticles") # recommended

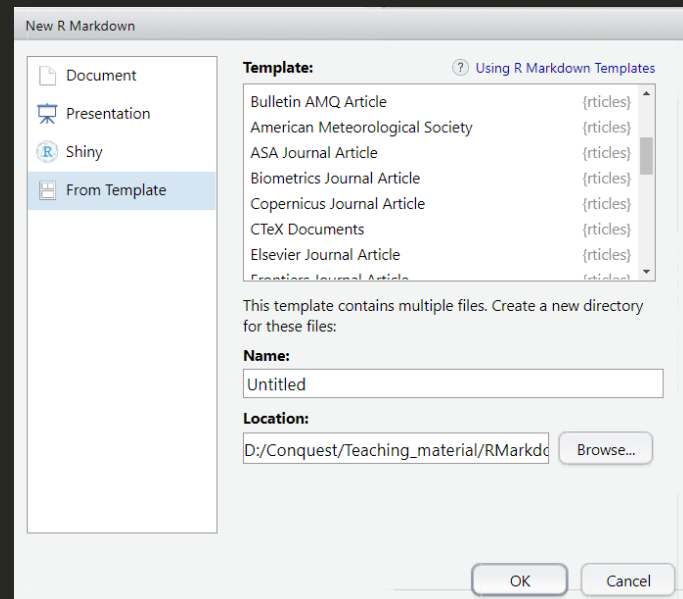
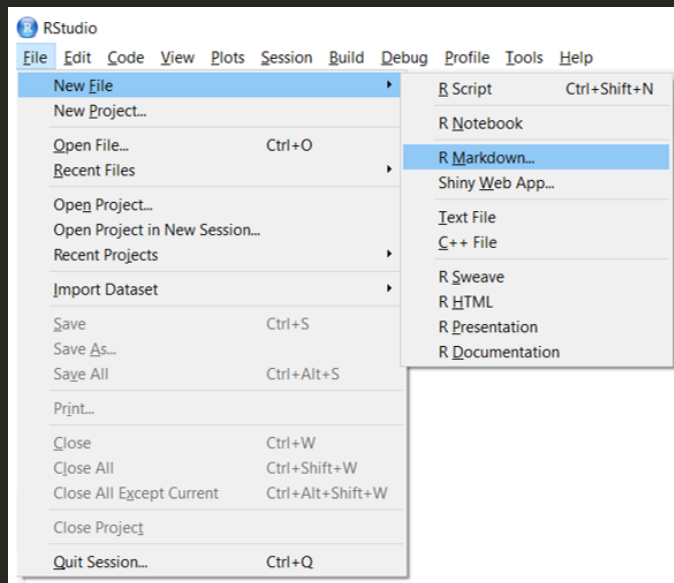
Load package
library(rticles)

Show full list of available journal templates
getNamespaceExports("rticles")
```

The **rticles** package (J. Allaire, Xie, R Foundation, et al. 2019) is designed to simplify the creation of documents that conform to submission standards.

# How to access the templates

## 1 - Using the interface



# How to access the templates

## 2 - Programmatically

```
rmarkdown::draft("MyArticle.Rmd",
 template = "elsevier_article",
 package = "rticles")
```

- A new R Markdown file will be created, which will contain
  - an extended YAML section compared to the basic R Markdown template, and
  - some guidelines/instructions.
- You can add text, code chunks, images, tables, etc.

# A few things to keep in mind

- Although most journal styles are supported, it is not possible to match the journal's layout exactly
- Check your journal's Guide for Authors for any journal-specific formatting requirements
- Additional class files and templates may be available for more complex articles (e.g. single-column and double-column articles)
- Some journals require a specific reference style, for which the relevant bibliographic style files for LaTeX may be downloaded from journal websites

# Importing references into your Mendeley library

## Option 1: Get references in BibTeX format via Google Scholar

The screenshot shows the Google Scholar interface with the search query "transcriptome busseola fusca". The search results list several articles. A red box highlights the citation icon (a small 'C' in a circle) next to the article "Transcriptome and differentially expressed genes of *Busseola fusca* (Lepidoptera: Noctuidae) larvae challenged with Cry1Ab toxin." by Peterson, B., Sanko, T. J., Bezuidenhout, C. C., & Van den Berg, J. (2019). A red arrow points from this citation icon to the "BibTeX" option in the citation window that is open over the article. The citation window shows the citation in various formats: MLA, APA, Chicago, Harvard, Vancouver, and BibTeX. The BibTeX format is highlighted with a red box.

Google Scholar search results for "transcriptome busseola fusca". The citation window is open, showing the citation in various formats. The BibTeX format is highlighted.

**Cite**

MLA Peterson, Bianca, et al. "Transcriptome and differentially expressed genes of *Busseola fusca* (Lepidoptera: Noctuidae) larvae challenged with Cry1Ab toxin." *Gene* 710 (2019): 387-398.

APA Peterson, B., Sanko, T. J., Bezuidenhout, C. C., & Van den Berg, J. (2019). Transcriptome and differentially expressed genes of *Busseola fusca* (Lepidoptera: Noctuidae) larvae challenged with Cry1Ab toxin. *Gene*, 710, 387-398.

Chicago Peterson, Bianca, Tomasz Janusz Sanko, Cornelius Carlos Bezuidenhout, and Johnnie Van den Berg. "Transcriptome and differentially expressed genes of *Busseola fusca* (Lepidoptera: Noctuidae) larvae challenged with Cry1Ab toxin." *Gene* 710 (2019): 387-398.

Harvard Peterson, B., Sanko, T.J., Bezuidenhout, C.C. and Van den Berg, J., 2019. Transcriptome and differentially expressed genes of *Busseola fusca* (Lepidoptera: Noctuidae) larvae challenged with Cry1Ab toxin. *Gene*, 710, pp.387-398.

Vancouver Peterson B, Sanko TJ, Bezuidenhout CC, Van den Berg J. Transcriptome and differentially expressed genes of *Busseola fusca* (Lepidoptera: Noctuidae) larvae challenged with Cry1Ab toxin. *Gene*. 2019 Aug 20;710:387-98.

**BibTeX** EndNote RefMan RefWorks



# Importing references into your Mendeley library

Option 2: This [video](#) illustrates how to use the [Mendely Web Importer](#) to import references directly from the internet into your online Mendeley library.

# Export Mendeley references to BibTeX

- First sync your **Online Mendeley Library** with your **Desktop Mendeley Library**.
- Open Mendeley Desktop and select the files you wish to export.
- Go to the **File** menu and select **Export**.
- Save as **BibTeX** type and save to the same folder that contains your Article.Rmd file.

# Create bibliography in your .Rmd file

## In-text citations

`@key` or `[@key]` where `key` is the citation key assigned (in the .bib files) to the source you want to cite (e.g. `@smith1994`).

`@smith1994` will produce `Smith (1994)`

`[@smith1994]` will produce `(Smith, 1994)`

*Note:* you should have 2 .bib files - one for references and one for packages.

- `references.bib` should contain references exported from Mendeley.
- `packages.bib` can be generated by the following command:

```
write_bib(c("tidyverse", "rmarkdown", "knitr", "pander",
"DT", "rorcid"),
 file = "packages.bib")
```

# Create bibliography in your .Rmd file

## References

A reference list will be automatically generated at the end of the document according to the citation style indicated in the YAML header.



# Submitting your manuscript

- Most journals accept a PDF of your manuscript at initial submission.
- When you are asked to submit your manuscript source files, do the following:
  - Build a PDF of your manuscript source files on your computer and attach it with item type 'Manuscript'.
  - Bundle all manuscript source files in a single archive (.zip) and attach it with item type 'LaTeX source files'.
  - Source files include LaTeX files, BibTeX files, figures, tables, all LaTeX classes and packages that are not included in TeX Live and any other material that belongs to your manuscript.

# Disadvantages

# Not as intuitive as Microsoft Word



moving an image  
slightly to the left

all text and images realign,  
4 new pages appear,  
earth orbit shifts by 2 meters,  
sirens in the distance

VIA 9GAG.COM

# It doesn't have clippy

**Did you know...**

By inserting a picture into this document I will be forced to rearrange everything you've done in the past hour the way I see fit!





# Acknowledgements

Slides created via the R package **xaringan**.

## Material compiled from:

- The **Coding Club: Getting Started with R Markdown** licensed under a **Creative Commons Attribution-ShareAlike 4.0 International License**
- **Author Carpentry** licensed under a **Creative Commons Attribution 4.0 International License**
- **R for Data Science, Chapter 27**
- **R Markdown: The Definitive Guide**
- **RStudio website**
- **RStudio lesson**
- **An R Markdown Template for Academic Manuscripts** written by Steven V. Miller (**@svmiller** on GitHub) with written permission
- **RMarkdown for writing reproducible scientific papers** written by Mike Frank & Chris Hartgerink