reworking your workflow

combining data analysis and reporting with dynamic documents

Christof Neumann

RTG Understanding Social Relationships

2021-11-18

content

- dynamic documents, huh?
- what is R markdown and why should we care?
- the dark side of a dynamic workflow
- let's get our hands dirty
- my assumptions:
    - you have used `R` before and use RStudio
    - you want to produce pdfs (article, thesis, presentation) or html (presentation, websites, `shiny`)
- adjust your expectations:
    - we scratch on the surface
    - always keep in mind what you **need** (and not what you **can do**)
    - fine line between what I would consider a 'good' presentation and showcasing the capabilities of `R` markdown

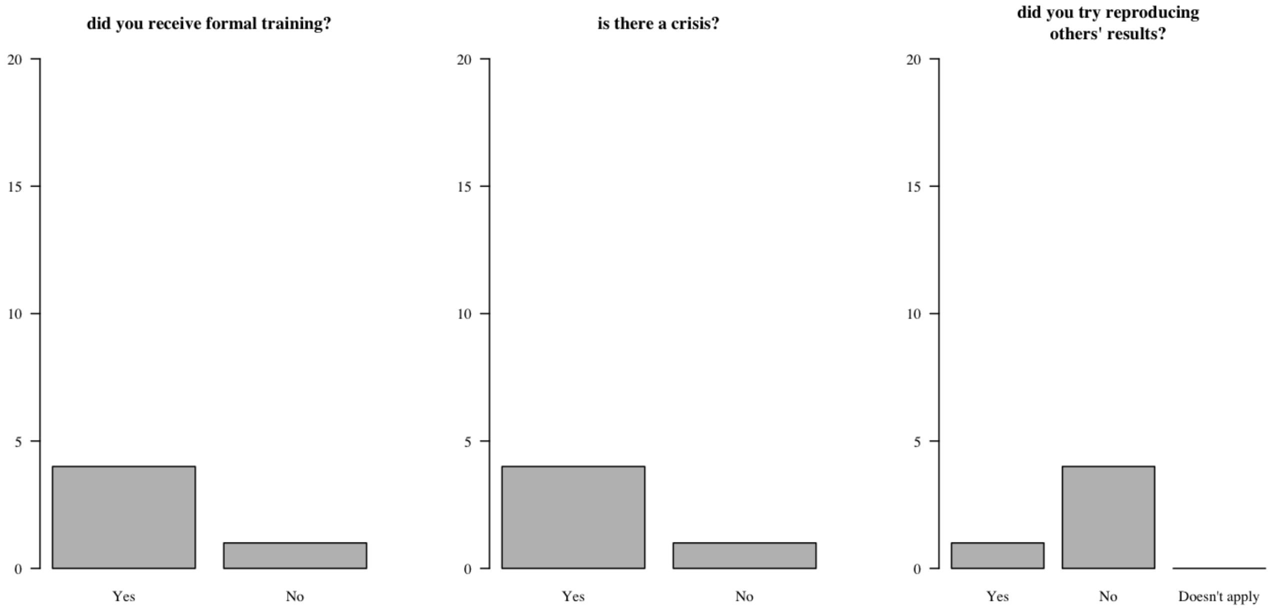
dynamic documents

dynamic (or living) documents

- continuous updates, revisions and edits (think Wikipedia)
- as opposed to static (dead) document (think printed lexicon)
- in scientific context:
  - preprints
  - during write-up
- can you do good and transparent science without it? YES.

a truly dynamic document

- this example grabs the data directly from the survey website



background

- to replicate

> obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data.

- to reproduce

> obtaining consistent results using the same input data, computational steps, methods, and conditions of analysis

https://doi.org/10.17226/25303

reproducibility

> If someone applies the same analysis to the same data, then the same result should recur.

- process failure:
  - no results obtainable
  - unavailability of data and code
  - incomplete information about the analysis, software or tools
- outcome failure:
  - different results
  - error in original study or in reproduction attempt

https://doi.org/10.1146/annurev-psych-020821-114157

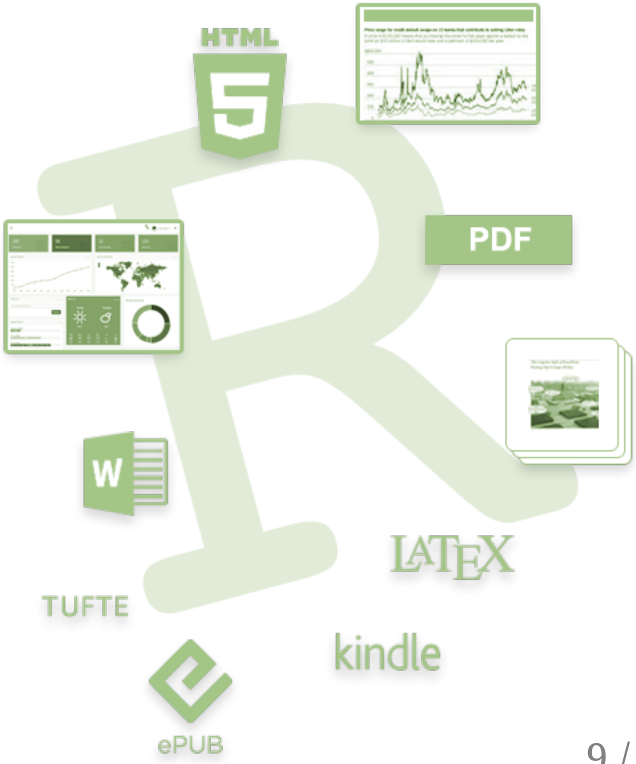
what is R markdown and why should we care?

R markdown?

- **markdown** is a programming language for converting plain text into formatted text

- **R markdown** builds on this and allows producing formatted output that builds on R code

- by now, the `rmarkdown` universe allows production of
  - text documents (journal articles, theses, books)
  - presentations
  - websites (via blogdown)
  - interactive applications/dashboards (via shiny)
  - exotic things (epub, docx, pptx, package documentation)



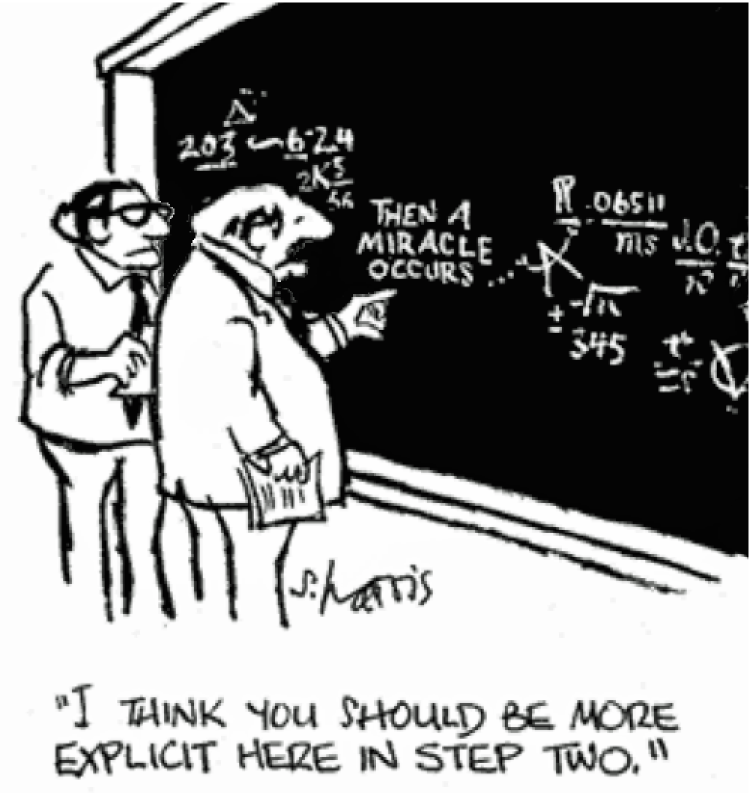
R markdown?

- all these have in common:
    - a shared logic and style for mixing written text and results from analyses
    - full reproducibility (at least in theory)
    - being open source and having an active community
    - still a high degree of flexibility
    - a certain aesthetic (it just looks good, no?)

R markdown!

- self-contained
- reproducible
- robust
- it's what the cool kids do these days (thanks, Roger)
- but there are downsides, too (we'll get to that)

why again?



I can't find the origin of this cartoon. If you know, please let me know.

your most important collaborator is your future self!!!

when does it get complicated?

- it might very well…
- actually, it's likely to happen sooner rather than later
- long execution time (think bootstraps, models in `stan`)
  - sending workspaces to Roger
  - save workspace or not? it depends
- collaborators/supervisors/students won't/can't be bothered
- (in)dependence of RStudio
- environment [R version, package versions] at time of writing (when it works) versus time of production [which is the same as writing but also may be months or years later (reproducibility)]
- subtle differences (e.g. this presentation format doesn't support references and only very simple tables)

the nitty and the gritty

markdown (markup language)

- allows **plain text** with **simple formatting** to be translated into other formats
- independent of `R`, RStudio and $\LaTeX$
- .md

```
Heading
=======

Sub-heading
-----------

Paragraphs **are** separated
by a blank line.
```

https://www.markdownguide.org/

R markdown

- extension of markdown
- allows combining markdown ('text', 'narrative') and R ('code')
- still plain text
- independent of LaTeX and RStudio (although the latter provides lots of convenience)
- .Rmd

~~~
We can combine markdown and `R` code.
Backticks are really important here.

```{r, fig.cap = "My results."}
plot(1:10)
```
~~~

https://rmarkdown.rstudio.com/

`knitr` (and `rmarkdown`)

- are R packages
- `knitr` takes the code chunks, runs them and 'knits' them back into markdown
- requires R, but its results are independent of R (because they are markdown)
- `rmarkdown` and `knitr` also provide several convenience functions

https://yihui.org/knitr/

Pandoc

- document converter
- transforms .md into .html/.tex
- can also do tons of other conversions (-> MS Office, LibreOffice, …)
- independent of `R` and RStudio

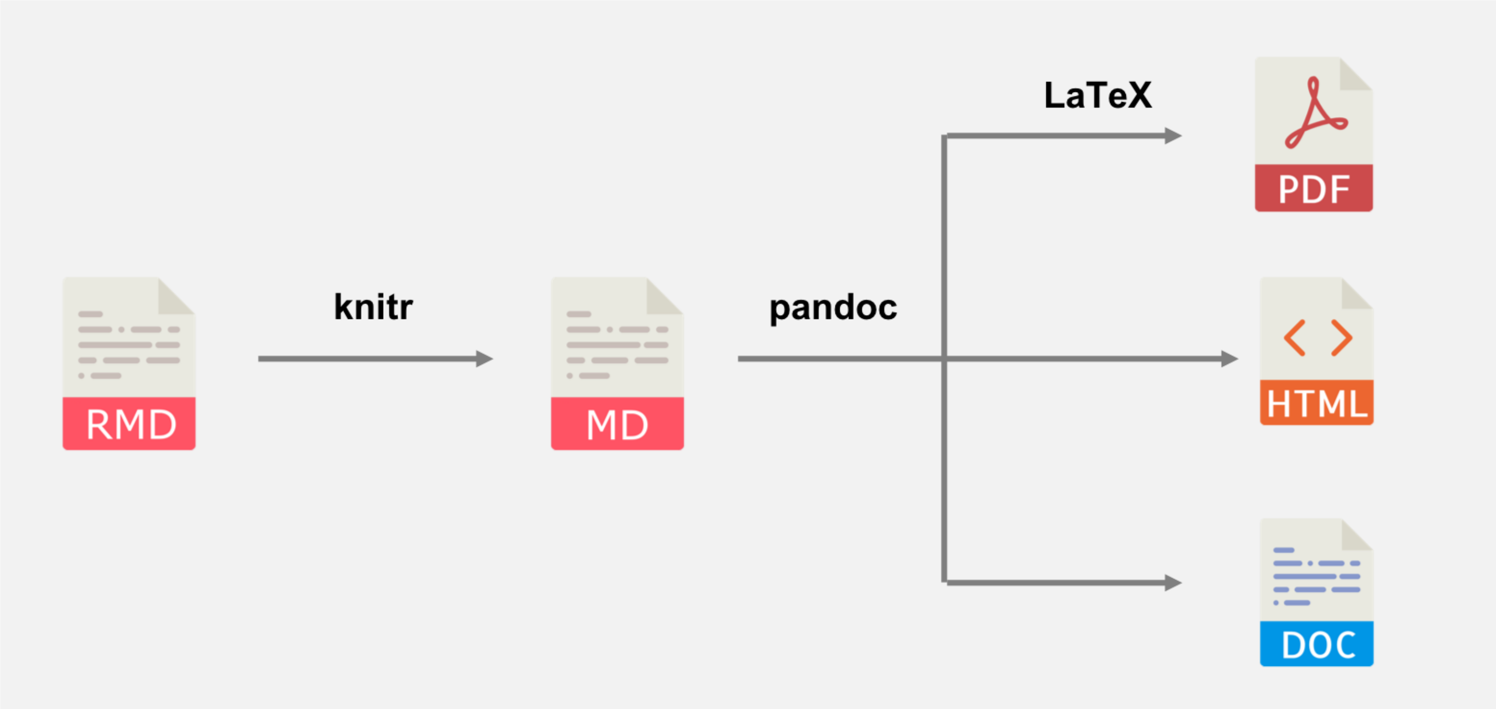
https://pandoc.org/

LaTeX

- builds .pdf files (article, thesis, presentation) from .tex files
- "a software system for document preparation"
- "separating presentation from content"
- independent of `R` and RStudio

https://www.latex-project.org/

in a nutshell



https://bookdown.org/yihui/rmarkdown-cookbook/

getting to it

prep

- create a new RStudio project
- create a new `R` markdown file
- memorize some keyboard shortcuts
    - cmd/ctrl + shift + k: build document
    - cmd/ctrl + option/alt + i: new R chunk
    - cmd/ctrl + shift + enter: run current chunk
    - cmd/ctrl + shift + 0: restart `R`

`yaml` header

- contains meta data for the final document (author, title, date)
- determines output format (here: `html_document`)
    - possibly with options (table of contents, highlighting options)
    - if we get to it, we might also try `pdf_document`
- other options:
    - control RStudio's behavior
    - bibliography data, reference style
    - line numbering and line spacing
- we won't look into this much, and mostly rely on the default options RStudio provides

`yaml` header

```
---
title: "Untitled"
author: "Christof Neumann"
date: "9/29/2021"
output:
  html_document:
    highlight: tango
editor_options:
  chunk_output_type: console
---
```

a simple paragraph

```
# Introduction

A short paragraph that contains a summary of the article, for
example the background and some major results. And note that
the header 'Introduction' was formatted as a header because
we put a \# in front of it (which is visible only in the source
code of the document).
```

Introduction

A short paragraph that contains a summary of the article, for example the background and some major results. And note that the header 'Introduction' was formatted as a header because we put a # in front of it (which is visible only in the source code of the document).

simple formatting

```
Text can be **bold** or in *italics*.
But we can also use ~~strike through~~.
Underlining is trickier and requires html code.
```

Text can be **bold** or in *italics*. But we can also use ~~strike through~~. Underlining is trickier and requires html code.

a first code chunk

~~~
This is just some text about how to generate random numbers in R:

```{r}
rnorm(3)
```

Easy, no?
~~~

This is just some text about how to generate random numbers in R:

```r
rnorm(3)
```

```
## [1] -0.61901958 -0.05424848 -0.04937414
```

Easy, no?

more on code chunks

~~~
```{r}
x <- rnorm(1000)
hist(x)
```
~~~

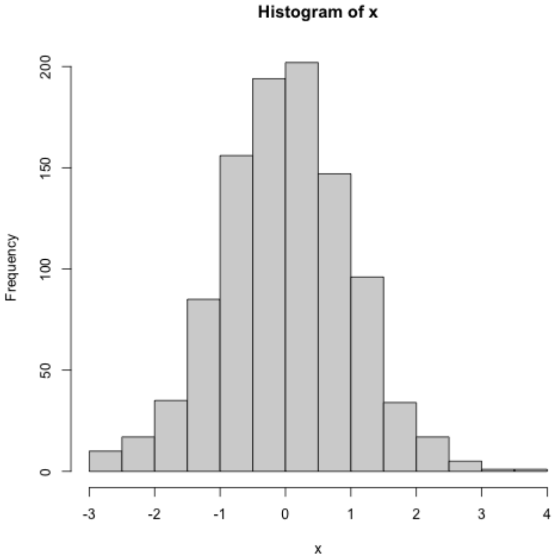
```r
x <- rnorm(1000)
hist(x)
```

more on code chunks

- use chunk options

~~~
```{r, out.width = "40%", fig.align = 'center'}
x <- rnorm(1000)
hist(x)
```
~~~

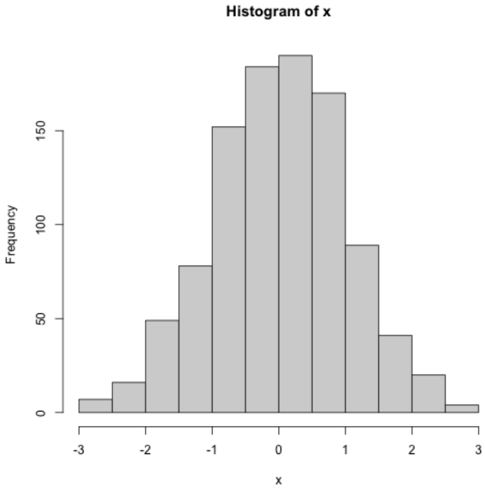
```r
x <- rnorm(1000)
hist(x)
```



more on code chunks

- hide the actual code, and add a caption (which is desired in many situations)

~~~
```{r, out.width = "35%", fig.align = 'center', echo = FALSE,
fig.cap = "This is the figure caption."}
x <- rnorm(1000)
hist(x)
```
~~~



This is the figure caption.

using R objects in the actual text part

~~~
```{r, echo = FALSE}
data("cars")
ncars <- nrow(cars)
mycor <- cor(cars$speed, cars$dist)
```

The cars data set has `r ncars` observations.
And the correlation between speed and distance is `r round(mycor, 2)`.
~~~

The cars data set has 50 observations. And the correlation between speed and distance is 0.81.

equations

```
We can also write fancy equations like $x = \sqrt{y} + c^2$, if they
are wrapped in dollar signs ($). But equations can also span
multiple lines, with a 'proper' math environment:

$$
x = \sum{\frac{42 + y}{log(z)}}
$$

Fancy, isn't it?
```

We can also write fancy equations like $x = \sqrt{y} + c^2$, if they are wrapped in dollar signs ($). But equations can also span multiple lines, with a 'proper' math environment:

$$
x = \sum{\frac{42 + y}{log(z)}}
$$

Fancy, isn't it?

tables

- are a bit trickier
- simple tables are possible with `knitr::kable()`
- more complex tables require the package `xtable`

~~~
```{r, echo = FALSE}
data("airquality")
knitr::kable(head(airquality))
```
~~~

Ozone	Solar.R	Wind	Temp	Month	Day
41	190	7.4	67	5	1
36	118	8.0	72	5	2
12	149	12.6	74	5	3
18	313	11.5	62	5	4
NA	NA	14.3	56	5	5
28	NA	14.9	66	5	6

references and bibliography

- (can't be demonstrated in this particular presentation format, bummer)
- requires `bibtex` references
- bibtex files are plain text files
- probably any reference manager can import/export references in bib(la)tex style (Zotero, Mendeley, Endnote)
- in my workflow, I manage my references with Zotero, and export regularly a .bib file with all the references I ever cited at some point, which I then use for citations in R markdown documents
- check out the "Better Bib(La)TeX" plugin for Zotero (https://github.com/retorquere/zotero-better-bibtex)
- a multi-platform graphic reference manager for bib(la)tex files is JabRef (http://www.jabref.org/)
- style sheets for different journals (.csl files) can be found here: https://www.zotero.org/styles (or here: https://github.com/citation-style-language/styles)

references and bibliography

- a typical bibtex entry for a journal article and book chapter (on the next slide) looks like this
- the crucial thing here is the citation key, i.e. the identifier (`forstmeier2011` in the first example)
- you need the citation key to cross-reference

```
@article{forstmeier2011,
  title = {Cryptic Multiple Hypotheses Testing in Linear Models:
    Overestimated Effect Sizes and the Winner's Curse},
  volume = {65},
  doi = {10.1007/s00265-010-1038-5},
  journaltitle = {Behavioral Ecology and Sociobiology},
  date = {2011},
  pages = {47--55},
  author = {Forstmeier, Wolfgang and Schielzeth, Holger}
}
```

references and bibliography

- here is an entry for a book chapter

```
@incollection{garamszegi2014,
  location = {Berlin},
  title = {Multimodel-Inference in Comparative Analyses},
  booktitle = {Modern Phylogenetic Comparative Methods and
    Their Application in Evolutionary Biology},
  publisher = {Springer},
  date = {2014},
  pages = {305--331},
  author = {Garamszegi, László Zsolt and Mundry, Roger},
  editor = {Garamszegi, László Zsolt}
}
```

references and bibliography

- update the `yaml` header to point to .bib file (biblographic entries) and .csl file (formatting style)

```
---
title: "Untitled"
author: "Christof Neumann"
date: "9/29/2021"
output:
  html_document:
    highlight: tango
editor_options:
  chunk_output_type: console
bibliography: data_files/refs.bib
csl: data_files/animal-behaviour.csl
---
```

references and bibliography

```
Blah blah [@forstmeier2011; @garamszegi2014].

Blah blah [see @forstmeier2011, p. 365; but see @garamszegi2014].

@forstmeier2011 say something.
```

- bibliography is inserted after the header `# references`

wrapping up

going further

- cross-references (to figures and tables)
- more table and plotting options
- conditional inclusion of chunks
- actually using $\LaTeX$
- caching of code chunks
- document formatting/layouts (e.g. thesis chapters)
- extended referencing (bibliographic)
- integrating other coding languages (Python, `stan`, `C++`, …)
- put it all under version control

wrapping up

- establish good practice
- you can't reproduce what doesn't exist
- think about your workflow
- your single most important collaborator is your future self