standardized_markdown

Jordy van Langen

December 10, 2019

The following contains a standardized workflow following best practices in the open science community (2019).

1. First steps

- Give a meaningful title to your markdown document.
- Change the default date - for automaticity - to: `"r Sys.Date()"` or `"r format(Sys.time(), "%B %d, %Y")"`
- Add 'urlcolor' if you want to highlight hyperlinks in a color of your choice in the rendered document. An example of a standardized markdown header:

```
---
 title: "standardized_markdown"
 author: "Jordy van Langen"
 date:   '`r format(Sys.time(), "%B %d, %Y")`'
 output:
  pdf_document: default
  html_document: default
urlcolor: blue
---
```

2. R-version check

- Create an R chunk that prints the current R version you are using.

```
R.version$version.string
```

```
## [1] "R version 3.6.1 (2019-07-05)"
```

3. Loading libraries

- Create an R chunk that loads all the libraries you are using.

```
library("ggplot2") #Version 3.2.1
library("here") #Version 0.1

packageVersion()
sessionInfo()
```

- Always make sure to install the latest version of a package when performing your analysis. You can manually check the version of each package with `packageVersion()`.
- `sessionInfo()` also gives you a lot of information about packages installed and your R version used, however, this might not be desired since you will receive a lot of information that is not always of interest.

- An alternative might be the `packrat` package, developed by Rstudio. This is a solid way to manage the R packages your project depends on in an isolated, portable, and reproducible way.

4. Importing data

- When loading data from your computer (i.e., not directly from an url), use the `here` package Müller, 2017. This offers a robust solution to make sure that file access does not break accross different computing platforms.A detailed description of the use of this package is described in: A Reproducible Data Analysis Workflow with R Markdown, Git, Make, and Docker, 2019.
- Use the following folder structure:

```
projectname (your project folder)

    data (folder containing your data)
    R (folder containing your code)
```

- An example on how to import data files, taken and adjusted from A Reproducible Data Analysis Workflow with R Markdown, Git, Make, and Docker, 2019.

Not robust

```
NHANES <-read.csv("/Users/jordyvanlangen/standardized_markdown/data/NHANES.csv")
```

Not robust

```
NHANES <-read.csv("data/NHANES.csv")
```

Robust

```
NHANES <-read.csv(here("data", "NHANES.csv"))
```

Robust

```
NHANES_df <- read.csv(here("data", "NHANES.csv"))
```

- Personal remark to this example: I would add `_df` to `NHANES` to indicate that you are working with a dataframe. This is also advised by the Statistical thinking for the 21st century course by Russel Poldrack (2019).

5. Data pre-processing and analysis

- At this stage you can start writing your analysis code.

6. Knitting

- You can knit your markdown to an HTML file, however that does not look really 'fancy' and not 'publication-ready'. Therefore, knitting to PDF might be a better option.
- When you want to knit your .Rmd to pdf, you'll need to have `LaTeX` installed. For R Markdown users who have not installed `LaTeX` before, it is recommended that you install the `TinyTex` package by YiHui Xie.

```
install.packages("tinytex")
```

7. Appendix:

The appendix includes an amount of useful sources that might improve your R (markdown) workflow.

- Check the R Markdown cheatsheet for the extensive possibilities that R markdown offers.
- There is the possibility of writing your whole paper (reported stats, tables, figures) in markdown complied APA style with the `papaja` package by Frederik Aust, 2019.
- The `redoc` package by Noam Ross, 2019: enables a two-way R Markdown-Microsoft Word workflow. It generates Word documents that can be de-rendered back into R Markdown, retaining edits on the Word document, including tracked changes.
- 4 key-points that researchers should take into account when sharing data. See Analysis of Open Data and Computational Reproducibility in Registered Reports in Psychology, 2019.
- Power analysis: although G*Power is still being used predominantly, it does not suit reproducible research. Therefore, a good alternative is to perform your power-analysis in R. For simpler power-analysis (e.g. t-tests) you can find a good tutorial from Dan Quintana (2019) on YouTube in which he explains the use of the `pwr` package Helios de Rosario, 2019. For more extensive, simulation-based, power-analysis have a look at the `ANOVApower` package by Daniël Lakens, 2019.
- If you want to unload a package and use a clean version of it again, you can use the following code:

```
if("package_name" %in% (.packages())){
  detach('package:package_name', unload=TRUE)}

library("package_name")
```

This workflow was made by Jordy van Langen, December 10, 2019.

If you have suggestions on how to improve, feel welcome to send a message to jordy.vanlangen@sydney.edu.au or contact me on Twitter jordyvanlangen