Alexander Strobel
Christoph Scheffel
Josephine Zerna

Collaboratively working on reproducible manuscripts via R Markdown and GitHub

Outline

R

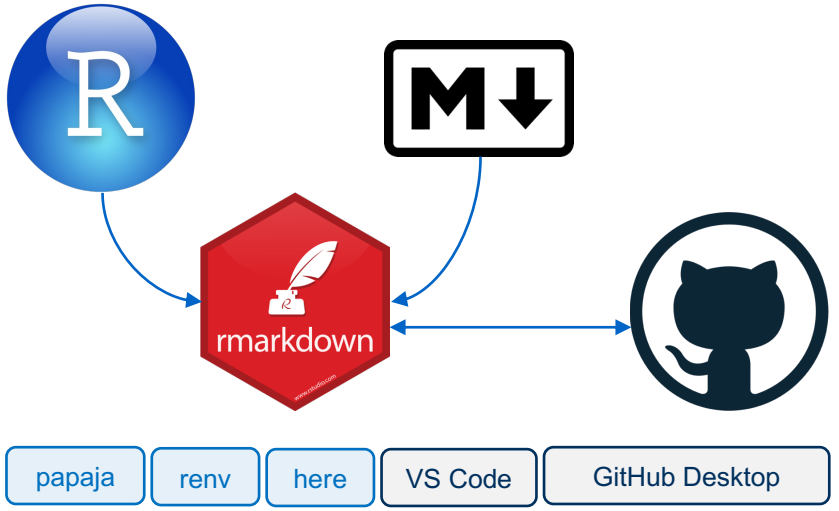
Markdown

R Markdown

GitHub

Recommended R packages

Recommended tools



Ultimate Goal

provide open data and code to ensure transparency

— effectively
— efficiently
— reproducibly

3. Open Analysis:
(a) Whenever possible, we publish, for every first-authored empirical publication, reproducible data analysis scripts ...

R and RStudio
Overview

R

open environment (not only) for statistical computing

easy to learn given basic statistical and programming skills

allows to provide open code along with your data

RStudio

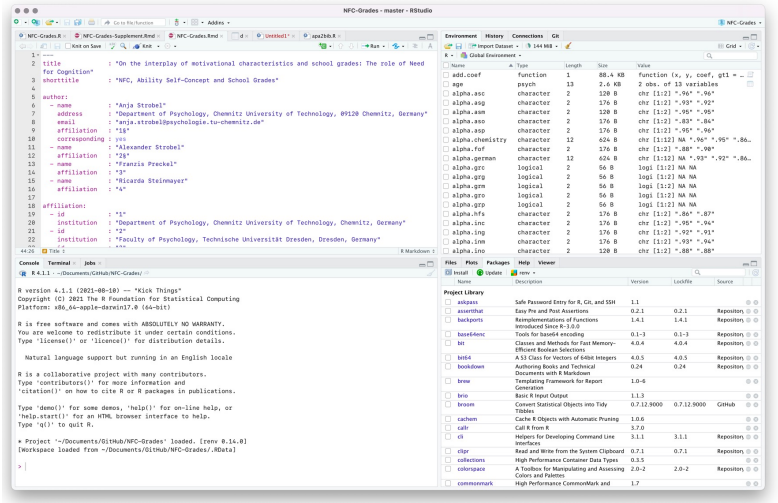
frontend to R (plus has prettier program icon)

comes with a number of benefits compared to stand-alone R including, but not limited to

— a graphical user inferface

— many built-in features incl. R Markdown

— GitHub and Python integration



https://www.r-project.org | https://www.rstudio.com

Markdown
Overview

Markdown

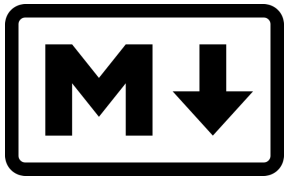
simplified markup language (i.e., an easy to learn alternative to HTML)

allows to format documents in an easy way, e.g.

```
# Heading 1
## Heading 2
*italic*
**bold**
> quote
```

````
Code block
```
````

```
$a^2 + b^2 = c^2$
```



Heading 1

Heading 2

italic

bold

> quote

```
code block
```

$$a^2 + b^2 = c^2$$
```

R Markdown
Overview

R Markdown

combines the benefits of R and Markdown (and LaTeX), i.e., one writes text in markdown and includes R code in so-called R code chunks, e.g.

```
# Method

We report how we determined our sample size, all data exclusions (if any), all
manipulations, and all measures in the study.

```{r correlation, echo = F}
set.seed(242)
X = rnorm(64)
Y = .5 * X + rnorm(64)
cor_XY = cor.test(X, Y)
Pval <- function(p) {
  if (p < .001) { pchar = '$p<.001$' }
  else { pchar = paste0('$p=', format(round(cor_XY$p.value, 3), nsmall = 3), '$') }
  return(pchar)
}

```

Results

As hypothesized, X was correlated with Y, $r=$ `r format(round(cor_XY$estimate, 2),
nsmall = 2)`, $df=$ `r cor_XY$parameter`, `r Pval(cor_XY$p.value)`.
```

Requirements

one needs to have R, RStudio & a LaTeX bundle installed, if not (or if there is an error) just type

```
tinytex::install_tinytex()
```

RStudio then

- takes the R Markdown file
- performs all statistical analyses
- places their results where you want them to appear in the format you want to have
- translates everything into a LaTeX file and compiles it to a PDF (or HTML/DOCX)

one needs to have some conception on how the results of statistical analyses should be reported and where in some R object they are stored

R Markdown
Overview

R Markdown

combines the benefits of R and Markdown (and LaTeX), i.e., one writes text in markdown and includes R code in so-called R code chunks, e.g.

```
# Method

We report how we determined our sample size, all data exclusions (if any), all
manipulations, and all measures in the study.

```{r correlation, echo = F}
set.seed(242)
X = rnorm(64)
Y = .5 * X + rnorm(64)
cor_XY = cor.test(X, Y)
Pval <- function(p) {
  if (p < .001) { pchar = '$p<.001$' }
  else { pchar = paste0('$p=', format(round(cor_XY$p.value, 3), nsmall = 3), '$') }
  return(pchar)
}

```

Results

As hypothesized, X was correlated with Y, $r=$ `r format(round(cor_XY$estimate, 2),
nsmall = 2)`, $df=$ `r cor_XY$parameter`, `r Pval(cor_XY$p.value)`.
```

Compiled PDF

looks like this (even if you do not know the exact results in advance, because you defined placeholders in the Results section):

Method

We report how we determined our sample size, all data exclusions (if any), all manipulations, and all measures in the study.

Results

As hypothesized, X was correlated with Y, $r = 0.40$, $df = 62$, $p < .001$.

R Markdown
Dynamic documents

R Markdown

allows for dynamic documents, i.e., if code changes, the document is updated and gives the correct results (no need to revise text by hand)

```
# Method

We report how we determined our sample size, all data exclusions (if any), all
manipulations, and all measures in the study.

```{r correlation, echo = F}
set.seed(242)
X = rnorm(64)
Y = .5 * X + rnorm(64)
cor_XY = cor.test(X[-1], Y[-1])
Pval <- function(p) {
  if (p < .001) { pchar = '$p<.001$' }
  else { pchar = paste0('$p=', format(round(cor_XY$p.value, 3), nsmall = 3), '$') }
  return(pchar)
}

```

Results

As hypothesized, X was correlated with Y, $r=$ `r format(round(cor_XY$estimate, 2),
nsmall = 2)`, $df=$ `r cor_XY$parameter`, `r Pval(cor_XY$p.value)`.
```

Compiled PDF

looks like this (after you removed observation 1 that was you for testing purposes) because you defined placeholders in the Results section:

Method

We report how we determined our sample size, all data exclusions (if any), all manipulations, and all measures in the study.

Results

As hypothesized, X was correlated with Y, $r = 0.39$, $df = 61$, $p = 0.001$.

first step towards computational reproducibility because R code and text are tightly intertwined

Great talk of Russ Poldrack on computational reproducibility
https://www.youtube.com/watch?v=XjW3t-qXAiE

R Markdown
Graphics integration

R Markdown

allows for integration of (updateable) graphics, so there is no need to create a new figure after something has changed

```
# Method

We report how we determined our sample size, all data exclusions (if any), all
manipulations, and all measures in the study.

```{r correlation, echo = F}
set.seed(242)
X = rnorm(64)
Y = .5 * X + rnorm(64)
cor_XY = cor.test(X[-1], Y[-1])
Pval <- function(p) {
  if (p < .001) { pchar = '$p<.001$' }
  else { pchar = paste0('$p=', format(round(cor_XY$p.value, 3), nsmall = 3), '$') }
  return(pchar)
}

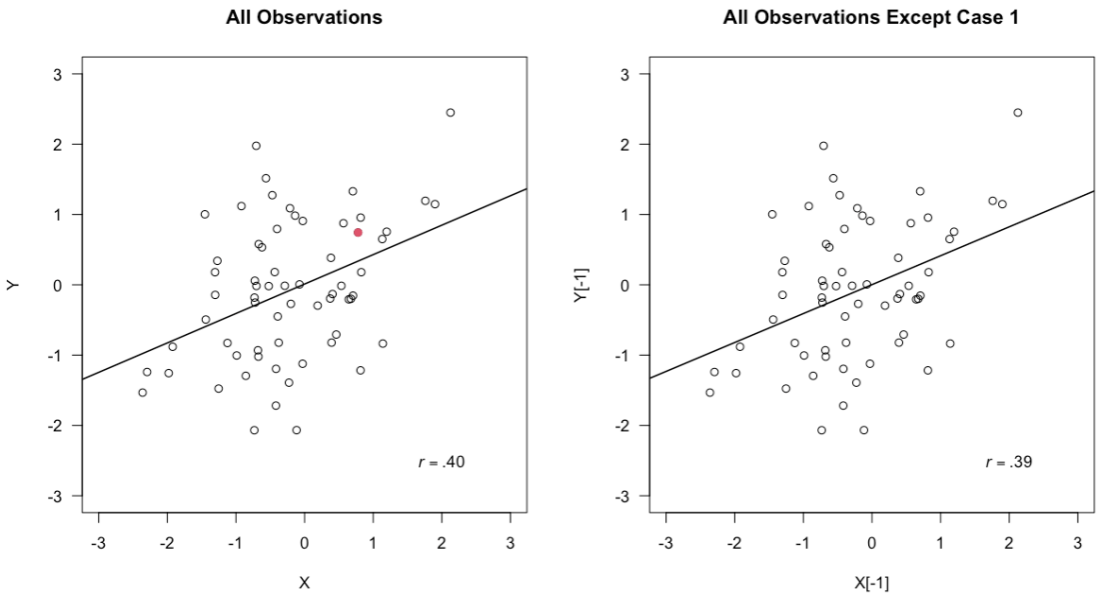
```

Results

As hypothesized, X was correlated with Y, $r=$ `r format(round(cor_XY$estimate, 2),
nsmall = 2)`, $df=$ `r cor_XY$parameter`, `r Pval(cor_XY$p.value)`.
```

Compiled PDF

looks like this (after you removed observation 1 that was you for testing purposes) as figures are create everytime anew if there were changes



R Markdown
LaTeX integration

R Markdown

requires some basic knowledge about LaTeX, as a number of formatting resorts to (or can be easier implemented with) LaTeX

```
# Method

We report how we determined our sample size, all data exclusions (if any), all
manipulations, and all measures in the study.

```{r correlation, echo = F}
set.seed(242)
X = rnorm(64)
Y = .5 * X + rnorm(64)
cor_XY = cor.test(X[-1], Y[-1])
Pval <- function(p) {
  if (p < .001) { pchar = '$p<.001$' }
  else { pchar = paste0('$p=', format(round(cor_XY$p.value, 3), nsmall = 3), '$') }
  return(pchar)
}

```

Results

As hypothesized, X was correlated with Y, $r=$ `r format(round(cor_XY$estimate, 2),
nsmall = 2)`, $df=$ `r cor_XY$parameter`, `r Pval(cor_XY$p.value)`.
```

Compiled PDF

nicely formats text marked as LaTeX via $...$ in the preferred way, i.e., statistical coefficients are printed in italics:

Method

We report how we determined our sample size, all data exclusions (if any), all manipulations, and all measures in the study.

Results

As hypothesized, X was correlated with Y, $r = 0.39$, $df = 61$, $p = 0.001$.

also allows for more complex formulae such as

```
$$z_{w} = \frac{\sum_{i=1}^{k}w_{i}z_{i}}{\sqrt{\sum_{i=1}^{k}w_{i}^2}}$$
```

$$z_w = \frac{\sum_{i=1}^{k} w_i z_i}{\sqrt{\sum_{i=1}^{k} w_i^2}}$$

R Markdown
BibTeX integration

R Markdown

requires some basic knowledge about BibTeX, as citation works via BibTeX (a standard all journals offer and references managers can handle)

```
# Method

We report how we determined our sample size, all data exclusions (if any), all
manipulations, and all measures in the study [cf. @Simmons2012].

```{r correlation, echo = F}
set.seed(242)
X = rnorm(64)
Y = .5 * X + rnorm(64)
cor_XY = cor.test(X[-1], Y[-1])
Pval <- function(p) {
  if (p < .001) { pchar = '$p<.001$' }
  else { pchar = paste0('$p=', format(round(cor_XY$p.value, 3), nsmall = 3), '$') }
  return(pchar)
}

```

Results

As hypothesized, X was correlated with Y, $r=$ `r format(round(cor_XY$estimate, 2),
nsmall = 2)`, $df=$ `r cor_XY$parameter`, `r Pval(cor_XY$p.value)`.
```

Compiled PDF

pastes reference according to some predefined *.csl (= citation language stylesheet) in the way you or the journal wants to have them

Method

We report how we determined our sample size, all data exclusions (if any), all manipulations, and all measures in the study (cf. Simmons et al., 2021).

the respective BibTeX entry and ref. look like this:

```
@article{Simmons2012,
  author = {J. P. Simmons and L. D. Nelson and U. Simonsohn},
  year  = {2012},
  title  = {A 21 word solution},
  url   = {http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2160588},
  doi   = {10.2139/ssrn.2160588},
}
```

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2012). *A 21 word solution.*

https://doi.org/10.2139/ssrn.2160588

R Markdown
One problem

One problem ...

with R Markdown is that one cannot as easily revise or comment on documents as in Word

```
# Method

We report how we determined our sample size, all data exclusions (if any), all
manipulations, and all measures in the study.
```

Here, you want to add a reference to Simmons, Nelson, & Simonsohn (2012)

```
```{r correlation, echo = F}
set.seed(242)
X = rnorm(64)
Y = .5 * X + rnorm(64)
cor_XY = cor.test(X[-1], Y[-1])
Pval <- function(p) {
  if (p < .001) { pchar = '$p<.001$' }
  else { pchar = paste0('$p=', format(round(cor_XY$p.value, 3), nsmall = 3), '$') }
  return(pchar)
}

```

Results

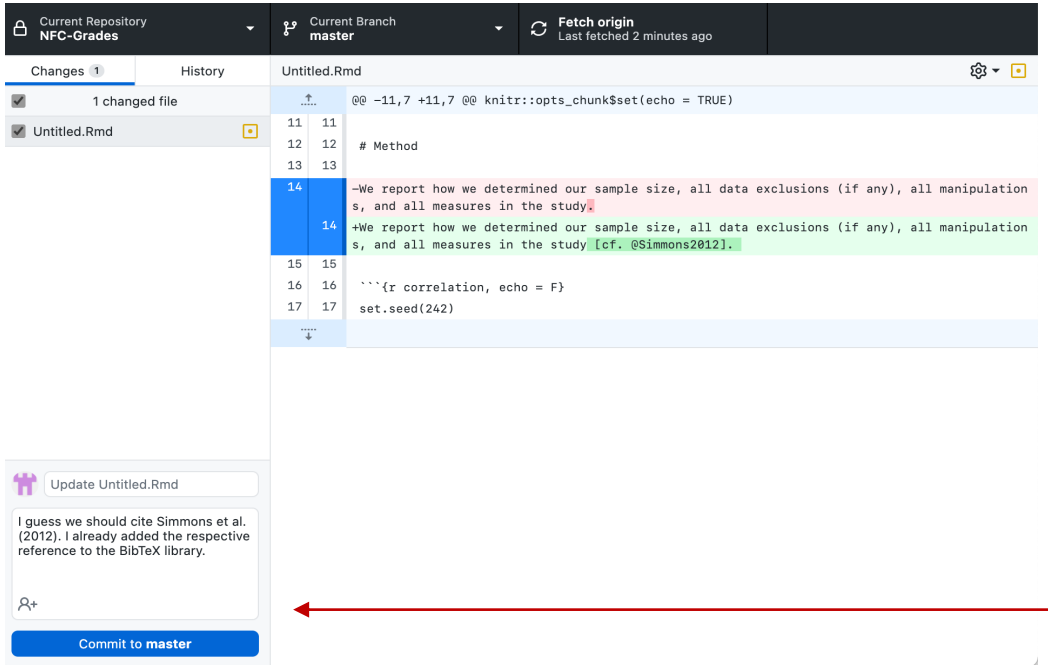
As hypothesized, X was correlated with Y, $r=$ `r format(round(cor_XY$estimate, 2),
nsmall = 2)`, $df=$ `r cor_XY$parameter`, `r Pval(cor_XY$p.value)`.
```

Here, you want state that according to APA, correlations are printed without leading zeroes

R Markdown
One solution

One solution ...

is GitHub, you have version control and can revise documents (and comment on revisions) without permanently changing the document



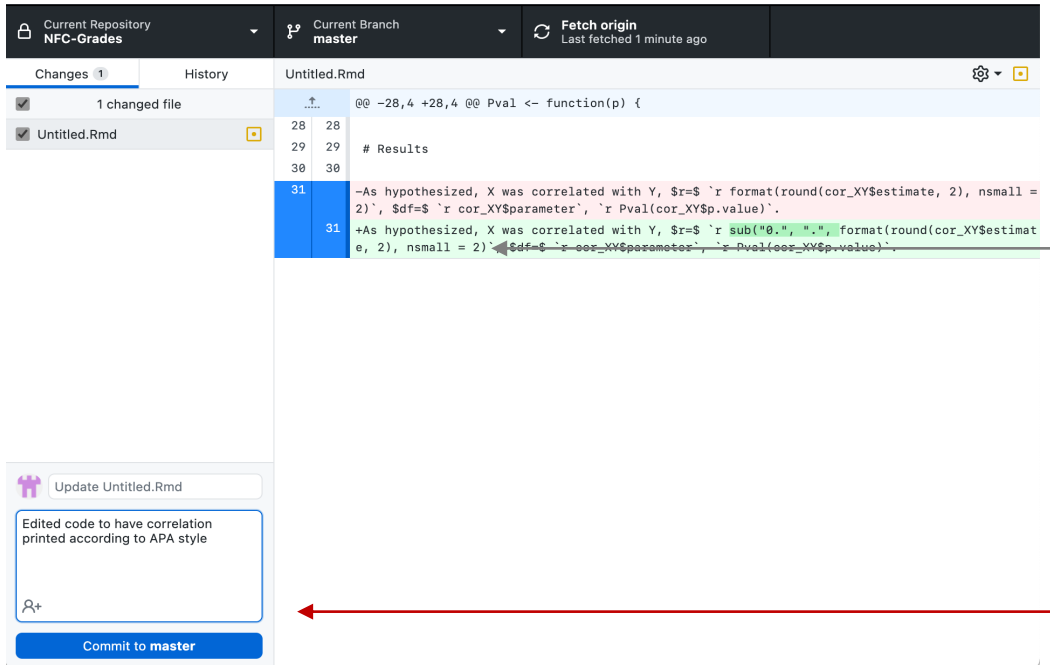
Here, you add a reference to Simmons, Nelson, & Simonsohn (2012), and the change is highlighted in GitHub Desktop

Here, you comment on your revision.

R Markdown
One solution

One solution ...

is GitHub, you have version control and can revise documents (and comment on revisions) without permanently changing the document



Here, you change some R code.

And here, you made a coding error, i.e., a missing closing parenthesis, but nevermind: this will stop the document compilation, you will get an error message (although not an informative one most of the time), but you can be assured that the code must work before a document will be compiled.

Here, you comment on your revision.

https://github.com

GitHub

GitHub

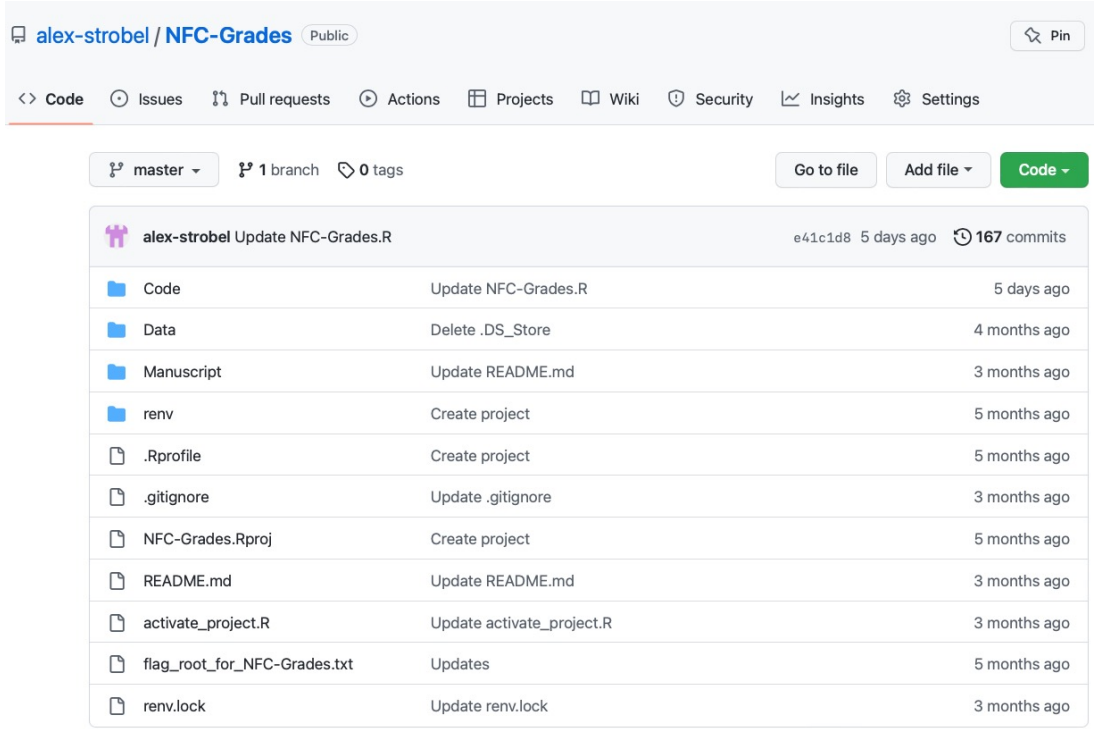
in case of R Markdown files, a group drive/sharepoint is of limited value, because if something changes, it may not be easy to keep track of these changes

also, if some change eventually results in a dead end, you may want to go back to a previous point in the "timeline" of your project

here, version control is really helpful, and GitHub enables it (in fact, it has been *made* for it, among others)

the above examples showcase how revisions and comments can be implemented

comes with great features apart from code organization such as *Issues* and *Projects* or a *Wiki*



GitHub
Basics

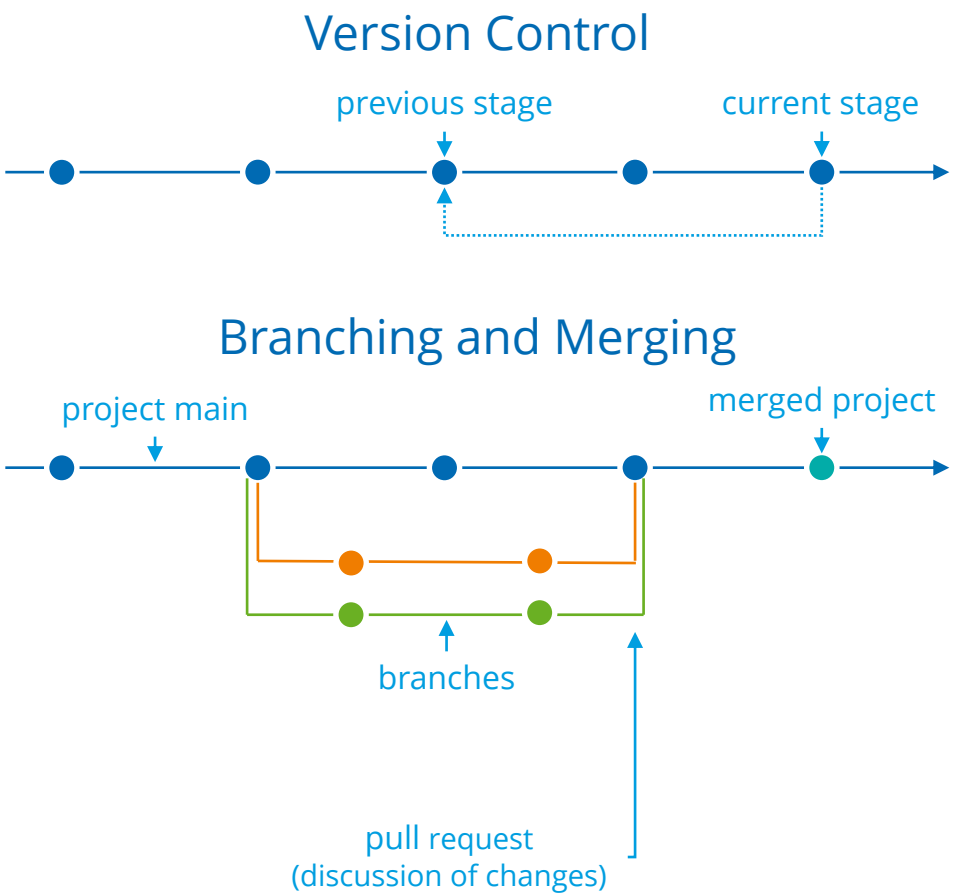
GitHub Basics

GitHub is great for working together on code (in the broadest sense)

comes with version control, so that if something went wrong, you go back in the project timeline, so there is no need for having multiple files to monitor project progress such as *Project_final.docx, Project_final2.docx* etc. in order to be able to return to earlier project stages

also allows for parallel working on projects on different project branches that are merged in the end, which is especially helpful when working with several co-authors, so no need to integrate multiple word documents

Collaborative writing can of course be done via Google Docs/Sharepoint, yet, there is no (good) version control and working on R/R Markdown code will be not feasible

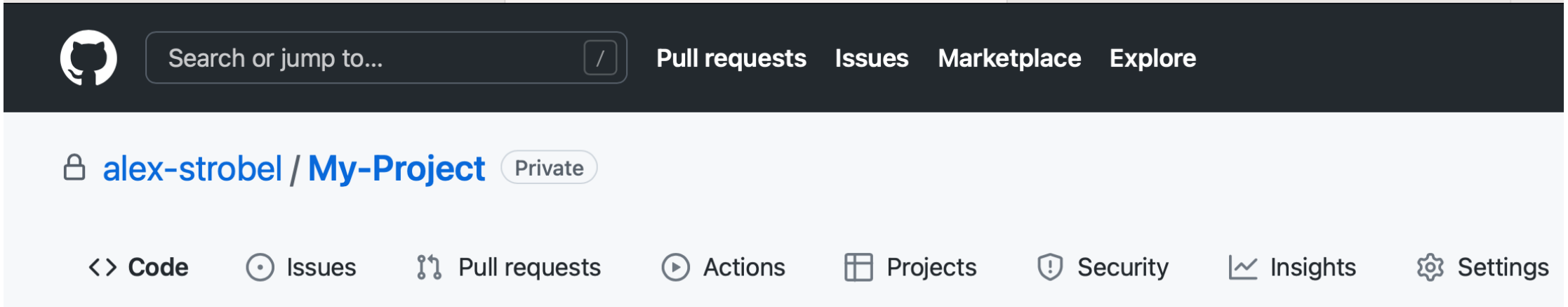


GitHub
Workflow

GitHub Workflow

Researcher A creates GitHub repository, invites collaborators B and C via “Settings“

A, B, and C either work together
— on the main project „Code“ or
— work on separate taks (as defined, e.g., via „Projects“ or „Issues“) or
— create a branch from the main timeline and later issue a „Pull request“
— organize their collaboration via Issues, Projects and perhaps a Wiki (here not visible)



GitHub
Drawbacks

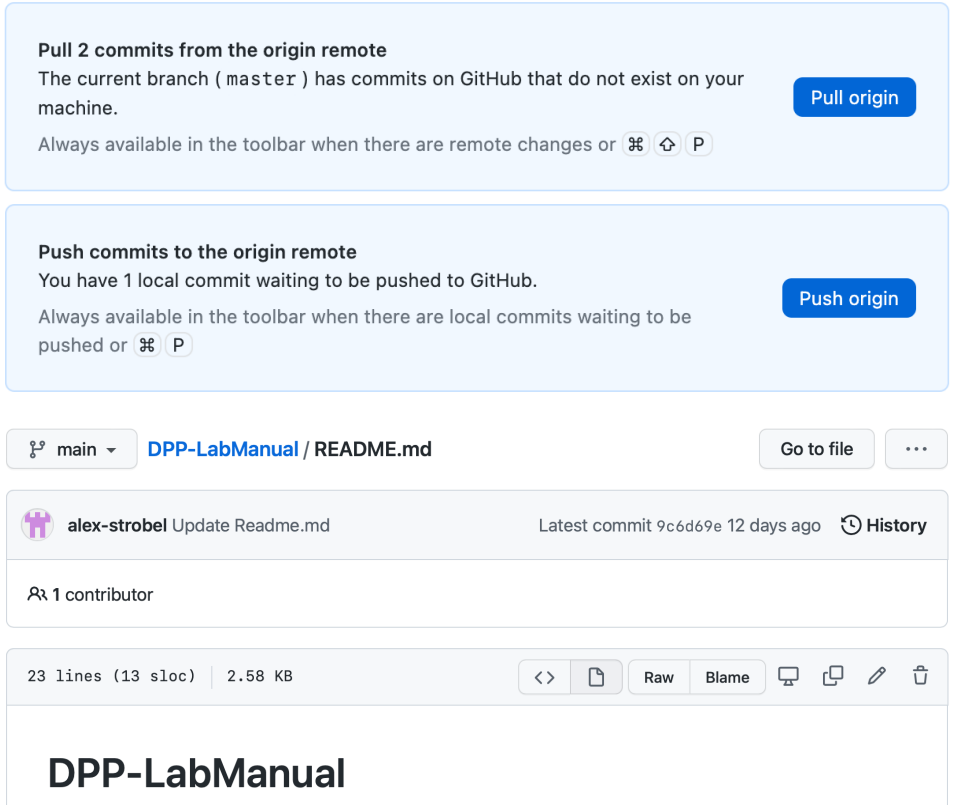
GitHub Drawbacks

GitHub has been made by and for people who one could call nerds; so if you aren't a bit of a nerd, you might get lost, because of GitHub's structure and terminology

GitHub is not immediately compatible with spoken language

— retrieving the current project status is called a **pull**
— changing something is called a **commit**
— uploading your changes is called a **push**
— having created a branch and wanting to have it merged with the main project is called a **pull-request**

synchronizing with GitHub will work differently depending on your OS and code editor; best results IMHO are achieved using GitHub Desktop and/or VisualStudio Code (see below)



GitHub
Benefits

GitHub Benefits

collaboration is as easy as possible

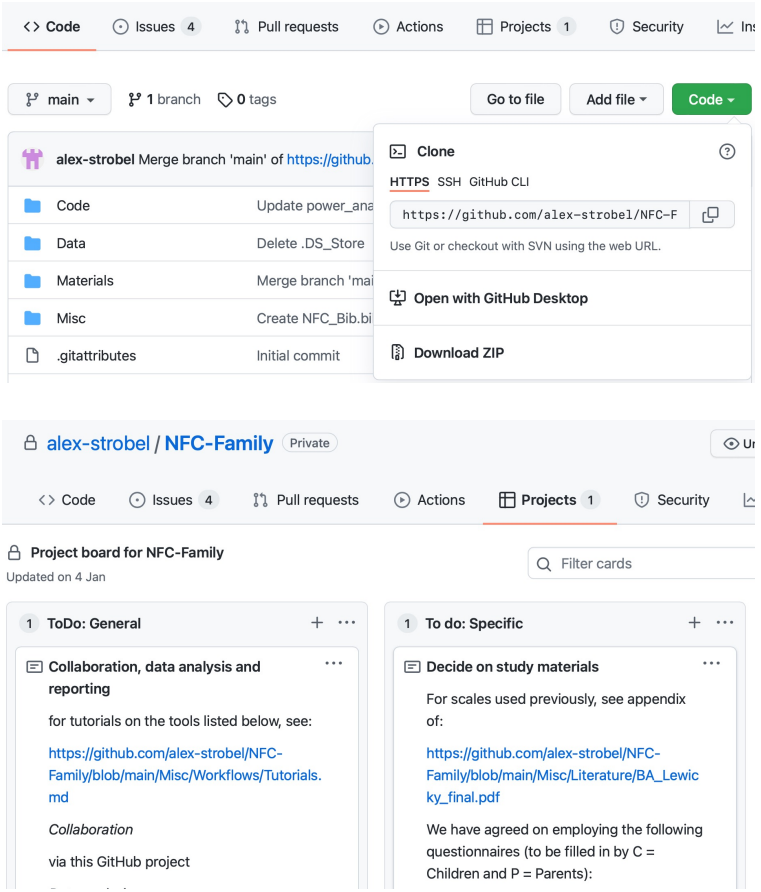
project data and code are in one place and are always up to date

is also great as backup even when working alone on a project

project progress is marked with a time stamp, so all steps are transparent all the time, no one can engage in some HARKing as it will be evident if some hypothesis is added to the manuscript file at a later time point than the analysis code has been updated (can easily be faked of course, but good will presumed ...)

does not only support collaboration, but also sharing of data and code, because a whole project can be downloaded (aka „cloned“) in an instant instead of downloading OSF files one at a time

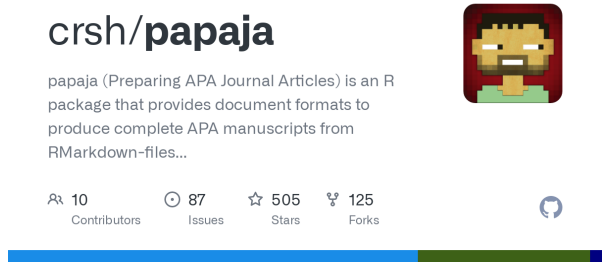
also has tools for project administration (Issues, Projects) and a built-in Wiki function



Recommended tools
R packages

papaja

lets you format your manuscript in APA6 format and includes a variety of helper functions, e.g., for reporting statistics in a proper (and properly formatted) way



renv

stands for **r**eproducible **env**ironment and stores your R analysis environment for a given project, so that people who want to reproduce your results not only have your code but the very R package versions you used

* using renv may sometimes be troublesome, so were are about to go for an alternative tool: conda

here

assists with regard to file paths; with *here*, there is no need to edit path names of downloaded or cloned projects

https://github.com/crsh/papaja | https://rstudio.github.io/renv/articles/renv.html | https://here.r-lib.org

Recommended tools
Software

GitHub Desktop

keeps track of all changes to your GitHub project, be them local or global

will prompt you to pull from the main project time line whenever something has changed and will support you to push changes from your timeline

VisualStudio Code

this program is *the* code editor to go for, even if you do not want to work with GitHub and R or R Markdown, but like/want/need to work on code

comes with code highlighting depending on the programming language you use, has a preview, has dark mode (some people seem to like this 😉)

https://desktop.github.com | https://code.visualstudio.com

Summary and outlook
Summary

Summary

Actually working on R Markdown files with GitHub integration at first might seem like „floxjam.com“:

Yet, after you have familiarized yourself with the terminology and mastered a first small project, you will not want to use other collaboration platforms anymore

Using R Markdown w/ GitHub will also enhance transparency and sharing of data and code

Using helper packages like *papaja*, *renv* and *here* brings added value

Using VisualStudio Code is in itself a great relief when it comes to coding, and together with GitHub, you can use its full potential (and if you have Spotify Premium, you can even listen to your music from within VS Code!)

But don‘t just take my word for it, listen to this woman! 👇

„GitHub makes collaboration so much easier! Everything is in one place and always up to date, no need to keep track of changes by yourself which is especially helpful when using R Markdown. And everything is so much more transparent …“

Summary and outlook
Outlook

Drawbacks

actually working on R Markdown files with GitHub will most certainly result in a number of issues

you will *initially* need to resort to astonishingly incomprehensible help pages (yet, mostly bc of R Markdown)

yet, you will also *later on* need to resort to astonishingly incomprehensible help pages, so why not start now?!

Benefits

After the first manuscript, you will master R Markdown to some degree and more so after the second, and after the third, you will be annoyed that your co-authors do not know R Markdown, so you have to compile everything to a stupid Word document for them to review/revise it

https://www.exemplifygroup.com/look-benefits-drawbacks-unified-communications/

Further resources

GitHub

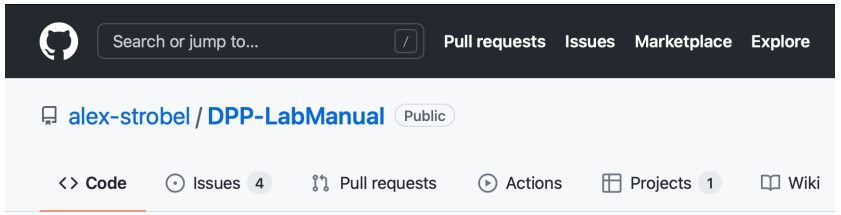
— GitHub's Git Guides
— Happy Git With R

R Markdown

— R Markdown Guide
— R Markdown Cheat Sheet
— R Markdown Course (developed by Christoph and me, workshop will be offered as soon as possible)

Using GitHub and renv/here to collaborate on R Markdown projects

— our tutorial on collaboration via GitHub (note, we are just about re-organizing our LabManual, so you may need to click through from the root)



Thank you!

Credentials

The creation of this workshop material was partially funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG; SFB 940/3).