Reproducible Research in R

How to do the same thing more than once

Andreas Brandmaier

University of Leipzig | 2024-09-23

Please ask questions anytime

Reproducible Research

=

same data + same analysis

=

same results

Why should we work reproducibly?

Why should we work reproducibly?

Many good reasons like

- Transparency
- Trustworthiness
- Replication

> Transparency and accessibility are central scientific values, and open, reproducible projects will increase the efficiency and veracity of knowledge accumulation (Bar-Anan and Nosek, 2012).

Your closest collaborator is you six months ago, but you don’t reply to emails.

From Karl Broman's lecture on reproducibility, paraphrasing Mark Holden

What should be reproducible?

Everything

(Almost) Everything

Statistical Data Analysis

- Replication attempt of articles in journal *Cognition* (Hardwicke, Mathur, MacDonald, Nilsonne, Banks, Kidwell, Hofelich Mohr, Clayton, Yoon, Henry Tessler, and others, 2018)
- Out of 35 published articles with open code and data, the results of 22 articles could be reproduced,
- Further assistance from the original authors was required in 11 cases
- For 13 articles, at least one outcome could not be reproduced—even with the original authors' assistance.

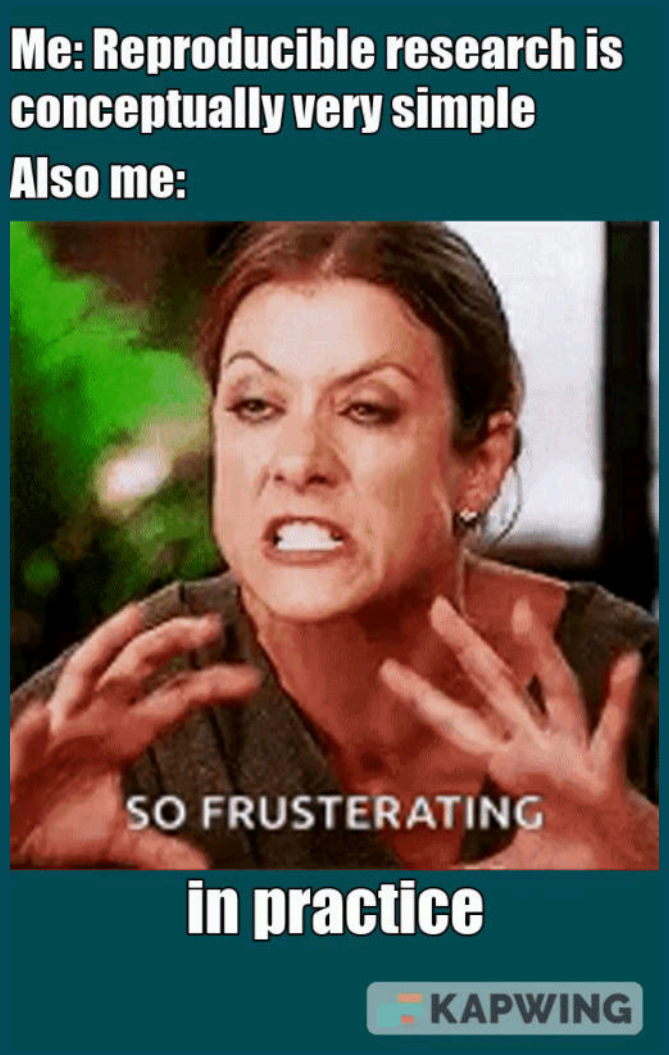
Statistical Data Analysis

- in 62 registered reports only 41 had data available, and 37 had analysis scripts available (Obels, Lakens, Coles, Gottfried, and Green, 2020).
- The authors could execute only 31 of the scripts without error
- They reproduced the results of only 21 articles (within a reasonable time).
- Lesson: Just because it's open data/code, it's not necessarily useful

Power Analysis

Reproducibility of power analyses (Thibault, Zavalis, Malički, and Pedder, 2024):

- 2% reproduced
- (likely) 28% reproduced with additional assumptions
- 75% not reported transparently enough to assess reproducibility (e.g., unclear whether power analysis matches later statistical analysis)



Meme by Aaron Peikert

Meme by Aaron Peikert

Rule 1: Make it portable & keep your stuff together!

Always avoid absolute file paths. (e.g., `setwd()`)

Probably (*not*) a good idea to add this to your user-wide `.Rprofile` (`usethis::edit_r_profile()`):

```
setwd <- function(...) { stop("Bad boy!")}
```

```
setwd("C:/Users/brandmaier/SOEP")
```

> Error in `setwd()`: Bad boy!

Use projects and `here`

Use RStudio and *R projects* instead

The command `here` retrieves files relative to project root

```
here("data/soep.csv")
```

```
## [1] "C:/Users/andreas.brandmaier/LokaleDokumente/GitHub/reproducibility_workshop/data/soep.csv"
```

Project should be self-contained

Sources of Failure to Reproduce Results

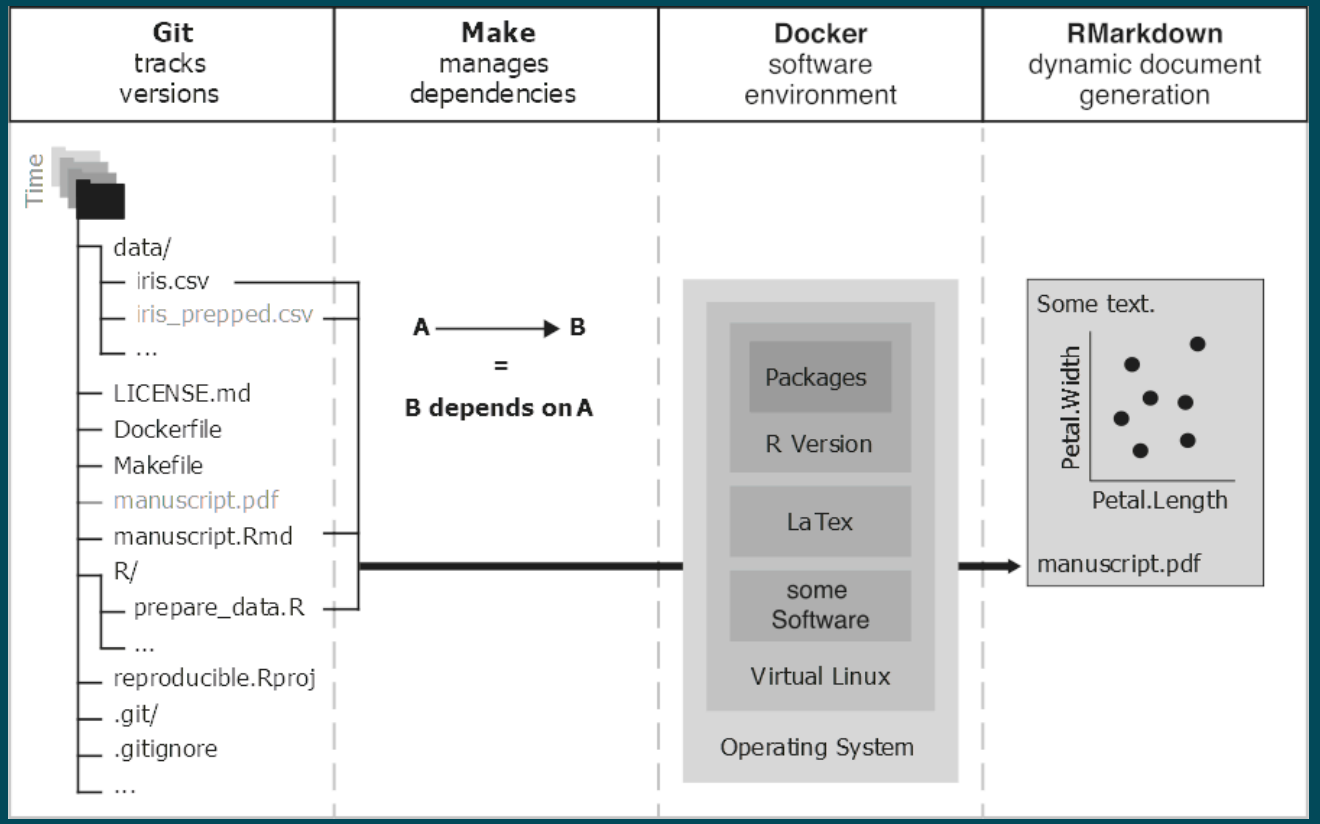
1. **Multiple versions of scripts/data** (e.g., dataset has changed over, i.e., was further cleaned or extended)

2. **Multiple scripts** in a pipeline; unclear which scripts should be executed in which order

3. **Copy&paste errors** (e.g., inconsistency between reported result and reproduced result)

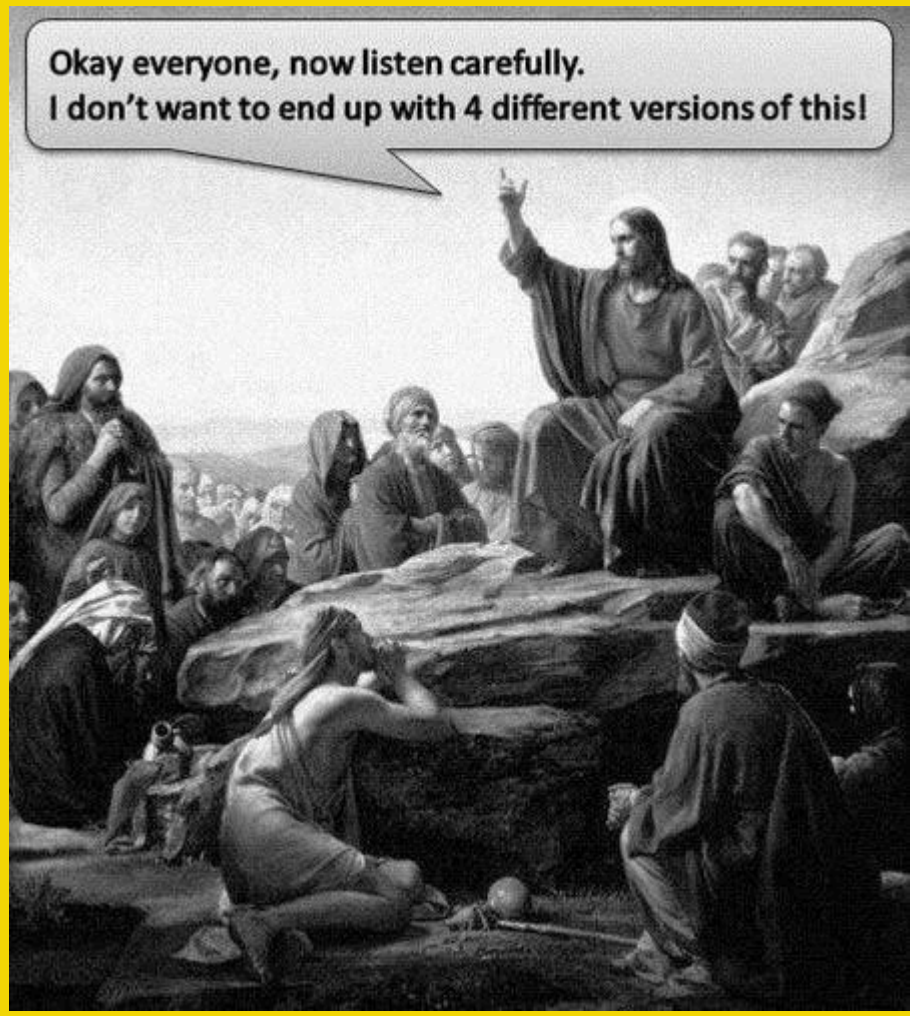
4. Broken **software dependencies** (e.g., analysis broken after update, missing package, just comes out differently on a different computer)

Four Elements of Reproducibility



from Peikert and Brandmaier (2020)

Version Control



Problem: Multiple versions of scripts/data

Version control

manuscript_revised.docx

manuscript_revised_final.docx

manuscript_revised_final_commentsAB.docx

manuscript_revised_final_commentsAB_final.docx

Version control in a nutshell

Version 1 → Version 2 → Version 3 → Current

+

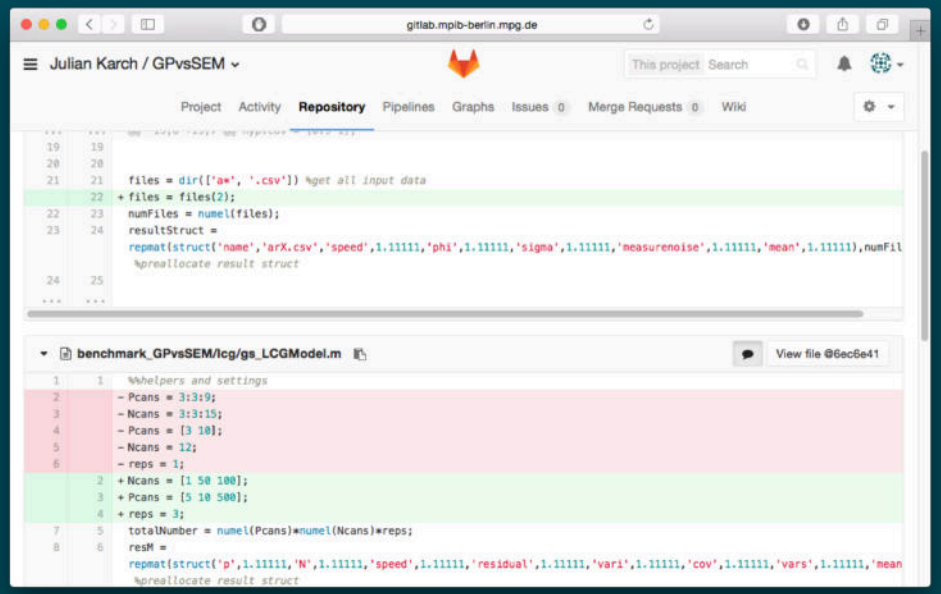
Time travel

+

Change detection

Version Control

- Version control systems (VCS) such as `git` record changes to a set of files over time; restore specific versions later.
- VCS guarantee that code and data are exactly the same version as used for publication.
- Allows collaboration and track-changes



Collaborative editing

- Multiple authors can **commit** changes to their local repositories and **push** (synchronize) with a central repository (automated merge of non-conflicting changes)
- Authors can work on separate branches, which can be merged later (e.g., **pull request** with mandatory **review**)
- Authors can **fork** (clone) the entire repository
- Added value in some systems (issues, project management, ...)

v0.2.0-submission

d5b3d38

Verified

Compare

Resubmission

Edit

aaronpeikert released this 19 days ago · 3 commits to master since this release

Releases on github.com

This is the release to the third version of the preprint and resubmission.

▾ Assets 4

manuscript.pdf	562 KB
reproducible-research.tar.gz	1.06 GB
Source code (zip)	
Source code (tar.gz)	

v0.1.1-submission

d840bf1

Verified

Compare

Submission

Edit

aaronpeikert released this on 12 Nov 2019 · 134 commits to master since this release

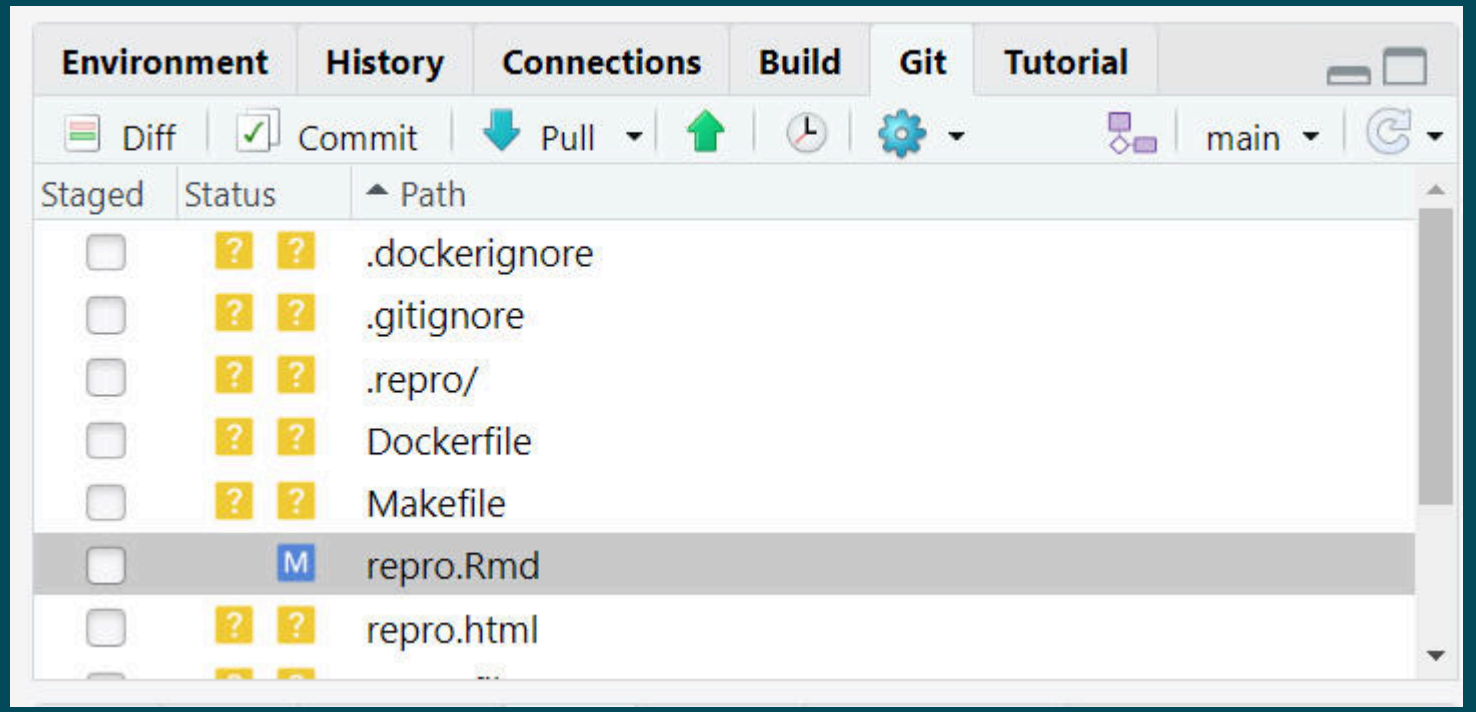
This is the release to the second version of the preprint and submission.

▸ Assets 4

Long-Term Storage

- GitHub has no long-term guarantees for the availability of its service
- even though they do LOCKSS (for Lots Of Copies Keeps Stuff Safe, e.g., the Artic World Archive)
- Mirror snapshots with meta-data and DOI to other providers (e.g., Zenodo, FigShare, **OSF via plugin**)
- This helps making the repository (F)indable (as in FAIR)

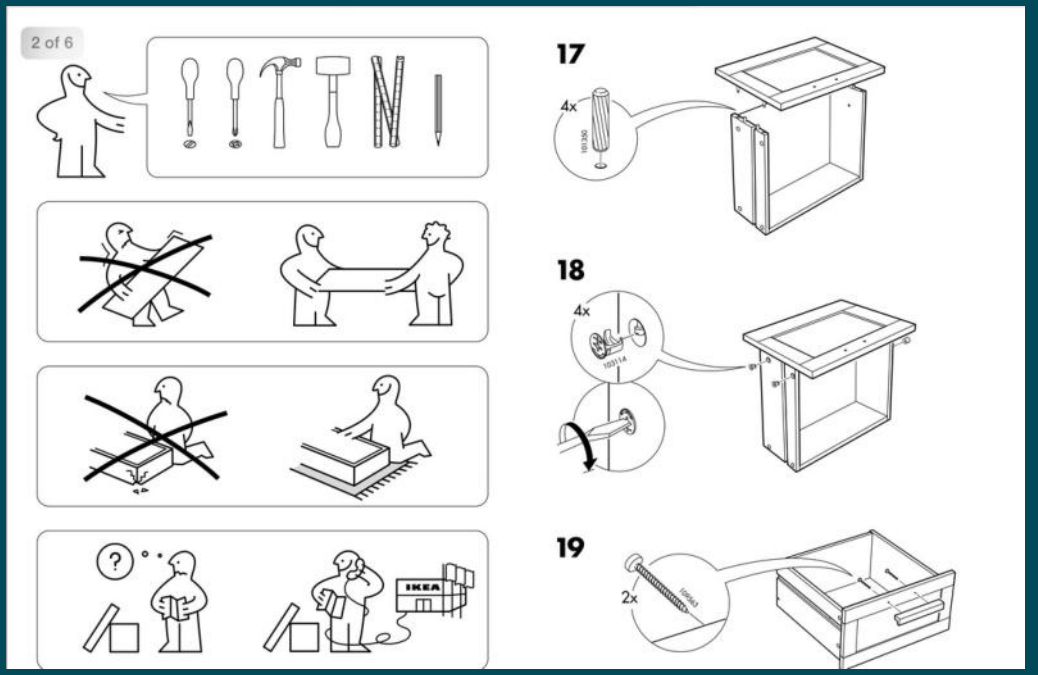
In RStudio


Environment
History
Connections
Build
Git
Tutorial
Diff
Commit
Pull
main
Staged
Status
Path
.dockerignore
.gitignore
.repro/
Dockerfile
Makefile
repro.Rmd
repro.html

Dependency Management

Problem

Once someone found our files... which of those files are executable and in which order are they to be executed?



What's cooking?

> Arrabbiata sauce, or sugo all'arrabbiata in Italian, is a spicy **sauce** for **pasta** made from **garlic**, **tomatoes**, and dried red **chili** peppers cooked in olive oil.

"Arrabbiata sauce" from Wikipedia under CC BY-SA 3.0, emphasis added

> 1. Bring 6 quarts of **water** to a boil in a large pot, and add 3 tablespoons **kosher salt**.
> 2. Meanwhile, put 4 tablespoons **olive oil** in a large sauté pan over medium heat, [...]

Source: cooking.nytimes.com

Solution: GNU Make

A Makefile

- contains a number of recipes
- Each recipe contains its ingredients (=dependencies) and commands to create the product
- There is a default recipe (defined entry point)
- By convention, there is an „all“ recipe to create everything
- Recipes can depend on other recipes and/or files

Example Makefile

Makefile Schema

```
recipe name: ingredients
  instructions
```

An example Makefile

```
all: manuscript.pdf

manuscript.pdf: data/iris_prepped.csv manuscript.Rmd
  Rscript -e 'rmarkdown::render("manuscript.Rmd")'

data/iris_prepped.csv: R/preprocess_data.R data/iris.csv
  Rscript -e 'source("R/preprocess_data.R")'
```

Key features

Missing ingredients will be generated,

e.g., if the cleaned data is missing, the raw data is first cleaned.

Newer ingredients trigger updates,

e.g., new data leads to recreation of the whole manuscript.

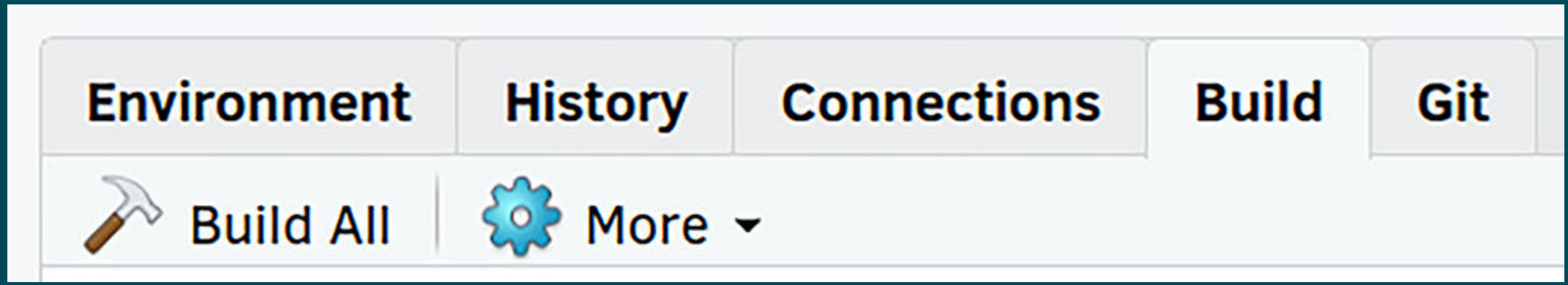
Always the same "button" that triggers reproduction,

e.g., regardless of programming language, file format, and intermediate steps.

Why Make

- A single, well-defined entry point for your analysis (default target): `make`
- Management of all external dependencies in one file via (dependend) targets, such as
  - Installing extra software
  - Starting external programs (such as pre-processing pipelines)
  - Downloading data from external repositories

Make in RStudio



Trigger default make using **build all**

Native R alternative?

Alternative: build automation using the `targets` package Landau (2021)

- using `tar_target(y, f(x))` instead of `y <- f(x)`
- then `tar_make()` builds/updates all targets
- like in Make, things are built only if they changed
- built-in visualization of dependency graph

Containerization

Problem: Broken *software dependencies* ('dependency hell')

Docker

A Docker container is like a shareable virtual machine that runs identically on any computer (Linux, macOS, Windows)

- You can either provide a short recipe how to create a container (few kilobytes) or
- the entire container (few gigabytes)

Docker Recipes

Recipes are instructions how to create a container from publicly available sources, e.g.,

- Rocker project (Boettiger & Eddelbuettel, 2017), with pre-configured Debian images including R/Rstudio
- Microsoft R Application Network (MRAN) providing CRAN snapshots in their „time machine“ (out of service now!)

```
FROM rocker/verse:3.6.1
ARG BUILD_DATE=2019-11-11
RUN install2.r --error --skipinstalled here lavaan
WORKDIR /home/rstudio
```

Docker Recipes

Recipes are instructions how to create a container from publicly available sources, e.g.,

- Rocker project (Boettiger & Eddelbuettel, 2017), with pre-configured Debian images including R/Rstudio
- Microsoft R Application Network (MRAN) providing CRAN snapshots in their „time machine“ (out of service now!)

```
FROM rocker/verse:3.6.1
ARG BUILD_DATE=2019-11-11
RUN install2.r --error --skipinstalled here lavaan
WORKDIR /home/rstudio
```

Run in Docker

Example of how to run R in a docker image from the terminal interactively (--it) and without saving its state (--rm)

```
# start an interactive R session in the
# container named 'reproducible-research'
docker run --rm -it reproducible-research R
```

more info in (Peikert and Brandmaier, 2021)

The *maybe good enough* way: renv

- The collection of installed packages is a *library* in R
- The R package `renv` (Ushey & Wickham, 2024) separates system library from *project-specific local libraries* in R
- Key concept: *isolation* of your project's dependencies from your system and other projects (different projects can use different package versions)

Components

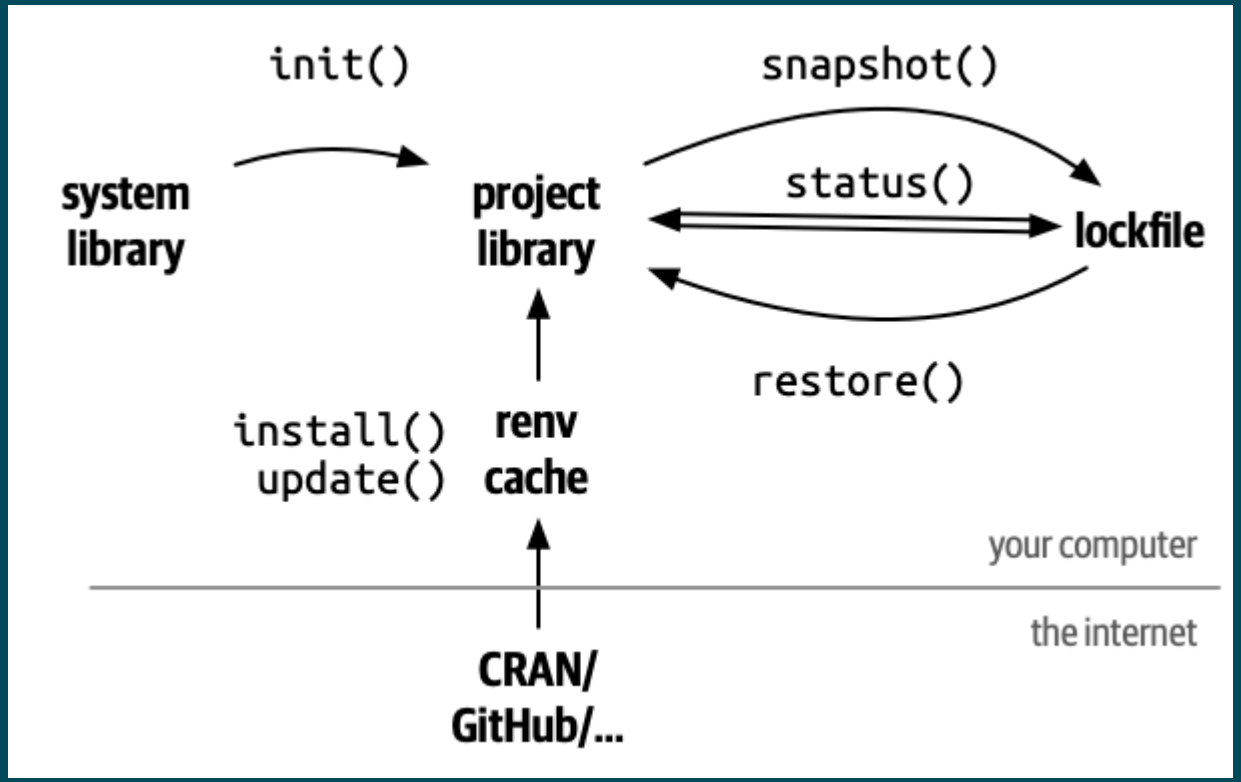
- The project library, renv/library, is a library that contains all packages currently used by your project
- the renv.lock file, which contains meta data about the local library (**share** this with collaborators)
- A project R profile `.Rprofile`, which is loaded once a project is loaded (remember `setwd()`)

Commands

- Create a private R library with `renv::init()` (the project will now always rely on the local library)
- Update a library with `renv::snapshot()` (e.g. when updating packages to a newer version)
- Restore a library with `renv::restore()`

cf. the WORCS approach by (Van Lissa, Brandmaier, Brinkman, Lamprecht, Peikert, Struiksma, and Vreede, 2021)

Overview



Source: https://rstudio.github.io/renv/articles/renv.html

But is this really enough?

The case of pseudo random numbers

- Pseudo random numbers are important in modern statistics, e.g. permutation tests, bootstrapping, random forests but also for reproducible allocation of participants to experimental conditions
- PRN generators can be thought of as long lists of numbers that repeat only after a very long time (efficiently represented as a mathematical function); the list is determined by a starting condition (**seed**)
- For reproducibility, we need to fix that initial **seed** (ideally by another random process)

Examples of Non-Reproducible Code in R

Here are some examples of non-reproducible code that cannot be captured easily from within a given R environment

1. Bugfix in random number generator in R between R 3.5 and R 3.6

```
set.seed(1234);
sample(1:10, 5)
```

```
2 6 5 8 9 (R3.5)
```

```
10 6 5 4 1 (R3.6.1)
```

Examples of Non-Reproducible Code in R

Confidence intervals (95%) of a simple regression coefficient estimate (with identical random seed):

```
[1] "R version 3.5.0 (2018-04-23)"

2.5 % 97.5 %
 0.0097 0.3842
```

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

2.5 % 97.5 %
-0.0005 0.3748
```

Note that the results are not reproduced but replicated.

Examples of Non-Reproducible Code in R

2. Locale-dependent behavior (e.g., English vs Lithuania):

```
sort(state.abb)

 [1] "AK" "AL" "AR" "AZ" "CA" "CO" "CT" "DE" "FL" "GA" "HI" "IA"
[13] "ID" "IL" "IN" "KY" "KS" "LA" "MA" "MD" "ME" "MI" "MN" "MO"
[25] "MS" "MT" "NC" "ND" "NE" "NH" "NY" "NJ" "NM" "NV" "OH" "OK"
[37] "OR" "PA" "RI" "SC" "SD" "TN" "TX" "UT" "VA" "VT" "WA" "WI"
[49] "WY" "WV"
```

3. Defaults change over time

- Loading data has different defaults from R 4.0+
- `read.csv()` now has stringsAsFactors = FALSE
- Earlier strings were converted to factors automatically (with levels sorted by local language) => non-reproducibility

```
c("0", "1", "A", "B", "a", "b")
```

```
## [1] "0" "1" "A" "B" "a" "b"
```

sorted to `01aAbB` in German, `01AaBb` in Norwegian, `01ABab` in `Korean`

There is a need for reproducible computational environments

Containers

Docker containers

- guarantee identical execution of its contents across platforms (and time) including cloud computing (such as github actions)
- can run interactively: R/Rstudio can run in the container (i.e., there is no need to ever run the analysis in your local computing environment, see Peikert and Brandmaier (2021))
- Analyses can be run in different containers (simulating different package versions) to find out what downstream updates broke your code / your results

Dynamic Document Generation

Problem: Oh these darn copy&paste errors!

R Markdown Example

my-report.Rmd

```
---
title: "My report"
author: "Anna Krystalli"
date: "14/05/2019"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

My report

My report

Anna Krystalli

14/05/2019

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

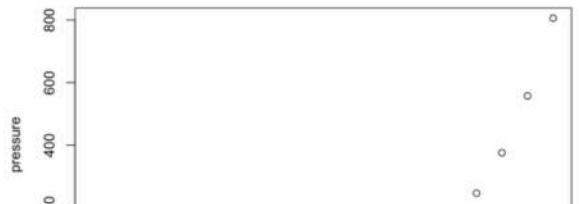
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Number reporting (APA style) with papaja

```
111
112  ### APA-style numbers
113
114
115  The observed values were
116  `r printnum(1)`,
117  `r printnum(0.0001)`, and
118  `r printnum(0.2)`.
119
120  The p values were
121  `r printp(1)`,
122  `r printp(0.0001)`, and
123  `r printp(0.2)`.
124
125  ```
```

APA-style numbers. The observed values were 1.00, 0.00, and 0.20.

The p values were > .999, < .001, and .200.

Reporting statistics with papaja

````
138
139  ```{r result="asis"}
140  oasis$ID <- 1:nrow(oasis)
141  aresult <-  afex::aov_ez(data=oasis, dv="nWBV", between =
     c("Gender","CDRc"),id = "ID")
142
143  apaa <- apa_print(aresult)
144  ```
145
146  Results were analyzed using a two-factor ANOVA. We found no
147  significant effect of gender (`r apaa$full_result$Gender`)
148  but a significant effect of diagnosis (`r
     apaa$full_result$CDRc`).
149  The interaction was not significant (`r
     apaa$full_result$Gender_CDRc`).
150
````

Results were analyzed using a two-factor ANOVA. We found no significant effect of gender ($F(1, 122) = 1.21$, $MSE = 0.00$, $p = .273$, $\hat{\eta}^2_G = .010$) but a significant effect of diagnosis ($F(1, 122) = 32.11$, $MSE = 0.00$, $p < .001$, $\hat{\eta}^2_G = .208$). The interaction was not significant ($F(1, 122) = 0.04$, $MSE = 0.00$, $p = .835$, $\hat{\eta}^2_G = .000$).

APA Table

```
mod1 <-
  " musicG =~ X1+X2+X3+X4+X5+X7+X8+X9+X10+X11+X12+X13 "

model1.fit <- cfa(mod1,data = model.dat)
fitMeasures(model1.fit,c("CFI","RMSEA","BIC"))
```

```
##      cfi    rmsea      bic
##    0.882    0.165 1439.369
```

```
sl<- standardizedSolution(model1.fit)
knitr::kable(sl,digits = 2,caption = "Standardized Estimates")
```

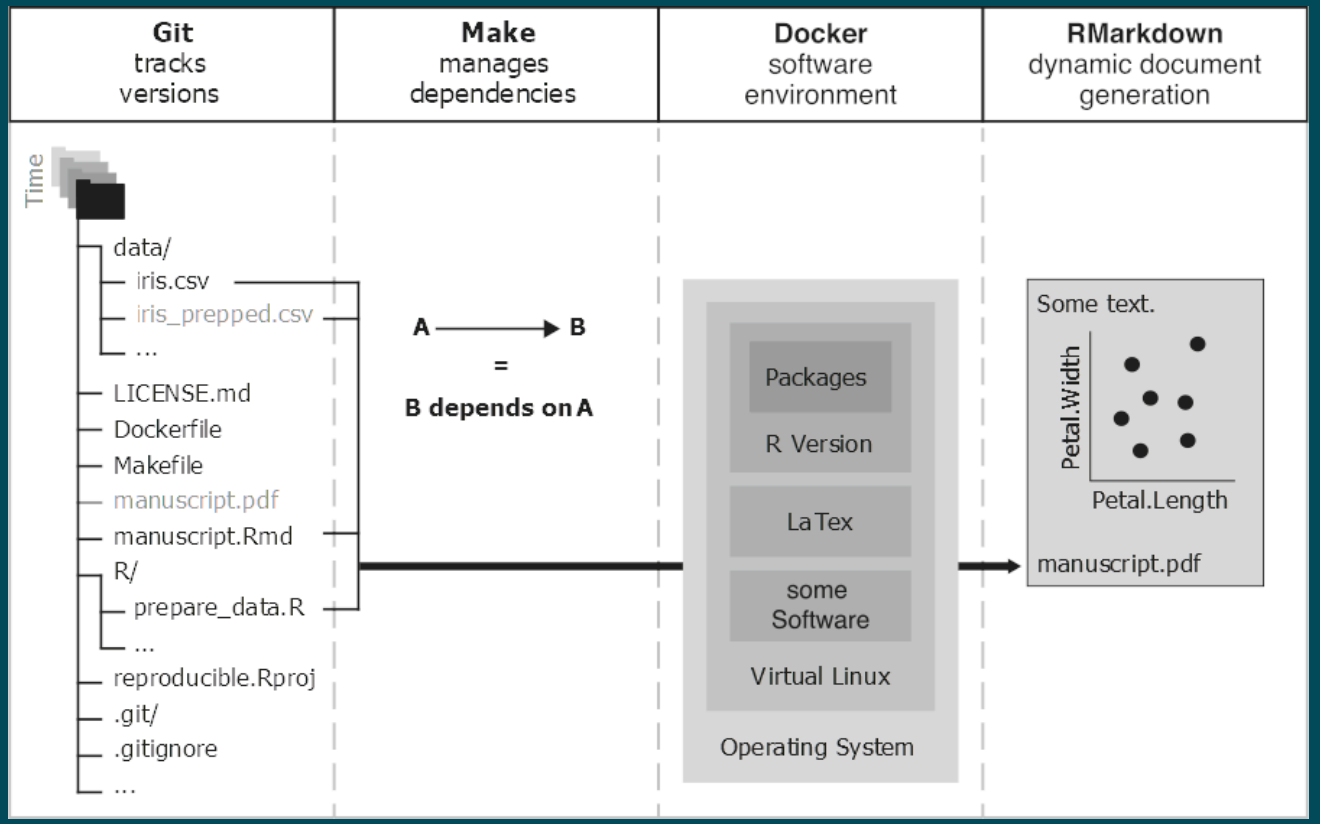
Table 1: Standardized Estimates

lhs	op	rhs	est.std	se	z	pvalue	ci.lower	ci.upper
musicG	=~	X1	0.89	0.03	29.27	0	0.83	0.95
musicG	=~	X2	0.90	0.03	33.27	0	0.85	0.95
musicG	=~	X3	0.76	0.06	13.60	0	0.65	0.87
musicG	=~	X4	0.81	0.05	17.61	0	0.72	0.90
musicG	=~	X5	0.88	0.03	27.11	0	0.81	0.94

Thank you, statcheck, we'll take it from here

The repro package

Four Elements of Reproducibility



from Peikert and Brandmaier (2020)

YAML header (repro)

`repro` extends the YAML header of **R Markdown** to track dependencies on code (internal scripts and external packages) and data files.

...and takes care of preparing your project for **git**, **Make**, and **Docker**

YAML header (R Markdown)

A standard YAML header:

```
---
title: My worst academic fails
author: Andreas Brandmaier
date: 2020-11-17
output: html_document
---
```

Example

```
---
title: My worst academic fails
author: Andreas Brandmaier
date: 2020-11-17
repro:
  packages:
    - usethis
    - fs
    - aaronpeikert/repro@d09def75df
  scripts:
    - R/clean.R
  data:
    mycars: data/mtcars.csv
output: html_document
---
```

System setup with repro

Load the package:

```
library(repro)
```

Run some checks:

```
check_git()
```

```
## ✓ Git is installed, don't worry.
```

```
check_make()
```

```
## ✕ Make is not installed.
## ℹ We recommend Chocolately for Windows users.
## ✕ Chocolately is not installed.
## • To install it, follow directions on:
##   'https://chocolatey.org/docs/installation'
```

Project setup with `repro`

- Use `repro::automate()` to make an existing RMarkdown analysis reproducible
- creates a Dockerfile & Makefile based on every RMarkdown in the project folder
- use `automate_load_packages()` to load all packages in your script, `automate_load_data`() for data, and `automate_load_scripts()` to attach external scripts

Under the hood

```
cat("
✓ Setting active project to 'C:/Users/andreas.brandmaier/LokaleDokumente/GitHub/reproducibility_wor
✓ Directory `.repro` created!
✓ Writing '.repro/Dockerfile_base'
✓ Writing '.repro/Dockerfile_packages'
✓ Writing '.repro/Dockerfile_manual'
✓ Writing 'Dockerfile'
✓ Writing '.repro/Makefile_Rmds'
✓ Writing '.repro/Makefile_Docker'
✓ Writing '.dockerignore'
• Modify '.dockerignore'
✓ Writing 'Makefile'
• Modify 'Makefile'
• Maybe you want to add:
  'repro.html'
  to the 'Makefile'-target 'all'.")
```

```
##
## ✓ Setting active project to 'C:/Users/andreas.brandmaier/LokaleDokumente/GitHub/reproducibility_worksho
## ✓ Directory `.repro` created!
```

Docker

```
# Generated by repro: do not edit by hand
# Please edit Dockerfiles in .repro/
FROM rocker/verse:4.2.2
WORKDIR /home/rstudio
RUN apt-get update -y && apt-get install -y rsync
RUN tlmgr install collection-latexrecommended
RUN install2.r --error --skipinstalled \
  here \
  patchwork \
  qrcode \
  showtext \
  stargazer \
  svglite \
  tidyverse \
  xaringanthemer
RUN installGithub.r \
  aaronpeikert/repro@fc7e884
```

Reproduction

To reproduce a project completely:

```
library(repro)
reproduce()
```

What could possibly go wrong?

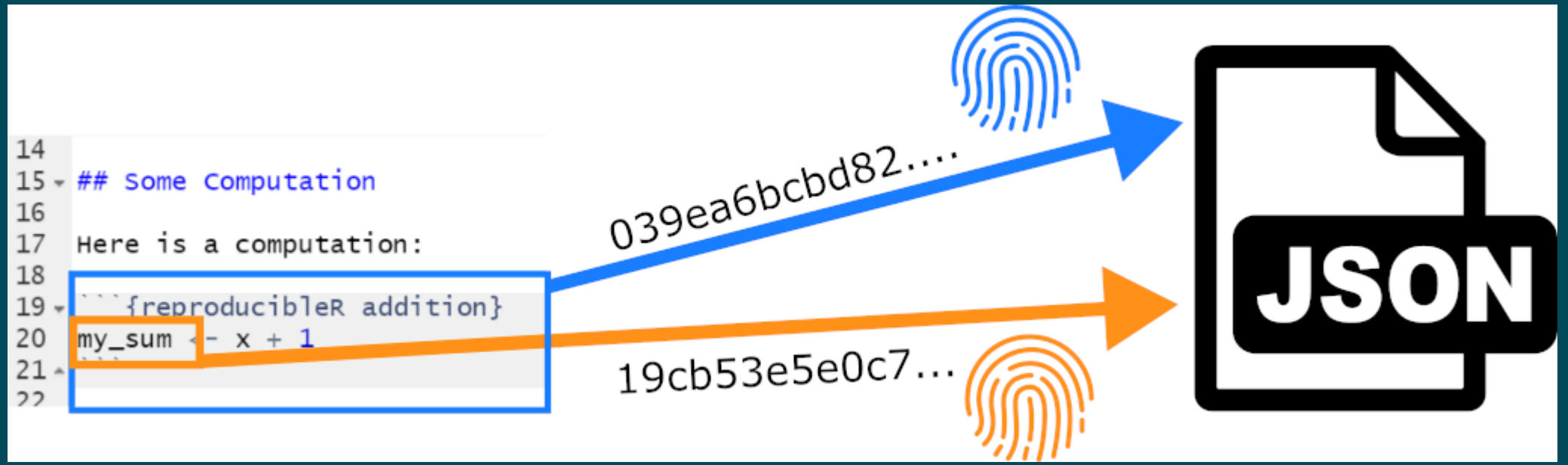
Problems

- Docker not available (e.g., no admin rights, unsupported computing platform)
- After upgrading to new R, new package versions, new computer, how do we know whether results still reproduce?

Solution

reproducibleRchunks are just like regular R Markdown chunks but verify reproducibility of computations

reproducibleRchunks



Reproduction reports

Here is a computation:

```
my_sum <- x + 1
```

Code Chunk Reproduction Report

Creating reproduction file This seems to be the first run of the R Markdown file including reproducible chunks. Storing reproducibility information for variables:

- my_sum

Reproduction reports (Success)

Here is a computation:

```
my_sum <- x + 1
```

Code Chunk Reproduction Report

- ✅ my_sum: REPRODUCTION SUCCESSFUL

Reproduction reports (Failed)

Here is a computation:

```
my_sum <- x + 1
```

Code Chunk Reproduction Report

- ❌ my_sum: **REPRODUCTION FAILED** Fingerprints are not identical.

How does it work?

```
set.seed(42)
numbers <- sample(1:10, 5)
```

becomes a serialized JSON file (whose changes are trackable via *git*):

{
"type": "integer",
"attributes": {},
"value": [1, 5, 10, 8, 2]
}

but really, we store a hash fingerprint for privacy and space reasons

What are valid variables?

All of the following variables (`x` of class *integer*, `y` of class *character*, `z` of class *lm*, and `lst` of class *list*) are valid examples for the automated reproducibility checks:

```
x <- 1:10
y <- "qr"
z <- lm(x~1, method=y)
lst <- list(x, y, z)
```

What should be reproducible?

(Almost) Everything

Some things we do not care about

Time information such as:

- current time or date
- run time of a model
- time elapsed for a given computation

Other information that changes between sessions (shoud be avoided anyhow):

- user names
- local paths

Extract only necessary bits from a model and verify them

The `broom` package has a generic function `tidy` with model-specific functions (e.g., lm, anova, factanal, glm, lavaan, gam,...) to extract all relevant parameter estimates:

```
x <- rnorm(10)
y <- rnorm(10)
broom::tidy(lm(y~x))
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    0.568     0.204      2.79  0.0237
## 2 x              0.487     0.205      2.38  0.0448
```

Reproducibility + Preregistration

Outlook: PAC

- preregistration currently entail a lot of work because we need two write two manuscripts (prereg + final ms)
- preregistrations still suffer from ambiguities of written language, omissions, underdefined processes
- With reproducible pipelines, we could move towards Preregistration as Code (PAC)

Outlook: PAC

Advantages

- Write your manuscript once with everything but the discussion
- Generate all tables and figures based on simulated data
- Once real data is collected, replace data, reproduce manuscript
- Deviate from original plan if necessary, summarize, also: all changes are *transparently* documented in git
- Write discussion, done.

How would this look in practice?

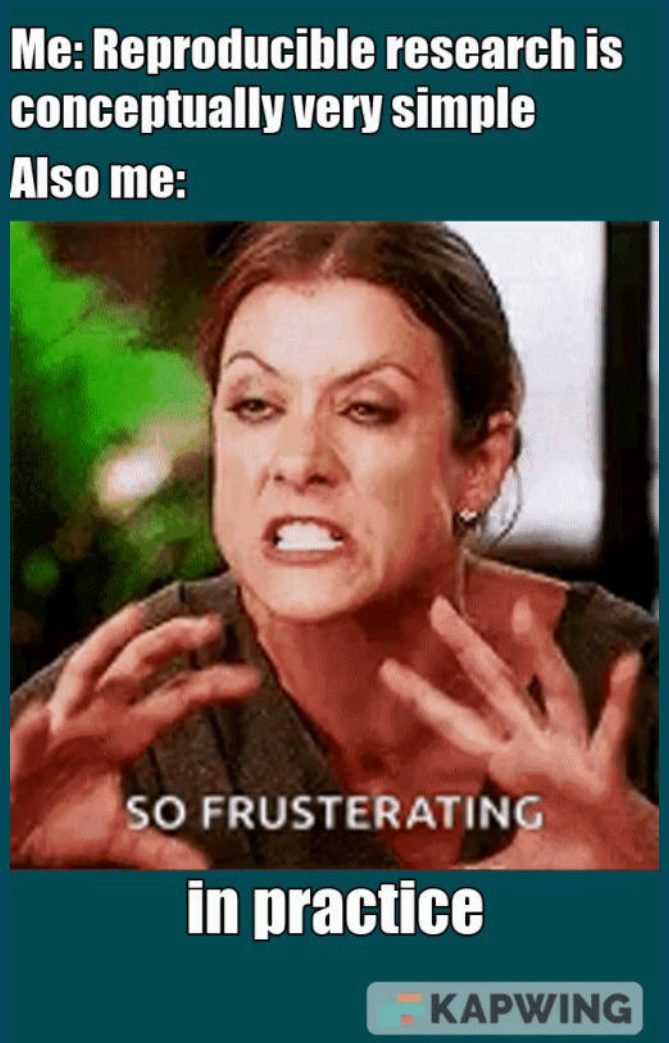
Example online here ("gender differences in Machiavellianism") (Peikert, Van Lissa, and Brandmaier, 2021) including power analysis

![https://github.com/aaronpeikert/repro-tutorial/blob/main/preregistration.Rmd]

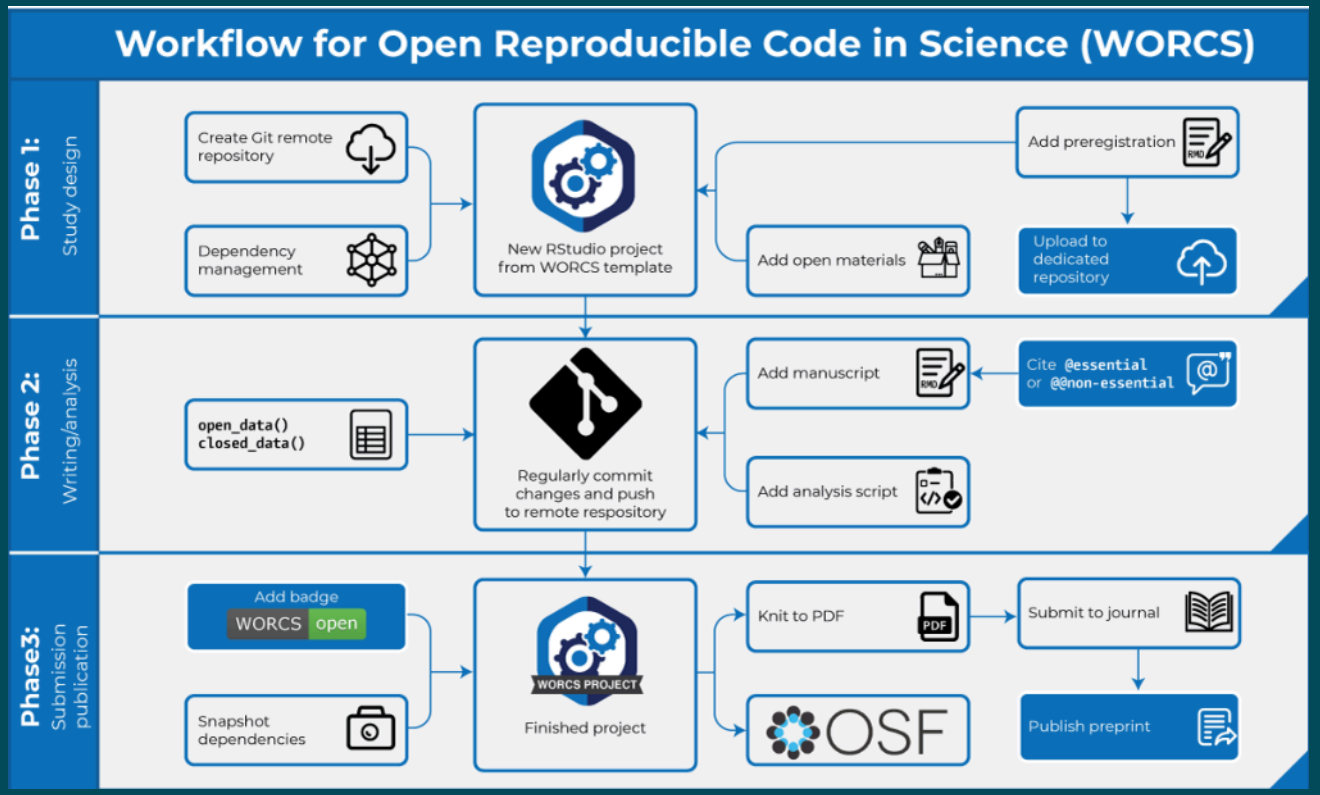
Modifications we had to make after the simulated PAC:

- recoding gender from (1,2) to (male, female)
- removal of NA values was missing
- half of the items of Machiavellianism are reverse-coded

Take-home message



Where to start



(Van Lissa, Brandmaier, Brinkman et al., 2021)

Where to start

Link to this talk:

Thank you!

Many thanks to the team: **Aaron Peikert**, **Caspar van Lissa**

If you want to engage:

- Ask us a question via a github issue
- Follow @brandmaier on twitter or **@brandmaier.bsky.social**
- Read our papers on the topic (see refs)
- Contribute to our repro repository on GitHub or reproducibleRchunks

License Information

- This presentation file is distributed under CC0 unless otherwise stated or other sources are included
- The GNU logo was provided under the Free Art License.

References

Bar-Anan, Y. and B. A. Nosek (2012). "Reporting intentional rating of the primes predicts priming effects in the affective misattribution procedure". In: *Personality and Social Psychology Bulletin* 38.9, pp. 1194-1208.

Hardwicke, T. E., M. B. Mathur, K. MacDonald, et al. (2018). "Data availability, reusability, and analytic reproducibility: Evaluating the impact of a mandatory open data policy at the journal Cognition". In: *Royal Society open science* 5.8, p. 180448.

Landau, W. M. (2021). "The targets R package: a dynamic Make-like function-oriented pipeline toolkit for reproducibility and high-performance computing". In: *Journal of Open Source Software* 6.57, p. 2959. URL: https://doi.org/10.21105/joss.02959.

Lin, Z., A. Werner, U. Lindenberger, et al. (2021). "Assessing music expertise: the Berlin Gehoerbildung Scale". In: *Music Perception: An Interdisciplinary Journal* 38.4, pp. 406-421.

Obels, P., D. Lakens, N. A. Coles, et al. (2020). "Analysis of open data and computational reproducibility in registered reports in psychology". In: *Advances in Methods and Practices in*

References

Peikert, A. and A. M. Brandmaier (2021). "A reproducible data analysis workflow with R Markdown, Git, Make, and Docker". In: *Quantitative and Computational Methods in Behavioral Sciences*, pp. 1-27.

Peikert, A., C. J. Van Lissa, and A. M. Brandmaier (2021). "Reproducible research in R: A tutorial on how to do the same thing more than once". In: *Psych* 3.4, pp. 836-867.

Thibault, R. T., E. A. Zavalis, M. Malički, et al. (2024). "An evaluation of reproducibility and errors in published sample size calculations performed using G* Power". In: *medRxiv*, pp. 2024-07.

Van Lissa, C. J., A. M. Brandmaier, L. Brinkman, et al. (2021). "WORCS: A workflow for open reproducible code in science". In: *Data Science* 4.1, pp. 29-49.