Automating Reproducibility

A Reproducible Data Analysis Workflow with RMarkdown, Git, Make, and Docker

Aaron Peikert[1]

[1] Center for Lifespan Psychology, Max Planck Institute for Human Development

Slides: `https://github.com/aaronpeikert/repro-workshop`

“Insanity is doing the same thing over and over again and expecting different results.”

– Albert Einstein

> “Insanity is doing the same thing over and over again and expecting different results.”
>
> – Albert Einstein

As it turns out, doing the
same thing
is pretty complicated.

Reproduction ≠ Replication

If everything is already there:

- published paper
- data originally used
- code originally used

Reproduction ≠ Replication

If everything is already there:

- published paper
- data originally used
- code originally used

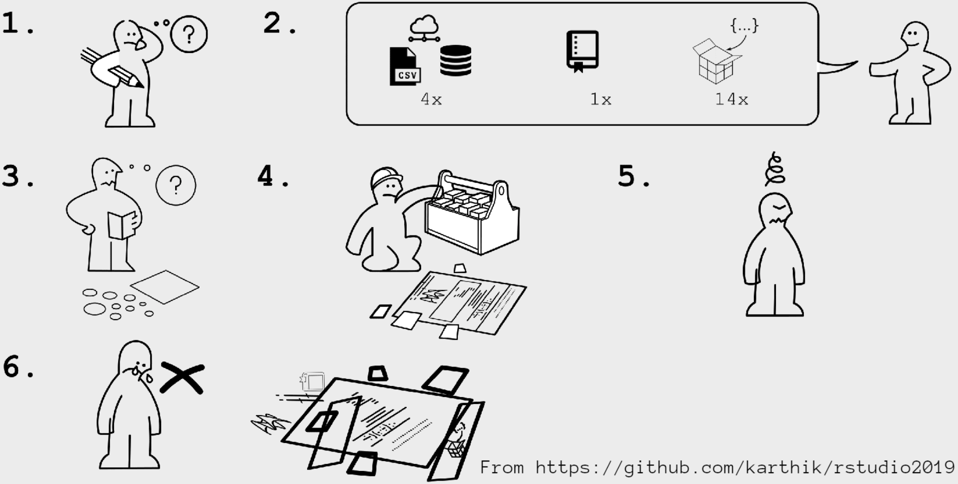
Shouldn't that be enough for Reproducibility?

Reproduction ≠ Replication

If everything is already there:

- published paper
- data originally used
- code originally used

Shouldn't that be enough for Reproducibility? Unlikely.



From https://github.com/karthik/rstudio2019

Problems

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

Problems

1. Copy&paste errors (e.g., inconsistency between reported result and reproduced result)
2. Multiple versions of scripts/data (e.g., the dataset has changed over time, i.e., was further cleaned or extended)

Problems

1. Copy&paste errors (e.g., inconsistency between reported result and reproduced result)
2. Multiple versions of scripts/data (e.g., the dataset has changed over time, i.e., was further cleaned or extended)
3. unclear which scripts should be executed in which order

Problems

1. Copy&paste errors (e.g., inconsistency between reported result and reproduced result)
2. Multiple versions of scripts/data (e.g., the dataset has changed over time, i.e., was further cleaned or extended)
3. unclear which scripts should be executed in which order
4. Broken software dependencies (e.g., analysis broken after an update, missing package)

Problems

Day I :

1. Copy&paste errors (e.g., inconsistency between reported result and reproduced result)
2. Multiple versions of scripts/data (e.g., the dataset has changed over time, i.e., was further cleaned or extended)
3. unclear which scripts should be executed in which order
4. Broken software dependencies (e.g., analysis broken after an update, missing package)

Problems

Day II:

1. Copy&paste errors (e.g., inconsistency between reported result and reproduced result)
2. Multiple versions of scripts/data (e.g., the dataset has changed over time, i.e., was further cleaned or extended)
3. unclear which scripts should be executed in which order
4. Broken software dependencies (e.g., analysis broken after an update, missing package)

Lessons from software engeniering

Day I :

1. Dynamic document creation
2. Version control
3. Dependency tracking
4. Software management

Lessons from software engeniering

Day II:

1. Dynamic document creation
2. Version control
3. Dependency tracking
4. Software management

Tools for R Users

In the R Universe and beyond, the most flexible tools are:

1. Dynamic document creation = RMarkdown*
2. Version control = Git**
3. Dependency tracking = Make**
4. Software management = Docker**

* RMarkdown supports more then 40 languages e.g.:
Python, Julia, SAS, Scala & Octave
** Language agnostic

Goals

Day I: Basics of Reproducibility

- Introduction to usethis
- Medeocre Level RMarkdown
- Git Basics
- Advanced GitHub

This skillset gets you 80% reproducibility.
My goal was to compress the frustration of months, into a few hours.

Download this GitHub repo

Please run in RStudio:

```r
usethis::use_course("aaronpeikert/repro-workshop")
```

If you are asked if you want to delete the ZIP file say yes (or no, I don’t care).

Goals

Day II: Advanced Reproducibility

- Automated use of Docker and Make
- Basics of Make
- Basics of Docker

This skillset gets you bullet proof reproducibility, but is quite technical.

A little taste

In the best case, you can put all instructions for reproducing something into a tweet.

```
git clone https://github.com/aaronpeikert/workflow-showcase.git
cd workflow-showcase
make build
make all DOCKER=TRUE
```

Specify Everything

The relations between
code, data, results and their environment
need to be unambiguously specified.

Specify Everything

The relations between
code, data, results and their environment
need to be unambiguously specified.

Why should I care?

Productivity:

- reuse
- easier collaboration

Why should I care?

Productivity:

- reuse
- easier collaboration!

Why should I care?

Good scientific practice:

- reproducibility is a precondition for replication
- increases transparency and (longterm) accessibility

To do (1 min)

This color indicates, that you should do this.

This color indicates, that you may want to do this (if you have time/energy).

Projects

Self contained folder
never `setwd()` again

Projects (4 min)

Self contained folder
never `setwd()` again

RStudio → File → New Project → New Directory → New Project
Project names tend to stick, think hard.
Or not: name generator!

Try in R `getwd()`

Take a look at the templates in:
RStudio → File → New Project → New Directory

Usethis

`usethis` automates commonly used steps for RProjects.
it is the backbone of `repro`
any tasks it does for you, you can do manually

Try in R `usethis::browse_github("usethis")`
Or open browser, google ”usethis github”, click first link.

Code of Conduct

A CoC signals a welcoming and respectfull environment.

Code of Conduct (2 min)

A CoC signals a welcoming and respectfull environment.

Give your project a Contributor Covenant CoC, in R with:

```
usethis::use_code_of_conduct()
```

Code of Conduct for this Workshop

I expect you to be nice to everyone, but this workshop has no formal code of conduct.

If you observe inappropriate behavior (not only towards yourself), approach Aaron or Tina via private chat.
Or ask someone else to approach us for you instead.
If you want to remain anonymous send a mail via www.guerrillamail.com/compose to peikert@mpib-berlin.mpg.de

If necessary, we will exclude a delinquent from further participating in this workshop.

Licenses

Without a license no one can reuse what you did.

Licenses (10 min)

Without a license no one can reuse what you did.

Give your project a CC0 license, in R with:

```
usethis::use_cc0_license()
```

Completely unrelated tech check:

```
repro::check_github_token()
```

Learn about one of the following licenses:
GPL: https://choosealicense.com/licenses/gpl-3.0/
MIT: https://choosealicense.com/licenses/mit/
CC0: https://creativecommons.org/publicdomain/zero/1.0/
CC-BY: https://creativecommons.org/licenses/by/4.0/

Summarise what you have learned for your neighbor. Discuss.

README.md

A README is the first thing that people visiting see.

README.md (5 min)

A README is the first thing that people visiting see.

Give your project a README.md, in R with:

```
use_readme_md()
```

Try out Markdown, paste:

```
#This is a Header
*italic* / **bold**
```

Click on Preview

Try some things from:
RStudio → Help → Markdown Quick Reference

README.Rmd

A README.Rmd generates a README.md with R code!

README.Rmd (10 min)

A README.Rmd generates a README.md with R code!

Give your project a README.Rmd, in R with:

```
use_readme_rmd()
```

Try out RMarkdown, paste:

~~~
This is a dynamically calculated number: `r 1 + 1`
Here you see the code:
```{r number-two}
2 * 5
```
~~~

Click on Preview

Try to add a plot:

```
with(mtcars, plot(mpg, hp))
```

Code Chunk Options

~~~
```{language chunk-name, option=value}
some_code
```
~~~

Insert with `Ctrl + Alt + I`

Code Chunk Options

~~~
```{language chunk-name, option=value}
some_code
```
~~~

Insert with `Ctrl + Alt + I`

~~~
```{r nocode, echo=FALSE}
hidden_code <- function(x){
  visible_result <- x
  return(visible_result)
}
```
~~~

~~~
```{r noresults, results='hide'}
visible_code <- function(x){
  visible_result <- x
  return(hidden_result)
}
```
~~~

Code Chunk Options (5 min)

~~~
```{language chunk-name, option=value}
some_code
```
~~~

Insert with `Ctrl + Alt + I`

Try out: `echo=FALSE` and then `results='hide'`
Figure out what: `include=FALSE` does.

Figure out how to hide (hint: `warning=?`):

~~~
```{r noresults, results='hide'}
warning("Look at me.")
```
~~~

Figure out how deal with (hint):

~~~
```{r noresults, results='hide'}
stop("Figure me out.")
```
~~~

Code Chunk Options (5 min)

Paste:

~~~
```{r slow, cache=TRUE}
slow_square <- function(x){
  Sys.sleep(5)
  x * x
}
slow_square(4)
```
~~~

Render. Render again. Observe needed time.

Try to generate a vector graphic:

~~~
```{r pretty, dev = 'svglite'}
library(ggplot2)
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
```
~~~

Figure out how to use higher quality pixel graphics with the package: `ragg`

YAML Metadata

Metadata for Markdown is stored in a format called YAML:

```
---
title: "Untitled"
author: "Jane Doe"
date: "3/29/2021"
output: html_document
---
```

RStudio → File → New File → R Markdown... → OK
Ctrl + S → ”anynameyouwant.Rmd”
Change Author + Title

Change: `html_document` → `word_document`

Try to add a table of contents:

```
output:
  html_document:
    toc: TRUE
```

YAML Metadata (5 min)

Metadata for Markdown is stored in a format called YAML:

```
---
title: "Untitled"
author: "Jane Doe"
date: "3/29/2021"
output: html_document
---
```

RStudio → File → New File → R Markdown... → OK
Ctrl + S → ”anynameyouwant.Rmd”
Change Author + Title

Change: `html_document` → `word_document`

Try to add a table of contents:

```
output:
  html_document:
    toc: TRUE
```

R Markdown Formats

RMarkdown can produce:

PDF, DOCX, HTML, MD

in hundreds of flavours!

R Markdown Formats (25 min)

RMarkdown can produce:

PDF, DOCX, HTML, MD

in hundreds of flavours!

Move to group rooms.

Try out one of the following formats (no particular order):

- prettydoc
- vignette
- xaringan
- ioslides
- Tufte Handout
- rmdformats

First Steps with Git

Git takes snapshots of your project files.
It has powerfull tools to compare snapshots.

First Steps with Git (10 min)

Git takes snapshots of your project files.
It has powerfull tools to compare snapshots.

Stay in group rooms. Help each other.

Try in R: `usethis::use_git()`
Change something in any file you like.
RStudio → Git Pane → Tic Changed File
Commit → "First Change" → Commit

If you have a GitHub Token:
Try in R `usethis::use_github()`

Or for a privat repro (only visible to you and via invite):

```
usethis::use_github(private = TRUE)
```

GitHub Fork

A fork is a copy on GitHub.
A clone is a copy on your computer.

GitHub Fork (10min)

A fork is a copy on GitHub.
A clone is a copy on your computer.

Stay in group rooms. Help each other.
Fork Repo:

```r
# Needs a GitHub Token!
# I show the manual way for others.
usethis::create_from_github(
"aaronpeikert/repro-workshop",
#creates new folder 'repro-workshop' at a
#default location, if you dont want this, uncomment:
#destdir = "path/to/some/folder",
fork = TRUE) # notice!
```

Skim happygitwithr.com/common-remote-setups.html

GitHub Fork (10min)

Only when you have NO GitHub Token.
Open: github.com/aaronpeikert/repro-workshop → Fork.

```r
usethis::create_from_github(
"yourname/repro-workshop",
#creates new folder 'repro-workshop' at a
#default location, if you dont want this, uncomment:
#destdir = "path/to/some/folder",
fork = FALSE) # notice!
usethis::use_git_remote(name = "upstream",
url="git@github.com:aaronpeikert/repro-workshop.git")
```

Skim happygitwithr.com/common-remote-setups.html

Reproduce `games.Rmd`

Time for the first reproduction!

Reproduce `games.Rmd` (5 min)

Time for the first reproduction!

Take a look at `games.Rmd`.
Install packages when necessary.
Click on Knit.

Try in R: `rmarkdown::render("games.Rmd")`
Do you see the green arrow in the Chunks? Click on it.

Change `games.Rmd` (5 min)

Make `games.Rmd` your own.

Change anything you like.
RStudio → Git Pane → Tic Changed File
Commit → ”Second Change” → Commit

RStudio → Git Pane → History → Look around.

Create a common repo (15 min)

Git is distributed. There are many copies.
Sometimes it is hard to keep track.

Delete folder `repro-workshop`.

Choose a group member at random.
Clone their repo with:
`usethis::create_from_github("random/repro-workshop")`

Only chosen one → `usethis::browse_github()`
Settings → Manage access → Invite a collaborator
Add all your group members.

Skim happygitwithr.com/common-remote-setups.html

Hunt for errors in `games.Rmd`

No one is perfect.

`games.Rmd` contains 10 intentional errors
(and a few real ones too).

Hunt for errors in `games.Rmd` (5 min)

No one is perfect.
`games.Rmd` contains 10 intentional errors
(and a few real ones too).

Find an error.

File the error as issue:
`usethis::browse_github_issues()`
Make sure you are in the repo of the chosen one.

Assign yourself to one of the other issues.

Take a look at the issues of your favorite package:
`usethis::browse_github_issues("ggplot2")`

Correct errors in `games.Rmd`

A branch sets up a parallel versions of your code.

A pull request (PR) integrates changes from a branch.

Correct errors in `games.Rmd` (15 min)

A branch sets up a parallel versions of your code.
A pull request (PR) integrates changes from a branch.

Create a branch:
`usethis::pr_init("fix-issue_number")`
Fix the error. Commit the file with message:
"fix #issue_number"

`usethis::pr_push()` → Create PR

Take a look at the original issue.

Review changes in `games.Rmd`

A PR is a place for discussion.

Review changes in `games.Rmd` (10 min)

A PR is a place for discussion.

As the reviewer, leave a general comment at the PR:

```r
# these slides do contain R code!
praise::praise(
  "${EXCLAMATION}! Thank you ${adverb_manner}!"
)

## [1] "WHEE! Thank you quickly!"
```

Admire a specific line e.g.: `This is primo!`

Merge PR.

Take a look at the original issue.

Pull changes in `games.Rmd`

A pull integrates changes from GitHub with your local copy.

Pull changes in `games.Rmd` (5 min)

A pull integrates changes from GitHub with your local copy.

Close your PR branch with `usethis::pr_finish()`

Then pull changes:
RStudio → Git Pane → Pull

Inspect RStudio → Git Pane → History.

Apply changes directly

A PR is the formal way to introduce changes.
It is the polite way and gives other the chance to chime in.
However, you can directly push trivial changes.

Apply changes directly (10 min)

A PR is the formal way to introduce changes.
It is the polite way and gives other the chance to chime in.
However, you can directly push trivial changes.

Add a file with your name. E.g. in R:

```
file.create("myname.txt")
```

Commit file.

Expect trouble.
Pull → Push → Repeat if necessary.
Inspect RStudio → Git Pane → History.

Learn about rebase.
Set it as default:

```
gert::git_config_global_set("pull.rebase", "true")
```

Merge conflicts in `games.Rmd`

If you are not highly coordinated e.g. via GitHub issues, expect trouble.

Merge conflicts in `games.Rmd` (10 min)

If you are not highly coordinated e.g. via GitHub issues, expect trouble.

Change the author of `games.Rmd` to yourself.
Commit file.

Expect a lot of trouble.
Pull → Push → Repeat if necessary.
Resolve conflicts in a way that all group members are authors.

Take a look at the emails GitHub has sent you.

Discuss PRs

If you write a manuscript and want your supervisor or collaborator to look at your changes, create a PR and request them as a reviewer.

Discuss PRs (10 min)

If you write a manuscript and want your supervisor or collaborator to look at your changes, create a PR and request them as a reviewer.

Propose some changes:
`usethis::pr_init("newchange")` → Change → Commit → `usethis::pr_push()`

Propose a change within another PR on GitHub.

Make changes to the PR locally via:
`usethis::pr_fetch(pr_number)` & `usethis::pr_push()`

Revert Changes

Anything you have committed, can always be retrieved.

Revert Changes (10 min)

Anything you have committed, can always be retrieved.

Inspect History on GitHub.

Watch out, uncommitted changes may be lost!
Look around with (in Terminal): `git checkout the_hash`
Come back with: `git checkout master`

Do NOT do this:
To destroy recent changes locally:

```
git reset the_hash --hard
```

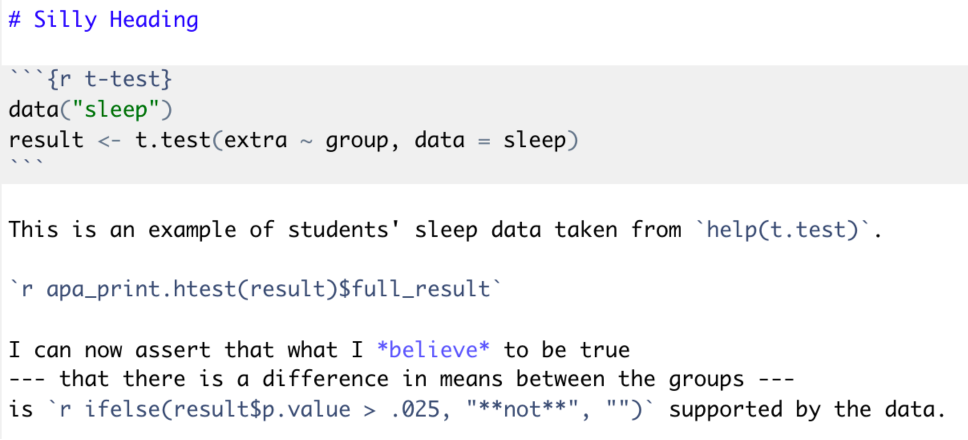
To destroy them then on GitHub:

```
git push --force-with-lease
```

RMarkdown—Literate Programming

Text and code are mixed
in a single source document
that can be dynamically compiled
into various representations:

- (APA conformable) manuscripts
- presentations
- websites
- books
- posters
- CV



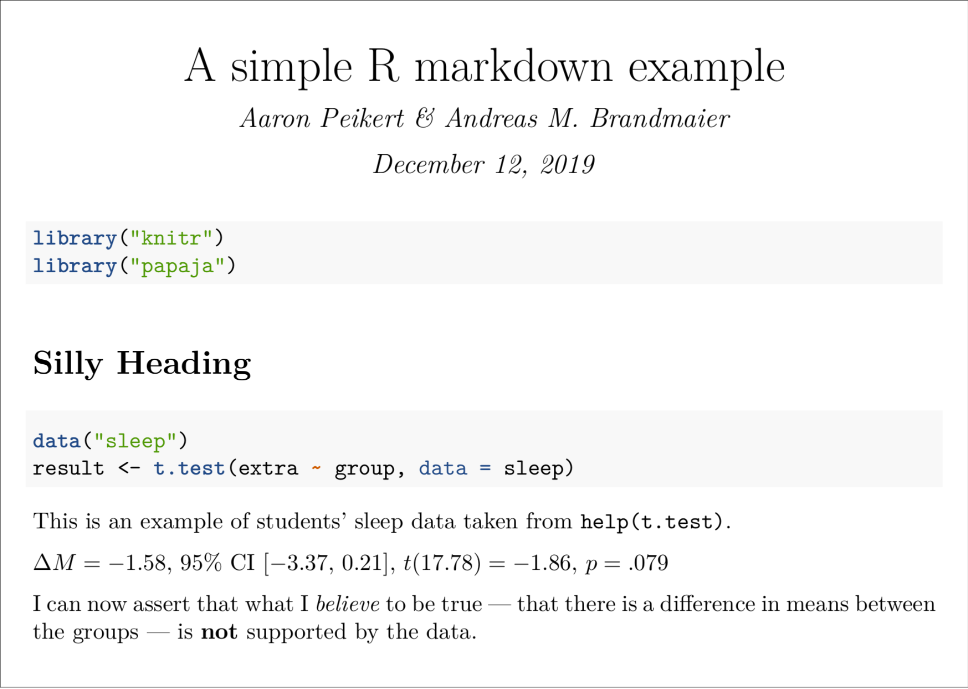
~~~
# Silly Heading

```{r t-test}
data("sleep")
result <- t.test(extra ~ group, data = sleep)
```

This is an example of students' sleep data taken from `help(t.test)`.

`r apa_print.htest(result)$full_result`

I can now assert that what I *believe* to be true
--- that there is a difference in means between the groups ---
is `r ifelse(result$p.value > .025, "**not**", "")` supported by the data.
~~~



A simple R markdown example

Aaron Peikert & Andreas M. Brandmaier

December 12, 2019

```r
library("knitr")
library("papaja")
```

Silly Heading

```r
data("sleep")
result <- t.test(extra ~ group, data = sleep)
```

This is an example of students' sleep data taken from `help(t.test)`.

$\Delta M = -1.58$, 95% CI $[-3.37, 0.21]$, $t(17.78) = -1.86$, $p = .079$

I can now assert that what I *believe* to be true — that there is a difference in means between the groups — is **not** supported by the data.

Git/GitHub—Version Control

Version control is a system that records changes to a set of files over time so that you can recall specific versions later.

It guarantees that code and data are exactly the same version as used for publication.

Make—Dependency Management

Make is a “recipe” language that describes how files depend on each other and how to resolve these dependencies.

```
cfcs-example.pdf: cfcs-example.Rmd data/CFCS.csv
  $(run) Rscript -e 'rmarkdown::render("$(current_dir)/$<")'

data/CFCS.csv: R/00load_data.R
  $(run) Rscript -e 'source("$(current_dir)/$<")'
```

Docker—Containerization

Docker is a lightweight virtual computer.
Dockerfiles are “recipes” that describe what to install on that virtual computer:

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

Advantages

Unambiguous

Advantages

Unambiguous Standardized

Advantages

Unambiguous Standardized Portable

Advantages

Unambiguous Standardized Portable Automated

Simplifying the tools

These tools require extensive training and need much time to configure correctly.

Simplifying the tools

These tools require extensive training and need much time to configure correctly.

The R package ’repro’ abstracts away the concrete technical implementation:

Simplifying the tools

These tools require extensive training and need much time to configure correctly.
The R package ’repro’ abstracts away the concrete technical implementation:

Live Demo

Cheat I — Custom Recipes

A few variations on zip:

```
data/mtcars.csv: data/mtcars.csv.zip
  unzip -p data/mtcars.csv.zip > data/mtcars.csv
```

```
data/mtcars.csv: data/mtcars.csv.zip
  unzip -p $< > $@
```

```
data/mtcars.csv: data/mtcars.csv.zip
  Rscript -e "unzip('data/mtcars.csv.zip', exdir = 'data/')"
```

Cheat II — Custom Software

Install 7zip in Docker:

```
RUN apt-get update -y &&\
  apt-get install -y --no-install-recommends p7zip
```

Use 7zip in Make:

```
data/mtcars.csv: data/mtcars.csv.zip
  7z e -y -odata/ data/mtcars.csv.zip
```

Backup Slides

Disadvantages

- requires complex software infrastructure
- depends on for-profit services
- diverges from the standard manuscript workflow

Focus: Longterm Archive

All software is bundled into the container, therefore all we need is:

- container software
- storage infrastructure

Focus: Longterm Archive

All software is bundled into the container, therefore all we need is:

- container software
- storage infrastructure

What happens when Docker and co. are not supported anymore?
Containers can be converted into a full system image ensuring support for decades.

Focus: Computing infrastructure

Dependency management + containerization

Focus: Computing infrastructure

Dependency management + containerization

=

distributed computation

Focus: Computing infrastructure

distributed computation
on
High Performance Computing cluster

Focus: Computing infrastructure

distributed computation
on
Cloud Computing infrastructure

Focus: Computing infrastructure

distributed computation
on
Cloud Computing infrastructure
=
Upon change, the manuscript is rerendered assuring reproducibility.

Focus: Modularity

- repro is a modular system

Focus: Modularity

- repro is a modular system
- potential integration of other workflows

Focus: Modularity

- repro is a modular system
- potential integration of other workflows
- “Lego system of reproducibility tools”

References

Slides:
`https://github.com/aaronpeikert/repro-talk`
Package:
`https://github.com/aaronpeikert/repro-thesis`
Workflow:
`https://doi.org/10.31234/osf.io/8xzqy`

Thank you

Questions?