WHU
Otto Beisheim School of Management

Data Management - Doctoral Orientation Week

Dr. Victor van Pelt

WHU – Otto Beisheim School of Management

August 21, 2025

Opening Question

If your laptop died right now and we handed you a new one, how long until you can reproduce your last analysis?

1. Less than one hour (I'm fully backed up and scripted)
2. Between one hour and a day (it's possible but painful)
3. More than one day (I'd be stuck)

Why should YOU care about Data Management?

Storing data, code, analyses, and papers can quickly turn into a mess:

descriptives	17/01/2021 16:18	Rich Text Format
descriptives	29/04/2020 17:40	LaTeX Source File
descriptivestime	17/01/2021 16:19	Rich Text Format
descriptivestime	13/01/2021 12:01	LaTeX Source File
do file	25/01/2021 23:56	Stata Do-file
do_clean	06/07/2023 15:47	Stata Do-file
dyndoc	12/12/2022 17:39	Brave HTML Docu...
effort	27/02/2020 16:28	Adobe Acrobat D...
epq	17/01/2021 16:22	Rich Text Format
epq	28/07/2020 12:12	LaTeX Source File
epq	07/02/2023 13:30	Microsoft Excel 97...
frequencies	17/01/2021 16:19	Rich Text Format
frequencies	29/04/2020 17:40	LaTeX Source File
gee	09/03/2020 11:33	LaTeX Source File
gee	09/03/2020 11:33	Text Document

Why should YOU care about Data Management?

Story Time!

But there is more at stake!

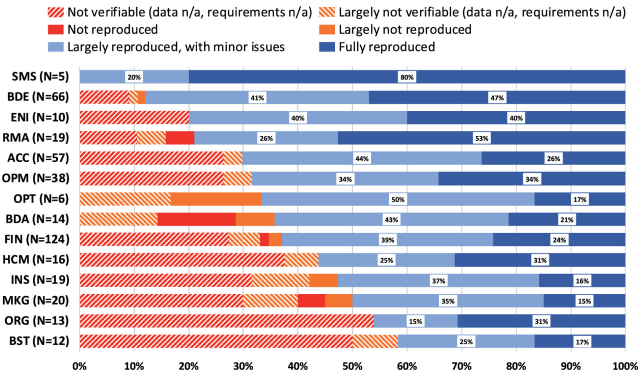
In 2019, *Management Science* introduced a Data and Code Disclosure Policy:

"Authors of accepted papers… must provide… the data, programs, and other details of the experiment and computations sufficient to permit replication."

A 2023 study examined nearly 500 articles published under this policy to see if the results could, in fact, be reproduced (Fišar et al. 2023).

What share of these articles do you think were largely reproducible?

But there is more at stake!



About 30% of the articles could not be reproduced, even under the new policy (Fišar et al. 2023)

But there is more at stake!

- Nearly one in three papers fails the reproducibility test.
- This is why good data and code management is not only a “nice to have.”
- It’s also essential for credible research and for us to trust the insights it produces.
- But this does not just depends on how professors manage their data and code.
- It all starts with:

Who cares? We have AI!

- People have made breathless claims about how AI is going to make learning code and working with data entirely obsolete.
- But people who insist on these misconceptions are often invested in selling the hype or they easily fall prey to believing in science fiction movies and sensational internet videos.
- LLMs are exciting, but to make them useful, you need to set realistic expectations:
  - LLMs are not conscious or sentient.
  - LLMs are far from perfect and even often wrong.
  - LLMs will not replace the need for you to code properly and organize your data.
- LLMs are exciting support tools, but you must vigilantly check everything.
- It's entirely valid to forgo LLMs altogether when coding and working with data.

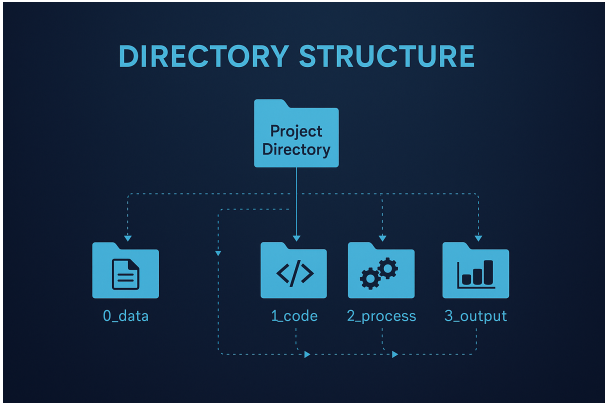
The main goals of this session:

- Share a few basic principles, techniques, and tools. Inspired by:
  - **Code and Data for the Social Sciences** (Gentzkow and Shapiro)
  - **How to keep your research projects organized** (De Kok)
  - My own mistakes and personal experience
- But let's make this interactive: Interupt, ask questions, + tell me about your experience.
- Disclaimer: Largely through the lens of a quantitative researcher.

What will we cover in this session?

1. Directory structure: How to organize your research project
2. Version control: Using Git and repositories (example using Github)
3. Integrating multiple languages: Example using Quarto

1. Directory Structure



1. Directory Structure: Fundamental Principles

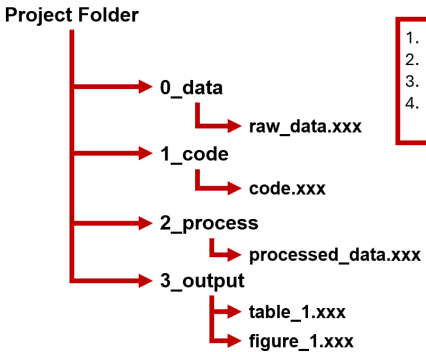
Following principles is critical, especially for projects with lots of data and code:

- Anyone can run the code anywhere and anytime, regardless of location.
- Anyone can understand the code and data without much effort.
- Anyone can update the code on the spot without breaking it (dynamic coding).

1. Directory Structure: How to accomplish this?

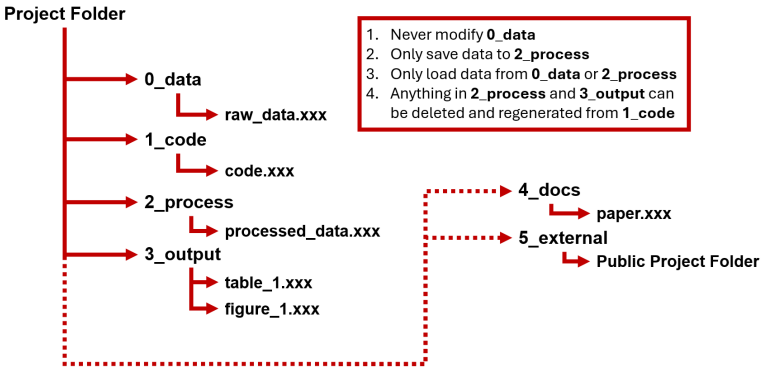
- Use a flow-inspired folder structure:
  - E.g., formatting files -> generating variables -> conducting analyses -> generating output.
- Leave the raw data untouched: You load it and use it to produce output:
  - E.g., raw data files -> input files -> process files -> output files.
- Use relative paths (e.g., "*../0_data/data.csv*") and not direct paths (e.g., "*C:/user[name]/Documents/research/project_1/0_data/data.csv*").
- The code is deterministic. Random and stochastic processes ideally require a seed.
- Do not leave blindspots in your coding and analysis
  - Use plenty of comments to explain what the code is doing.
  - Be as descriptive as you can in folder and filenames
- Use environmental variables for credentials and software paths

1. Directory Structure: An simple starting point



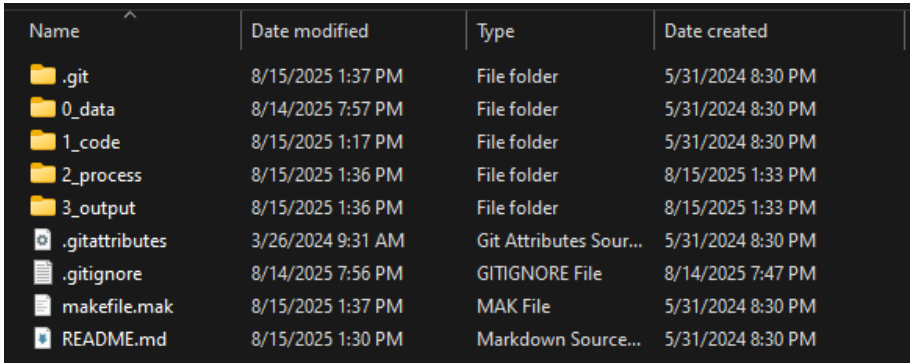
1. Never modify **0_data**
2. Only save data to **2_process**
3. Only load data from **0_data** or **2_process**
4. Anything in **2_process** and **3_output** can be deleted and regenerated from **1_code**

1. Directory Structure: Additional folders



1. Directory Structure: An example for Stata

Let's take a closer look at how this (in principle) could work:



Name	Date modified	Type	Date created
.git	8/15/2025 1:37 PM	File folder	5/31/2024 8:30 PM
0_data	8/14/2025 7:57 PM	File folder	5/31/2024 8:30 PM
1_code	8/15/2025 1:17 PM	File folder	5/31/2024 8:30 PM
2_process	8/15/2025 1:36 PM	File folder	8/15/2025 1:33 PM
3_output	8/15/2025 1:36 PM	File folder	8/15/2025 1:33 PM
.gitattributes	3/26/2024 9:31 AM	Git Attributes Sour...	5/31/2024 8:30 PM
.gitignore	8/14/2025 7:56 PM	GITIGNORE File	8/14/2025 7:47 PM
makefile.mak	8/15/2025 1:37 PM	MAK File	5/31/2024 8:30 PM
README.md	8/15/2025 1:30 PM	Markdown Source...	5/31/2024 8:30 PM

Clone this repository!

1. Directory Structure: Stata do-file under “1_code\”

1_code > do-file.do

```
//clear everything to ensure nothing came before
clear all

//go to the main director
capture cd 1_code   // if we're at the project root, step into 1_code (otherwise nothing)
cd ..               // now we're guaranteed to be at the project root

//import raw data and store as input data file in process-folder
// also for merging data
import delimited "0_data\gen_ai_earnings.csv"
save "2_process\gen_ai_earnings.dta", replace

//import input data, transform the data, and save as edited data
clear all
use "2_process\gen_ai_earnings.dta", replace

gen earnings_scaled = earnings / 10
drop if earnings_scaled<0

save "2_process\edit_gen_ai_earnings.dta", replace

//import edited data, conduct analysis to generate output
clear all
use "2_process\edit_gen_ai_earnings.dta"

eststo: regress earnings_scaled gen_ai, r

esttab using "3_output\Table 1.rtf", ///
replace stats(r2 F p df_m N) b(3) aux(se 3) star(* 0.10 ** 0.05 *** 0.01) obslast onecell nogaps ///
compress title(TABLE 1 - REGRESSIONS OF SCALED EARNINGS ON GENERATIVE AI) addnotes(p-levels are two-tailed, * p
eststo clear

//close stata
exit, STATA clear
```

1. Directory Structure: Reproducing all results instantly

Who can reproduce all their results using a single command or script?

1. Directory Structure: makefile.mak in root

makefile.mak

```
# Default goal is run. So you can just type "make"
.DEFAULT_GOAL := run

# Paths (insert your local paths or environmental variable)
STATA = "C:\Program Files\Stata18\StataSE-64.exe"
RSCRIPT = "C:\Program Files\R\R-4.5.1\bin\Rscript.exe"


# code variables
DOFILE = 1_code\code.do
RFILE    = 1_code\code.r


# run command runs the stata code
run: do

# Stata code. Run with "make do". Last line removes the log file.
do: $(DOFILE)
    $(STATA) /e /q do $(DOFILE) > NUL 2>&1
    @cmd /c "if exist "*.log"  del /q /f "*.log"

# R code. Run with "make r"
r:
    $(RSCRIPT) --vanilla --quiet "$(RFILE)" >NUL 2>&1

# housekeeping to remove stuck logfiles, quarto files, and clear process and output folders. Run with "make clean"
clean:
    @cmd /c "if exist "*.log"  del /q /f "*.log"
    @cmd /c "if exist "*.smcl"  del /q /f "*.smcl"


    @cmd /c powershell -NoLogo -NoProfile -Command "If(!(Test-Path '2_process')){New-Item -ItemType Directory -Path '
    @cmd /c powershell -NoLogo -NoProfile -Command "If(!(Test-Path '3_output')){New-Item -ItemType Directory -Path '

# Phony targets. Type "make r," "make do," or "make clean"
.PHONY: run do r        clean
```

1. Directory Structure: makefile.mak in root

You run “*make do*” and “*make r*” in the CLI

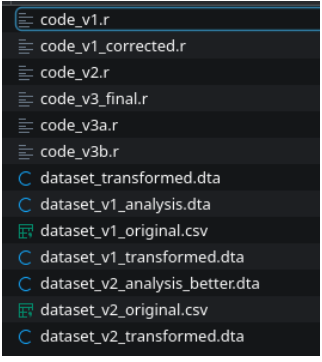
```
PS C:\Users\victo\OneDrive\Documents\github stuff\data_management_example> cd "C:\Users\victo\
OneDrive\Documents\github stuff\data_management_example"
PS C:\Users\victo\OneDrive\Documents\github stuff\data_management_example> make do
"C:\Program Files\Stata18\StataSE-64.exe" /e /q do 1_code\code.do > NUL 2>&1
PS C:\Users\victo\OneDrive\Documents\github stuff\data_management_example> make r
"C:\Program Files\R\R-4.5.1\bin\Rscript.exe"  --vanilla --quiet "1_code\code.r" >NUL 2>&1
PS C:\Users\victo\OneDrive\Documents\github stuff\data_management_example>
```

1. Directory Structure: Is your code slow?

- Does your code:
  - Take a long time to run?
  - Require too much memory?
- Ask your supervisor for a new laptop.
- Otherwise, consider looking into running your code remotely:
  - “super computer,”grid”, or “server” allows you to run your code using way more powerful hardware.

2. Version control: Opening question

Whose folders look something like this?



2. Version control

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

2. Version control: How to use version control

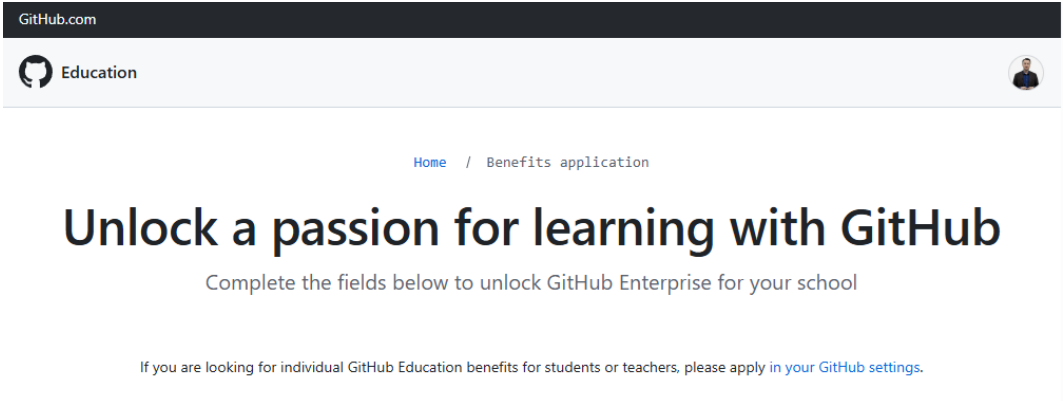
- Many cloud services have some forms of version control (e.g., Dropbox and Onedrive), but they can be too simplistic and unreliable.
- The most widely used software is called **git**
- You can run **git** locally using the CL, but it is easier to use an online provider.
- Saving your version control online reduces the likelihood you loose stuff.
- Three major **git** providers:
  - GitHub –> The best choice!
  - BitBucket
  - GitLab

2. Version control: What is Github?

- GitHub Inc. is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git as well as adding its own features.
- Why use it?
  - Clean and easy to use interface
  - Native support for all kinds of software
  - GitHub Desktop application is very simple and convenient
- Bonus feature: you can use GitHub pages to host your website for free! See mine **www.victorvanpelt.com**

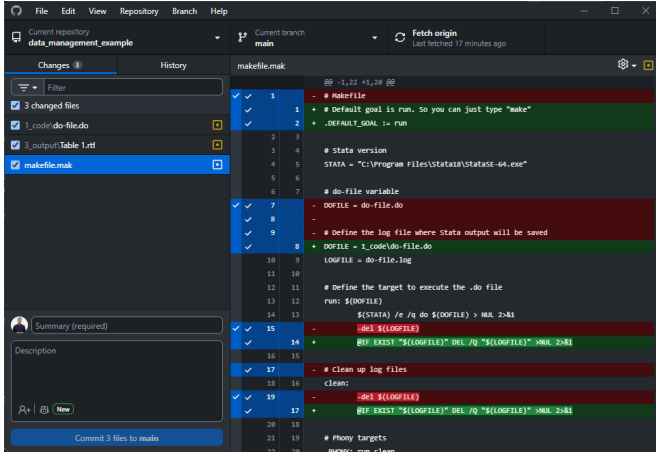
2. Version control: GitHub Enterprise

You can apply for GitHub Enterprise as a researcher or student **here**.



2. Version control: GitHub Desktop

Download the GitHub Desktop for Windows or MacOS **here**.



2. Version control: Basic Workflow

- First time:
  - Create a new repository on GitHub
  - Clone it to your computer (Copy URL and enter in GitHub Desktop)
- When you start working:
  - Sync with Github first: “Pull” changes
- When making changes:
  - Sync with Github first: “Pull” changes
  - Create commit (add summary and descriptions)
  - Push commit to GitHub.
- Use a README.md file to state the project’s title and a brief description of the repository.

2. Version control: .gitignore files

- There are lots of things you might not want to sync with Github
  1. Private or non-public data
  2. Personal credentials
  3. “Byproduct” files
- You would like to think about what to precisely sync with Github.
- A .gitignore file allows you to select files that should automatically be ignored by git.

2. Version control: .gitignore syntax

- The basic syntax can be found **here.**
- Some common expressions:
  - # comment
  - * any file name
  - ** any folder depth
  - / at the end folder
  - ! keep (don't ignore)

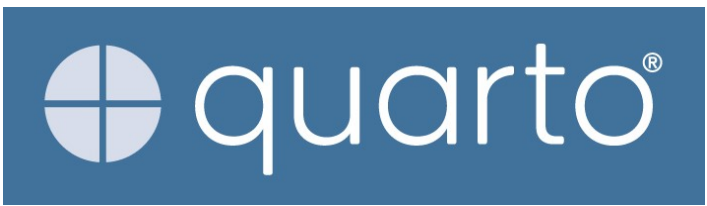
2. Version control: .gitignore example

.gitignore

```
# Ignore everything
*

# Keep these specific files
!.gitignore
!makefile.mak
!README.md

# Keep folder 0_data but not its contents
!0_data/
!0_data/.gitkeep

# Keep folder 1_code and its contents
!1_code/
!1_code/**

# Keep folder 2_process but not its contents
!2_process/
!2_process/.gitkeep

# Keep folder 3_output and all its contents
!3_output/
!3_output/**
```

2. Version control: Why should you care?

- Go back to any version at any time!
- Journals and institutions are increasingly requesting access to your research materials (i.e., code, instrument, and data).
- At the very least, they want to ensure it exists.
- At the very most, they will put a researcher on checking every part of your code (e.g., MS)
- This trend is growing… For example, two journals in my area:
  - Journal of Accounting Research
  - Management Science


quarto®

3. Integrating multiple languages: What's the remaining challenge?

- So, now you will have:
  1. A good directory structure
  2. Version control.
- A remaining challenge is that we use different software, coding languages, and files.
  - Python, R, Markdown, HTML, Office, Stata, LaTeX, etc.
  - Tables, datasets, code, and docs.
- There have been efforts over the past decade to put all processes under one umbrella.
- One system can integrate everything into one process: Quarto.

3. Integrating multiple languages: What is Quarto?

- Quarto is an open-source scientific and technical publishing system
- Quarto is not only used for data. It can use data to produce a wide variety of outputs:
  - Articles and papers
  - Presentations (this presentation is made using Quarto)
  - Dashboards
  - Websites
- Best integration support for Markdown, R, Python, Jupyter, and LaTeX.

3. Integrating multiple languages: How to get started?

Setup:

1. Install Quarto from the website: **https://quarto.org/docs/get-started/**.
2. Choose a coding environment (I recommend **VS Code**).
3. Install the **Quarto VS Code Extension** in VS Code.

Usage:

- Quarto uses .qmd files. Check the **guide**.
- You can produce output in formats, such as .ppp, .docx, .pdf and integrate R and Stata code.
- I also added a .qmd example to the GitHub repository to illustrate the functionality for a presentation.
- However, you can also use it to integrate everything into a research paper.

3. Integrating multiple languages: Quarto Presentation Example

```yaml
---
title: "Quarto Presentation Example"
author: "Victor van Pelt"
format:
  beamer:
    theme: Madrid
    fontsize: 9pt
---
```

Specify in the ”YAML” the type of qmd file

Use R as you would in code.r

3. Integrating multiple languages: Quarto Presentation Example

Quarto Presentation Example

Victor van Pelt

Table 1

TABLE 1 - REGRESSIONS OF SCALED EARNINGS ON GENERATIVE AI

	Model 1
(Intercept)	−1 115 423.471
	(30 072.242)
gen_ai	1 108 333.446
	(5044.061)
R2	0.988
N	590
df_m	1.000

* p < 0.1, ** p < 0.05, *** p < 0.01
p-levels are two-tailed, * p < 0.10, ** p < 0.05, *** p < 0.01; robust SE in parentheses.

3. Integrating multiple languages: makefile.mak extension

makefile.mak

```
# Default goal is run. So you can just type "make"
.DEFAULT_GOAL := run

# Paths (insert your local paths or environmental variable)
STATA = "C:\Program Files\Stata18\StataSE-64.exe"
RSCRIPT = "C:\Program Files\R\R-4.5.1\bin\Rscript.exe"
QUARTO  = "C:\Program Files\Quarto\bin\quarto.exe"

# code variables
DOFILE = 1_code\code.do
RFILE   = 1_code\code.r
QMD     = 1_code\code.qmd

# run command runs the stata code
run: do

# Stata code. Run with "make do". Last line removes the log file.
do: $(DOFILE)
    $(STATA) /e /q do $(DOFILE) > NUL 2>&1
    @cmd /c "if exist "*.log"  del /q /f "*.log"

# R code. Run with "make r"
r:
    $(RSCRIPT) --vanilla --quiet "$(RFILE)" >NUL 2>&1

# Quarto code. Run with "make quarto." Immediately removes quarto.html after running and moves code.pdf to 3_output
quarto:
    $(RSCRIPT) --vanilla --quiet "$(RFILE)" >NUL 2>&1
    $(QUARTO) render "$(QMD)" --execute --embed-resources
    @cmd /c "if exist "1_code\code.pdf" move "1_code\code.pdf" "3_output\code.pdf"

# housekeeping to remove stuck logfiles, quarto files, and clear process and output folders. Run with "make clean"
clean:
    @cmd /c "if exist "*.log"  del /q /f "*.log"
    @cmd /c "if exist "*.smcl"  del /q /f "*.smcl"
    @cmd /c "if exist "1_code\code.html" del "1_code\code.html""

    @cmd /c powershell -NoLogo -NoProfile -Command "If(!(Test-Path '2_process')){New-Item -ItemType Directory -Path '
    @cmd /c powershell -NoLogo -NoProfile -Command "If(!(Test-Path '3_output')){New-Item -ItemType Directory -Path '3

# Phony targets. Type "make r," "make do," or "make clean"
.PHONY: run do r quarto clean
```

Summary and Wrap-up

- Across all fields of science, reproducibility has been under threat (Open Science Collaboration 2015; Baker 2016; Camerer et al. 2016; Hail, Lang, and Leuz 2020; Fišar et al. 2023)
- Good data management is vital. You, as a WHU doctoral student, can help by:
  1. Maintaining a good directory structure
  2. Using version control
  3. Integrating multiple languages
- Ideally, we also need:
  - Research material sharing (data, instruments, and code)
  - Pre-registration
- Many journals and institutions offer ways to pre-register and collect research materials in one place.
  - You can sync your repositories as well!
- My recommendation is to use the Open Science Foundation

Thank you!

Dr. Victor van Pelt
Assistant Professor of Accounting
Finance and Accounting Group
WHU – Otto Beisheim School of Management

Campus Vallendar, Burgplatz 2, 56179 Vallendar, Germany
Tel.: +49 (0)261 6509 483
Victor.vanPelt@whu.edu
https://www.victorvanpelt.com

References

Baker, Monya. 2016. “Reproducibility Crisis.” *Nature* 533 (26): 353–66. https://doi.org/10.1038/533452a.

Camerer, Colin F, Anna Dreber, Eskil Forsell, Teck-Hua Ho, Jürgen Huber, Magnus Johannesson, Michael Kirchler, et al. 2016. “Evaluating Replicability of Laboratory Experiments in Economics.” *Science* 351 (6280): 1433–36. https://doi.org/doi.org/10.1126/science.aaf0918.

Fišar, Miloš, Ben Greiner, Christoph Huber, Elena Katok, and Ali I. and Ozkes. 2023. “Reproducibility in Management Science.” *Management Science*. https://doi.org/10.1287/mnsc.2023.03556.

Hail, Luzi, Mark Lang, and Christian Leuz. 2020. “Reproducibility in Accounting Research: Views of the Research Community.” *Journal of Accounting Research* 58 (2): 519–43. https://doi.org/10.1111/1475-679X.12305.

Open Science Collaboration. 2015. “Estimating the Reproducibility of Psychological Science.” *Science* 349 (6251): aac4716. https://doi.org/10.1126/science.aac4716.