

# “Works for me”

## DevOps for Reproducibility in the Social Sciences

David Seseke, January 30, 2025

### I ABOUT ME

- Used to work in Mechanical Engineering
- Transferred to programming and DevOps
- Somehow ended up in Sociology
- Works with social network analysis, technical sociology of the digital and reproducibility

### 2 OVERVIEW

1. Motivation
2. Introduction to DevOps
3. Entry level variant
4. Advance variant
5. Conclusion

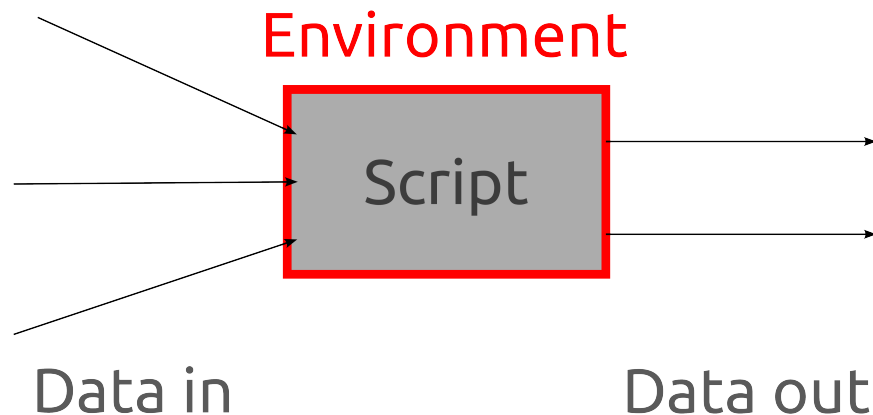
### 3 MOTIVATION

- Well-known replication problems in traditional quantitative social sciences (Open Science Collaboration 2015)
- Computational modeling could circumvent theory crisis (Oberauer and Lewandowsky 2019; Guest and Martin 2021)
- A general lack of computational sociology methodology (Hox 2017)
- Personal experiences in research and teaching of network analysis

#### 3.1 COMPUTATIONAL IRREPRODUCIBILITY IN SOCIAL SCIENCES

- Known examples of irreproducibility (Herndon, Ash, and Pollin 2014; Breznau et al. 2022)
- Transparency and reproducibility research practices not well established (Hardwicke et al. 2020; Freese, Rauf, and Voelkel 2022; Gayle and Connelly 2022)
- Complex toolchains worsen the dependency on hardware/software environments (Acher et al. 2024)
- Random but also chaotic deterministic system models susceptible to floating type rounding errors (Massonnet et al. 2020)
- Machine learning models could create new threats to reproducibility like “data leakage” (Gibney 2022)

## 4 UNDERSTANDING RANDOMNESS IN THE PROCESS



The question at hand is “What leads to irreproducibility?” Several factors lead to this, for example oversight or the pressure of completing a paper. The environment, highlighted in red, plays a critical role in enabling our script to operate effectively.

This script imports data and produces various outputs, such as images and tables. While we aim to manage the inputs to the script, achieving precise control is challenging. These inputs can encompass data and a range of dependent variables.

Other complexities such as the use of diverse architectures like ARM, x64, and potentially RISC-V exist. Moreover, high-performance computing (HPC) systems often feature intricate hardware configurations that can further complicate reproducibility.

## 5 IS (TECHNICAL) REPRODUCIBILITY A NEW PROBLEM?

No, it is really not. This is a longterm problem in the field of Information Technology for which different approaches have been tried again and again.

The professionals focused on this problem are commonly referred to as Development and Operations, or DevOps for short.

This profession is understood as a bridge between developers and system administrators. Developers are responsible for coding, which includes writing new code and reviewing existing code, while system administrators primarily manage the physical infrastructure and its interaction patterns.

## 6 (RELEVANT) CONCEPTS OF DEVOPS (MEYER 2018: 240)

- Development and Operations: The link between developers and system administrators
- Agile development (CI/CD)
- Testing
- Provisioning of environments

■ So what can we transfer from this field of work to our use case in the social sciences?

## 7 DISCLAIMER

I focus on Python and R as languages.

Stata, SPSS, Matlab and friends are not covered as I consider reproducibility a problem of the developers of those programs.

## 8 REQUIREMENTS

- Is reproducibility a required condition?
- Do you have a complex toolchain?
- Do you expect your research to be reproducible longer than 5 years?
- Do you live in the 🌍?

If you answered yes to several questions, there might be more complex tools required than what is shown in the next few slides. Also consider finding a balance between your current skill level and your expectations of reproducibility. Reflect on the complexity of your toolchain as you do this.

## 9 ENTRY LEVEL VARIANT

- Documentation
- Local environments
- Tests
- Version Control

In the next few slides I will show a couple of beginner-friendly approaches to improving reproducibility which does not require an in-depth knowledge.

### 9.1 DOCUMENTATION

- Think about what you have done to set up your environment.
- Document your setup along the way.
- Use preferably a markup language for documentation.
- Try to reproduce your documentation (and only that) with a clean setup.
- Ask a colleague with a different system if they can try to run your setup.

### 9.2 LOCAL ENVIRONMENTS

- If possible
- Python: `python -m venv .venv`
- R (not the internal ones): They do not exist.

### 9.3 TESTS

- Write a script which tests for basic functionality.
- Check for working imports.
- Reduce your data set.
- Reduce your calculation time.

## 9.4 VERSION CONTROL

- By far the hardest one.
- Learning it is worth for several reasons:
  - Participating in the development process
  - Not worrying about saves anymore
  - Social networking for nerds
  - Version controlling your PhD
- Use a GUI.

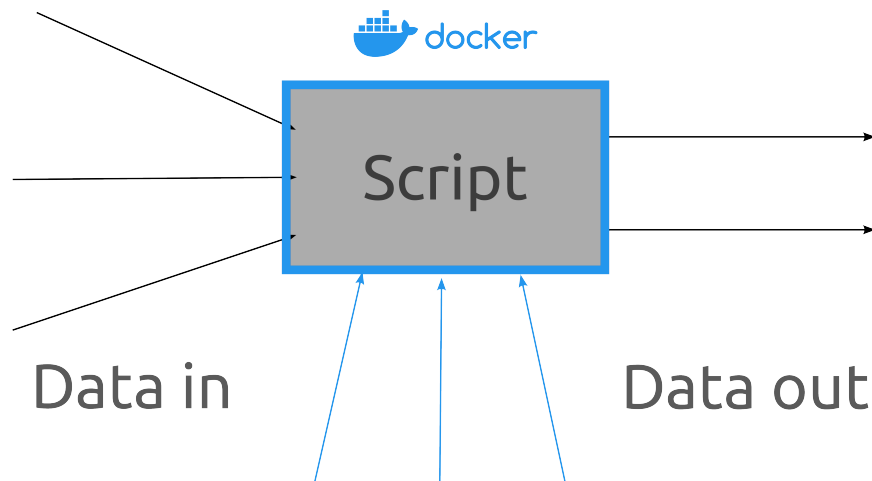
## 10 ADVANCED LEVEL VARIANT

- This requires command line knowledge.
- This is often badly documented (but it's getting better).
- Overview for containers (Moreau, Wiebels, and Boettiger 2023)
- *But*: Containers can be non-deterministic

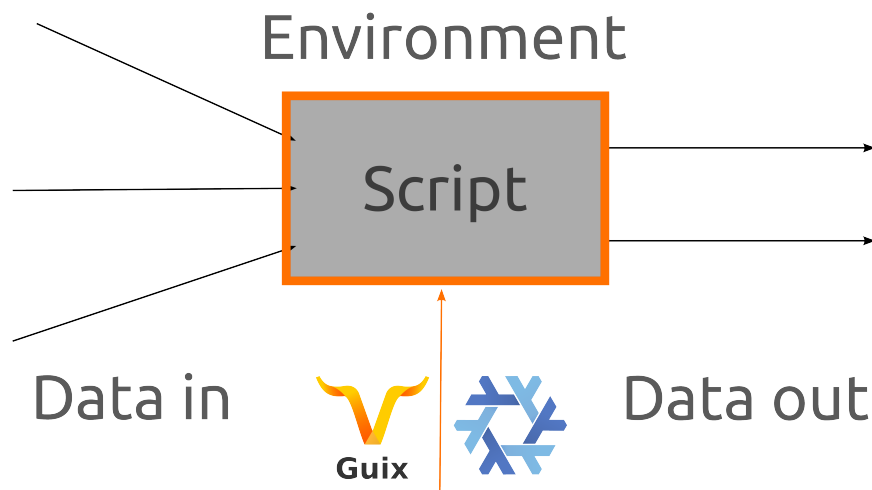
The non-deterministic nature of containers may require some clarification. Containers are constructed in layers, where each layer represents a specific component of the overall container. This is evident when you define your Docker container with a command like at the top. Beneath sits another layer which defines the python environment which you can utilize later. Depending on the underlying layer certain parts in that setup might not be fully deterministic.

At this stage, you haven't even started creating a Dockerfile, yet there is already an element of randomness embedded into the container.

### 10.1 THE PROBLEM WITH DOCKER



## 10.2 PURELY FUNCTIONAL PACKAGE MANAGEMENT

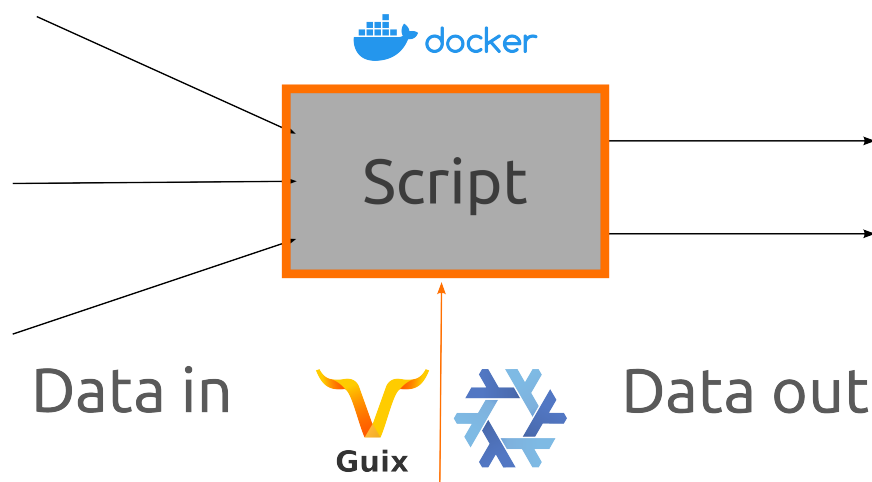


Purely functional package management shifts its focus from building containers to making the dependency tree entirely deterministic. To achieve this, all dependencies must be packaged in a specific manner.

This is a significant and ongoing undertaking.

## 10.3 LIVE DEMO

## 10.4 PURELY FUNCTIONAL DOCKER



## II WHAT IS POSSIBLE VS. WHAT IS FEASIBLE?

DevOps established themselves due to several factors: The primary reason was the increasing com-

plexity in toolchains in programming, which required an integrated approach to software development. Additionally, the demand for scalability became an important consideration in modern development companies and startups.

This leads us to the question: how much of this burden can we place on social scientists when the initial motivation of DevOps was the overwhelming complexity for professional developers?

## I2 CONCLUSION

- Purely functional package management: Still immature
- KISS: Keep it simple, stupid!
- Use tooling that compliments the complexity of your research
- Collaborate with people that live in the 🐉 from the beginning

I still view tools like Nix as being quite unstable. This instability also creates numerous tools within the ecosystem which are often not long-lived.. However, it is important keep an eye on developments in this area.

I am also convinced that there is no one-size-fits-all solution to complex systems. Complex tools necessitate sophisticated tooling if we are to achieve reproducibility and reliability in our processes.

## I3 IS THERE A NEED FOR SOCIAL SCIENCE DEV OPS?

## I4 LINK TO THE SHOWN EXAMPLES (AND SOME MORE)



<https://gitlab.rrz.uni-hamburg.de/reproducibilitystuttgart>

## I5 HELPFUL LINKS

- How to publish software: <https://media.ccc.de/v/38c3-basics-of-software-publication>
- Simple Nix-Installer: <https://lix.systems/>
- Using Nix with PEP-621 pyproject.toml files (unstable): <https://github.com/pyproject-nix/pyproject-nix>
- <https://wiki.nixos.org/wiki/Python>

- <https://wiki.nixos.org/wiki/R>
- Package management for R with extensive documentation: <https://docs.ropensci.org/rix/> (Broken on Mac)

## REFERENCES

- Acher, Mathieu et al. (2024). “Embracing Deep Variability For Reproducibility and Replicability.” In: *Proceedings of the 2nd ACM Conference on Reproducibility and Replicability*. ACM REP ’24. New York, NY, USA: Association for Computing Machinery, pp. 30–35. DOI: 10.1145/3641525.3663621.
- Breznau, Nate et al. (2022). “Observing Many Researchers Using the Same Data and Hypothesis Reveals a Hidden Universe of Uncertainty.” In: *Proceedings of the National Academy of Sciences* 119.44, e2203150119. DOI: 10.1073/pnas.2203150119.
- Freese, Jeremy, Tamkinat Rauf, and Jan Gerrit Voelkel (2022). “Advances in Transparency and Reproducibility in the Social Sciences.” In: *Social Science Research* 107, p. 102770. DOI: 10.1016/j.ssresearch.2022.102770.
- Gayle, Vernon and Roxanne Connelly (2022). “The Stark Realities of Reproducible Statistically Orientated Sociological Research: Some Newer Rules of the Sociological Method.” In: *Methodological Innovations* 15.3, pp. 207–221. DOI: 10.1177/20597991221111681.
- Gibney, Elizabeth (2022). “Could Machine Learning Fuel a Reproducibility Crisis in Science?” In: *Nature* 608.7922, pp. 250–251. DOI: 10.1038/d41586-022-02035-w.
- Guest, Olivia and Andrea E. Martin (2021). “How Computational Modeling Can Force Theory Building in Psychological Science.” In: *Perspectives on Psychological Science* 16.4, pp. 789–802. DOI: 10.1177/1745691620970585.
- Hardwicke, Tom E. et al. (2020). “An Empirical Assessment of Transparency and Reproducibility-Related Research Practices in the Social Sciences (2014–2017).” In: *Royal Society Open Science* 7.2, p. 190806. DOI: 10.1098/rsos.190806.
- Herndon, T., M. Ash, and R. Pollin (2014). “Does High Public Debt Consistently Stifle Economic Growth? A Critique of Reinhart and Rogoff.” In: *Cambridge Journal of Economics* 38.2, pp. 257–279. DOI: 10.1093/cje/bet075.
- Hox, Joop J. (2017). “Computational Social Science Methodology, Anyone?” In: *Methodology* 13. Supplement 1, pp. 3–12. DOI: 10.1027/1614-2241/a000127.
- Massonnet, François et al. (2020). “Replicability of the EC-Earth3 Earth System Model under a Change in Computing Environment.” In: *Geoscientific Model Development* 13.3, pp. 1165–1178. DOI: 10.5194/gmd-13-1165-2020.
- Meyer, Albin (2018). *Softwareentwicklung: Ein Kompass Für Die Praxis*. 1st ed. Description based on publisher supplied metadata and other sources. Berlin/München/Boston: Walter de Gruyter GmbH. 1 p. ISBN: 978-3-11-057837-9.
- Moreau, David, Kristina Wiebels, and Carl Boettiger (2023). “Containers for Computational Reproducibility.” In: *Nature Reviews Methods Primers* 3.1, pp. 1–16. DOI: 10.1038/s43586-023-00236-9.
- Oberauer, Klaus and Stephan Lewandowsky (2019). “Addressing the Theory Crisis in Psychology.” In: *Psychonomic Bulletin & Review* 26.5, pp. 1596–1618. DOI: 10.3758/s13423-019-01645-2.
- Open Science Collaboration (2015). “Estimating the Reproducibility of Psychological Science.” In: *Science* 349.6251, aac4716. DOI: 10.1126/science.aac4716.