Reproducibility in an AI World

Lars Vilhuber

2024-12-14

A PDF version of this presentation is available here.

Introduction

AI introduces **challenges for reproducibility**.

Not unlike **difficulties** researchers face with

- any black-box systems
- existing commercial software
- external APIs of any kind

I will discuss

- algorithmic transparency
- data dependencies
- archiving machine learning models

Computational reproducibility

In this talk, we focus on **computational reproducibility**, though the ultimate goal remains *replicability*.

LLM vs. AI

I will distinguish **LLM** (large language models) from **AI** (artificial intelligence):

- LLM: models that are trained for a specific (possibly broad) purpose
- AI: **online** systems that use LLMs, such as GPT, Claude, etc.

My background

- Data Editor for journals of the American Economic Association
- 1700+ reproducibility packages
- but: ⚠️ almost none with LLM or AI!

Reproducibility may be hard but important

What does Claude say (edited)?

> What should a presentation on the reproducibility of AI-based research address?

Response

Claude What should a presentation on ...

LV What should a presentation on the reproducibility of AI-based research address?

A presentation on the reproducibility of AI-based research should comprehensively address several critical aspects:

1. Methodology Transparency
   - Detailed documentation of data sources, preprocessing steps, and data cleaning techniques
   - Complete description of model architecture, hyperparameters, and training procedures
   - Explicit explanation of any data augmentation or synthetic data generation methods
2. Computational Environment
   - Specification of hardware used (GPU/CPU types, memory, computational resources)
   - Exact software and library versions (Python, PyTorch, TensorFlow, etc.)
   - Containerization or environment replication strategies (Docker, Conda

Response (edited)

A presentation on the reproducibility of AI-based research should comprehensively address several critical aspects:

1. Methodology Transparency
2. Computational Environment
3. Data Considerations
4. Experimental Reproducibility
5. Code and Implementation
6. Ethical and Contextual Considerations
7. Validation Strategies
8. Reporting Challenges

By comprehensively addressing these areas, the presentation can provide a robust framework for understanding and potentially replicating AI-based research, ultimately contributing to the scientific integrity and advancement of the field.

Comparison

Claude:

1. Methodology Transparency
2. Computational Environment
3. Data Considerations
4. Experimental Reproducibility
5. Code and Implementation
6. Ethical and Contextual Considerations
7. Validation Strategies
8. Reporting Challenges

OpenAI:

1. Introduction to Reproducibility
2. Challenges in Reproducibility
3. Data Accessibility
4. Algorithm and Model Transparency
5. Documentation and Reporting Standards
6. Tools and Platforms for Reproducibility
7. Case Studies and Examples
8. Community and Collaboration
9. Ethical and Legal Considerations
10. Future Directions and

Targets

We want to check that

- [ ] the materials can be accessed by others within a reasonable timeframe
- [ ] whether the materials can be preserved is made clear
- [ ] the extent to which re-running the same code yields the same results

Tools and Standards

In Economics,

- Use the Template README (presentation)
- Describe data provenance and access conditions for all **raw data**
- Describe all data transformations starting with **raw data**
- Provide **all code**, including for data you cannot share

But first...

Let’s talk about data

Types of Data

- Data used for training
- Data used for analysis
- Data output by the algorithm

Questions for Data

- Where did the (training/analysis) data come from?
  - Can you share it?
  - Can others obtain access?
  - Is it still there?
- Where did you put the analysis data?
  - Can you share it?
  - If not, why not?
  - Can you preserve it?

Guidance in README

Data and Code Availability Statement

It contains information about the sources of data used in the replication package, in addition to or instead of such detailed description in the manuscript. This is sometimes referred to as a “Data Availability Statement,” or if it also describes where additional code might be obtained, “Data and Code Availability Statements” (DCAS). A DCAS goes beyond a typical data citation, as it describes additional information necessary for the obtention of the data. These may include required registrations, memberships, application procedures, monetary cost, or other qualifications, beyond a simple URL for download which is typically part of Data Citations.

Data Provenance

> Are **models** data or software? - will treat as **software** here.

Data Provenance

- Are the source data **preserved**?
  - Often large **text** archives
  - Format relevant: physical or electronic copy?

Good example

“Immigration Restrictions as Active Labor Market Policy: Evidence from the Mexican Bracero Exclusion, Replication files and raw data” (Michael Clemens)

- Hosted on **Harvard Dataverse** at https://dataverse.harvard.edu/dataverse/bracero
- Contains two datasets:
  - Clemens, Michael, 2017, “**Raw scanned PDFs of primary sources for workers, wages, and crops**”, https://doi.org/10.7910/DVN/DJHVHB, Harvard Dataverse, V1

Your analysis data

Probably requires

- substantial computing resources (time, cost, space)
- lesser storage resources

Good example

> “Immigration Restrictions as Active Labor Market Policy: Evidence from the Mexican Bracero Exclusion, Replication files and raw data” (Michael Clemens)

- Hosted on **Harvard Dataverse** at https://dataverse.harvard.edu/dataverse/bracero
- Contains two datasets:
  - Clemens, Michael, 2018, “Replication Data for: Immigration Restrictions as Active Labor Market Policy: Evidence from the Mexican Bracero Exclusion”, https://doi.org/10.7910/DVN/17M4ZP, Harvard Dataverse, V1

LLM-specific considerations

Generically,

`pre-trained LLM` ▶ `tuned LLM` ▶ `analysis data`

LLM-specific considerations

- `tuned LLM` = f(`raw data`,`pre-trained LLM`)
- `analysis data` = f(`tuned LLM`,`raw data`)

Both should be preserved

LLM-specific considerations

- size?
- where?

LLM-specific considerations

`tuned LLM`:

- can you release it? (privacy)
- does **Hugging Face** have a preservation policy? (*no*)
- license to apply to it?

LLM-specific considerations

`analysis data`:

- can be preserved as part of the replication package
- could be preserved elsewhere, if multi-purpose

Run it all again

The very first test is that your code must run, **beginning to end,** top to bottom, without error, and ideally without any user intervention. This should in principle (re)create all figures, tables, and numbers you include in your paper.

TL;DR

This is pretty much the most basic test of reproducibility.

> This has nothing to do with LLM/AI!

If you cannot run your code, you cannot reproduce your results, nor can anybody else. So just re-run the code.

Exceptions

Code runs for a very long time

What happens when some of these re-runs are very long? See later in this chapter for how to handle this.

Making the code run takes YOU a very long time

While the code, once set to run, can do so on its own, *you* might need to spend a lot of time getting all the various pieces to run.

This should be a warning sign:

If it takes you a long time to get it to run, or to manually reproduce the results, it **might take others even longer**.[3]

Furthermore, it may suggest that you **haven’t been able to re-run** your own code very often, which can be indicate **fragility** or even **lack of reproducibility**.

Takeaways

- [x] your code runs without problem, after all the debugging.
- [ ] your code runs without manual intervention, and with low effort
- [ ] it actually produces all the outputs
- [ ] your code generates a log file that you can inspect, and that you could share with others.
- [ ] it will run on somebody else's computer

Why is this not enough?

Does your code run without manual intervention?

Automation and robustness checks, as well as efficiency.

Can you provide evidence that you ran it?

Generating a log file means that you can inspect it, and you can share it with others. Also helps in debugging, for you and others.

Will it run on somebody else's computer?

Running it again does not help:

- because it does not guarantee that somebody else has all the software (including packages!)
- because it does not guarantee that all the directories for input or output are there
- because many intermediate files might be present that are not in the replication package
- because you might have run things out of sequence, or relied on previously generated files in ways that won't work for others
- because some outputs might be present from test runs, but actually fail in this run

Hands-off running: Creating a controller script

- Your code must run, **beginning to end**, top to bottom, without error, and **without any user intervention**.
- This should in principle (re)create all **figures**, **tables**, and **in-text numbers** you include in your paper.

Seem trivial?

> Out of **8280** replication packages in ~20 top econ journals, only **2594** (**31.33%**) had a main/controller script.[4]

TL;DR

- Create a “main” file that runs all the other files in the correct order.
- Run this file, without user intervention.
- It should run without error.

Creating a main or master script

In order to be able to enable “hands-off running”, the **main (controller) script is key**.

Example 1: Querying Claude.ai

- for the first example, I was **lazy** - I typed the prompt into the Claude.ai website.
- Can you repeat it?
- What if I have to repeat it 100 times, with slight variations?

R

Set the root directory (using `here()` or `rprojroot()`).

```
# main.R
## Set the root directory
# If you are using Rproj files or git
rootdir <- here::here()
# or if not
# rootdir <- getwd()
## Run the data preparation file
source(file.path(rootdir, "01_data_prep.R")
       echo = TRUE)
## Run the analysis file
source(file.path(rootdir, "02_analysis.R"),
       echo = TRUE)
## Run the table file
source(file.path(rootdir, "03_tables.R"), e
## Run the figure file
source(file.path(rootdir, "04_figures.R"),
## Run the appendix file
source(file.path(rootdir, "05_appendix.R"),
```

R

Call each of the component programs, using `source()`.

```r
# main.R
## Set the root directory
# If you are using Rproj files or git
rootdir <- here::here()
# or if not
# rootdir <- getwd()
## Run the data preparation file
source(file.path(rootdir, "01_data_prep.R")
       echo = TRUE)
## Run the analysis file
source(file.path(rootdir, "02_analysis.R"),
       echo = TRUE)
## Run the table file
source(file.path(rootdir, "03_tables.R"), e
## Run the figure file
source(file.path(rootdir, "04_figures.R"),
## Run the appendix file
source(file.path(rootdir, "05_appendix.R"),
```

Notes for R

The use of `echo=TRUE` is best, as it will show the code that is being run, and is thus more transparent to you and the future replicator.

Notes for Python

- There are many ways to do this in Python (as there are more in R)
- Defining **functions** separately, and then calling them in the **main** file.
- Constructing a **package** and calling that package.

Notes for Python

If using **procedural** Python code, might use a `bash` script:

```
# This is main.sh
python 01_data_prep.py
python 02_analysis.py
python 03_tables.py
python 04_figures.py
```

Caution

What you do should remain **transparent** to other users!

Caution

Writing a scientific paper is different than writing a useful function on the internet.

> You are not writing `mynumpy`, you are writing a paper.

... though there are grey areas there.

Takeaways

- [x] your code runs without problem, after all the debugging.
- [x] your code runs without manual intervention, and with low effort
- [ ] it actually produces all the outputs
- [ ] your code generates a log file that you can inspect, and that you could share with others.
- [ ] it will run on somebody else’s computer

An example

Korinek (2023)

Existing Replication Package

- Article (Korinek 2023b) (also NBER WP (Korinek 2024))
- Package at (Korinek 2023a)
- Additional materials at https://www.genaiforecon.org/index.html
- I vetted this package!

Created in 2023

- Python based
- README states

```
Ensure you have the necessary Python libraries installed:

  pip install openai pandas numpy


To execute the simplest example, run the script:

 python simple_example_chat1.py


The results will be displayed on the screen.
```

Ex-post critique

- Missing a `requirements.txt`
- No instructions to set an environment

> These are now systematically requested for replication packages!

Trying it out

I created a `requirements.txt` file

```
numpy==2.2.0
openai==1.57.4
pandas==2.2.3
openpyxl
```

(note: created using `pipreqs` Python package, plus hand-edit).

Trying it out (2)

Create environment

```
python3.11 -m venv venv-311
source venv-311/bin/activate
pip install -r requirements.txt
```

(note: running on Linux, openSUSE, Python 3.11.10)

Trying it out (3)

Get the API key

- Go to https://platform.OpenAI.com
- Go to `API keys` on the left side
- Verify phone number (a challenge while roaming!)
- `+ Create new secret key`
- Save the API key in file `.env`

Trying it out (4)

Run the script

```
python simple_example_chat1.py
```

Failure!

```
> python simple_example_chat1.py
Please enter your OpenAI API key:
  File "/path/korinek-2023/simple_example_chat1.py", line 24, in <module>
    openai.api_key = input("Please enter your OpenAI API key: ")
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

KeyboardInterrupt
```

Attempt to fix

- README speaks of environment variable

```
export OPENAI_API_KEY=sk-proj-lxxxxxxxxxxx
```

- Run again, same error!

Reasons

- Not scripted enough, requires manual intervention!
- Ignores `.env` file (one way of doing it)
- Ignores environment variable (another way), despite doing it in another script!

Quick fix

- I fixed the script to read from environment variable.

```python
openai.api_key = os.environ.get('OPENAI_API_KEY')

# If the API key isn't found in the environment variable, prompt the user f
if not openai.api_key:
    openai.api_key = input("Please enter your OpenAI API key: ")
```

> NEVER RECORD YOUR API KEY IN SCRIPTS!

IMPORTANT

- These are **standard Python** issues, *not AI issues*!
- But they are crucial for reproducibility!

Result

```
Traceback (most recent call last):
  File "/path/korinek-2023/simple_example_chat1.py", line 37, in <module>
    completion = openai.ChatCompletion.create(
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/path/korinek-2023/venv-311/lib64/python3.11/site-packages/openai/lib/_old_api.py", line 39, in __cal
    raise APIRemovedInV1(symbol=self._symbol)
openai.lib._old_api.APIRemovedInV1:

You tried to access openai.ChatCompletion, but this is no longer supported in openai>=1.0.0 - see the README

You can run `openai migrate` to automatically upgrade your codebase to use the 1.0.0 interface.

Alternatively, you can pin your installation to the old version, e.g. `pip install openai==0.28`

A detailed migration guide is available here: https://github.com/openai/openai-python/discussions/742
```

THIS IS A STANDARD PYTHON - API ISSUE!

- These are **standard API** issues, *not AI issues*!
- APIs change
- Libraries change
- Having **latest** is not always best.
- But they are crucial for reproducibility!

> Provide `requirements.txt` and pin versions!

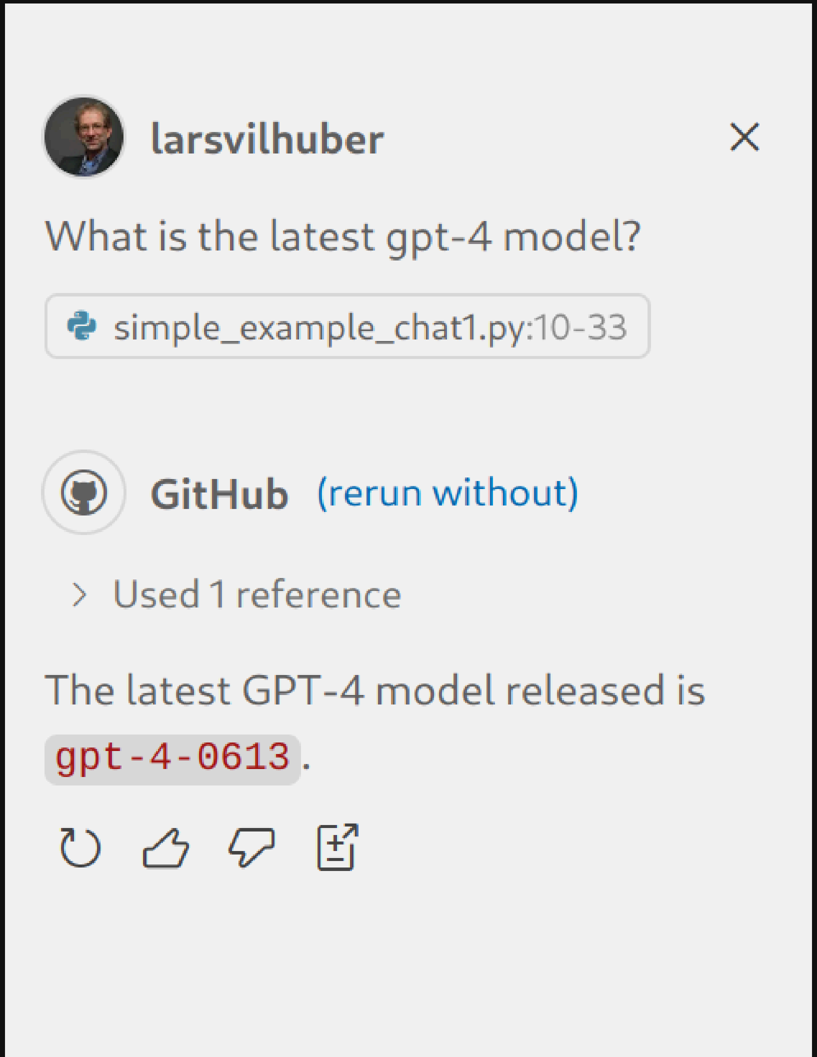
(We will talk later about API issues!)

Next attempt

- Fix `requirements.txt`, re-install

```
> python simple_example_chat1.py
Traceback (most recent call last):
  File "/path/korinek-2023/simple_example_chat1.py", line 37, in <module>
    completion = openai.ChatCompletion.create(
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
<snip>
  File "/path/korinek-2023/venv-311/lib64/python3.11/site-packages/openai/api_requestor.py", line 765, in _ir
    raise self.handle_error_response(
openai.error.InvalidRequestError: The model `gpt-4-0613` does not exist or you do not have access to it.
```

Fixing this



CoPilot response

(note: `github.copilot-chat` *0.23.1, updated 2024-12-14, 15:31:36)*

Turns out...

- I thought I had credits, I did not.
- It was the `you do not have access to it` part that was crucial!

Results

Original content

```
1. Job loss due to automation in lower-skille
2. AI-driven wealth concentration in tech-sav
3. Digital literacy gap leading to economic d
4. Lack of universal access to AI technology.
5. AI-driven bias in hiring and selection pro
6. Imbalance in job market due to AI speciali
7. Data privacy issues affecting vulnerable p
8. AI-driven services predominantly targeting
9. Algorithms exacerbating social inequality
10. Inclusive AI product development lacking.
11. Higher prices due to AI-enhanced products
12. AI-fueled gentrification in tech-centered
13. Anticompetitive practices bolstered by AI
14. Lack of labor rights for jobs displaced b
15. Educational imbalance due to AI-learning
16. AI in healthcare excluding lower socioeco
17. Disproportionate influence of AI in polit
18. Undervaluing of human skills in favor of
19. Biased AI systems perpetuating discrimina
20. AI reinforcing societal hierarchies via d
```

Content as of 2024-12-14:

```
1. Job displacement due to automation.
2. Wealth concentration in tech industries.
3. Increased surveillance disproportionately
4. Unequal access to AI technology.
5. AI-driven discrimination in hiring.
6. AI bias in credit scoring.
7. Inequality in AI education and training.
8. AI in healthcare favoring wealthier patien
9. AI-driven gentrification in cities.
10. AI in law enforcement targeting minoritie
11. AI in marketing exploiting vulnerable con
12. AI in politics manipulating voters.
13. AI in insurance favoring privileged group
14. AI in social media amplifying hate speech
15. AI in education favoring affluent student
16. AI in agriculture favoring large-scale fa
17. AI in transportation favoring urban areas
18. AI in retail favoring wealthier consumers
19. AI in entertainment creating cultural div
20. AI in research favoring developed countri
```

Next...

One of the most frequently asked questions...

“but I have confidential data…”

Equivalently

“but I have a big LLM model…”

So what happens when…

You cannot share a file

The file no longer exists on the internet

The code takes ages to run

How can you show that you actually ran the code?

Creating log files

In order to document that you have actually run your code, a **log file**, a transcript, or some other evidence, may be useful. It may even be required by certain journals.

TL;DR

- **Log files** are a way to document that you have run your code.
- In particular for code that runs for a very long time, or that uses data that cannot be shared, log files may be the only way to document basic reproducibility.

Overview

- Most statistical software has ways to keep a record that it has run, with the details of that run.
- Some make it easier than others.
- You may need to instruct your code to be “verbose”, or to “log” certain events.
- You may need to use a command-line option to the software to create a log file.

In almost all cased, the generated log files are simple text files, without any formatting, and can be read by any text editor (e.g., Visual Studio Code, Notepad++, etc.).

If not, ensure that they are (avoid *Stata SMCL* files, for example, or *iPython* output).

Creating log files explicitly

Generically: see *separate* tutorial.

Python

Create a wrapper that will capture the calls for any function

```
from datetime import datetime
def track_calls(func):
    def wrapper(*args, **kwargs):
        with open('function_log.txt', 'a') as f:
            timestamp = datetime.now().strftime('%
            f.write(f"[{timestamp}] Calling {func._
        result = func(*args, **kwargs)
        return result
    return wrapper

# Usage
@track_calls
def my_function(x, y,default="TRUE"):
    return x + y

my_function(1, 2,default="false")
```

Python

Activate the wrapper

```python
from datetime import datetime
def track_calls(func):
    def wrapper(*args, **kwargs):
        with open('function_log.txt', 'a') as f:
            timestamp = datetime.now().strftime('%
            f.write(f"[{timestamp}] Calling {func._
        result = func(*args, **kwargs)
        return result
    return wrapper

# Usage
@track_calls
def my_function(x, y,default="TRUE"):
    return x + y

my_function(1, 2,default="false")
```

Python

Ideally, capture the output

```
# Usage
@track_calls
def my_function(x, y,default="TRUE"):
    return x + y

my_function(1, 2,default="false")
# Output
# [2024-12-15 12:05:37] Calling my_function with ar
```

Creating log files automatically

An alternative (or complement) to creating log files explicitly is to use native functionality of the software to create them. This usually is triggered when using the **command line** to run the software, and thus may be considered an **advanced topic.** The examples below are for Linux/macOS, but similar functionality exists for Windows.

Python

In order to capture screen output in Python, on Unix-like system (Linux, macOS), the following can be run:

```
python main.py | tee main.log
```

which will create a log file with everything that would normally appear on the console using the `tee` command.

Takeaways

- [x] your code runs without problem, after all the debugging.
- [x] your code runs without manual intervention, and with low effort
- [x] it actually produces all the outputs
- [x] your code generates a log file that you can inspect, and that you could share with others.
- [ ] it will run on somebody else’s computer

Environments

TL;DR

- Search paths and environments are key concepts to create **portable**, **reproducible** code, by **isolating** each project from others.
- Methods exist in all (statistical) programming languages
- For more details, see other guidance

What software supports environments?

- R: `renv` package
- Python: `venv` or `virtualenv` module
- Julia: `Pkg` module

Understanding search paths

Generically, all “environments” simply modify where the specific **software searches** (the “search path”) for its components, and in particular any supplementary components (packages, libraries, etc.).[5]

Reproducing and documenting environments in Python

Python allows for pinpointing exact versions of packages in the *PyPi* repository. This is done by creating a `requirements.txt` file that lists all the packages that are needed to run your code. In principle, this file can

The issue

```
pip freeze
```

will output all the packages installed in your environment. These will include the packages you explicitly installed, but also the packages that were installed as dependencies. Some of those dependencies may be specific to your operating system or environment. In some cases, they contain packages that you needed to develop the code, but that are not needed to run it.

```
pip freeze > requirements.txt
```

will output all the packages installed in your

The solution

The solution is to create a minimal environment, and document it. This is done in two steps:

1. Identify the packages that are needed to run your code. There are packages that may help you with this, but in principle, you want to include everything you explicitly `import` in your code, and nothing else. This is the minimal environment.
2. Prune the `requirements.txt` file to only include the packages that are needed to run your code. This will be the file you provide to replicators to recreate your necessary environment, and let the package installers

Conclusion

AI and LLMs are not special

... when it comes to reproducibility

But difficulties are magnified

… compared to the **average** difficulty in economics papers

Solutions

- Be **reproducible** from the start
  - Use **environments**
  - Use **logging** as evidence, especially when **repetitions** are expensive

Solutions

- **Computational empathy**
  - Remember your own difficulties in getting this to work
  - Now put yourself in others' computer (shoes)
  - Be very clear about what is needed - you are **cutting edge**, others may not be!

Solutions

- Be **precise** about **versions**
  - of input data (including RAG/training/fine-tuning)
  - of software used (Python, libraries, but also *models* - do not use “latest”!)

Solutions

- Include **all code**
  - that includes *prompts*, intermediate *responses*
  - even if data are not included

Solutions

- Include all **metadata**
  - fix random seeds, where possible
  - **hyperparameters**, *temperature*, or whatever it is called
  - *prompts* could be considered **metadata**

Solutions

- Include **data** where possible
  - licenses
  - size
  - intermediate data where useful/time-consuming (but: license!)

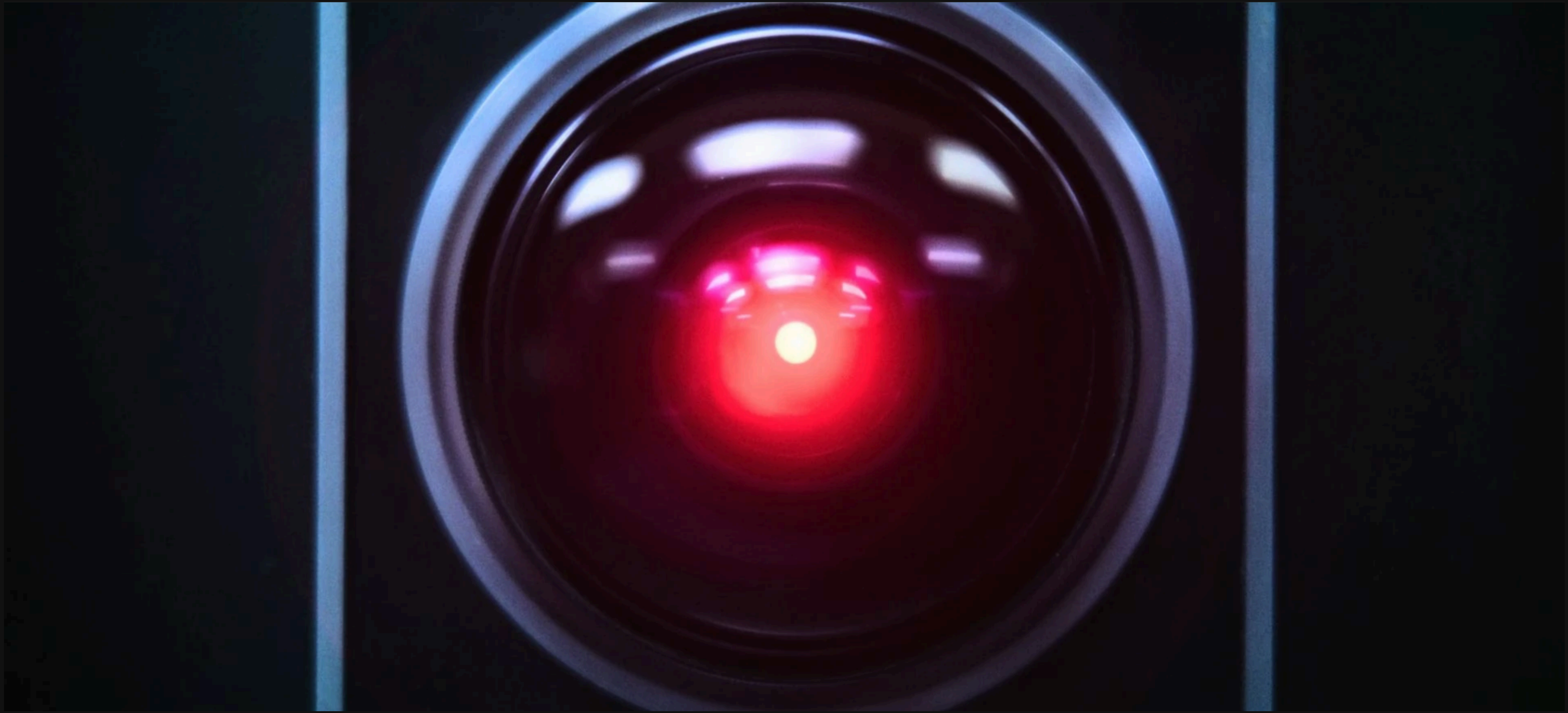
Solutions

- Consider how and where to **preserve**
  - industry repositories may be fine for **sharing**
  - academic repositories (**Zenodo**, **Dataverse**) handle **preservation**
  - Toolkit is still in its infancy for the preservation of large data ($>=$ 200GB)

Solutions

- Use existing **resources**
  - Template README (presentation) as guidance
  - Self-check your code (presentation)

Questions



This presentation

- Github
- Presentation (PDF)

Useful links

- GitHub
- Lars Vilhuber
- Presentation QR Code:

- Last Run on 2024-12-18

Appendix

What did the AI say?

What did Claude say?

6

A presentation on the reproducibility of AI-based research should address several key areas to ensure a comprehensive understanding of the topic. Here are some important points to consider:

1. **Introduction to Reproducibility**:
   - Define reproducibility in the context of AI research.
   - Explain why reproducibility is crucial for scientific integrity, validation, and progress.

2. **Challenges in Reproducibility**:
   - Discuss common challenges such as lack of access to data, proprietary algorithms, and insufficient documentation.
   - Highlight issues related to computational resources and environment dependencies.

3. **Data Accessibility**:
   - Emphasize the importance of open datasets and the challenges of data privacy and security.
   - Discuss strategies for sharing data, such as anonymization and data licensing.

4. **Algorithm and Model Transparency**:
   - Address the need for sharing model architectures, hyperparameters, and training procedures.
   - Discuss the role of open-source software and platforms in promoting transparency.

5. **Documentation and Reporting Standards**:
   - Highlight the importance of thorough documentation, including code comments, readme files, and research papers.
   - Discuss existing standards and guidelines, such as those from journals or conferences, for reporting AI research.

6. **Tools and Platforms for Reproducibility**:
   - Introduce tools and platforms that facilitate reproducibility, such as version control systems (e.g., Git), containerization (e.g., Docker), and workflow management systems.
   - Discuss the role of cloud computing in providing scalable and reproducible environments.

7. **Case Studies and Examples**:
   - Present examples of successful reproducibility efforts in AI research.
   - Discuss lessons learned from reproducibility failures and how they were addressed.

8. **Community and Collaboration**:
   - Highlight the role of the research community in fostering a culture of reproducibility.
   - Discuss collaborative efforts, such as reproducibility challenges and workshops.

9. **Ethical and Legal Considerations**:
   - Address ethical concerns related to data sharing and model transparency.
   - Discuss legal implications, such as intellectual property rights and compliance with regulations.

10. **Future Directions and Recommendations**:
    - Suggest ways to improve reproducibility in AI research, such as developing better standards, incentives, and educational resources.
    - Discuss the potential impact of emerging technologies and methodologies on reproducibility.

11. **Conclusion**:
    - Summarize the key points discussed.
    - Reinforce the importance of reproducibility for advancing AI research and its applications.

By covering these areas, the presentation can provide a well-rounded view of the challenges and solutions related to the reproducibility of AI-based research.

What did OpenAI say?

7

A presentation on the reproducibility of AI-based research should address several key areas to ensure a comprehensive understanding of the topic. Here are some important points to consider:

1. **Introduction to Reproducibility**:
   - Define reproducibility in the context of AI research.
   - Explain why reproducibility is crucial for scientific integrity, validation, and progress.

2. **Challenges in Reproducibility**:
   - Discuss common challenges such as lack of access to data, proprietary algorithms, and insufficient documentation.
   - Highlight issues related to computational resources and environment dependencies.

3. **Data Accessibility**:
   - Emphasize the importance of open datasets and the challenges of data privacy and security.
   - Discuss strategies for sharing data, such as anonymization and data licensing.

4. **Algorithm and Model Transparency**:
   - Address the need for sharing model architectures, hyperparameters, and training procedures.
   - Discuss the role of open-source software and platforms in promoting transparency.

5. **Documentation and Reporting Standards**:
   - Highlight the importance of thorough documentation, including code comments, readme files, and research papers.
   - Discuss existing standards and guidelines, such as those from journals or conferences, for reporting AI research.

6. **Tools and Platforms for Reproducibility**:
   - Introduce tools and platforms that facilitate reproducibility, such as version control systems (e.g., Git), containerization (e.g., Docker), and workflow management systems.
   - Discuss the role of cloud computing in providing scalable and reproducible environments.

7. **Case Studies and Examples**:
   - Present examples of successful reproducibility efforts in AI research.
   - Discuss lessons learned from reproducibility failures and how they were addressed.

8. **Community and Collaboration**:
   - Highlight the role of the research community in fostering a culture of reproducibility.
   - Discuss collaborative efforts, such as reproducibility challenges and workshops.

9. **Ethical and Legal Considerations**:
   - Address ethical concerns related to data sharing and model transparency.
   - Discuss legal implications, such as intellectual property rights and compliance with regulations.

10. **Future Directions and Recommendations**:
    - Suggest ways to improve reproducibility in AI research, such as developing better standards, incentives, and educational resources.
    - Discuss the potential impact of emerging technologies and methodologies on reproducibility.

11. **Conclusion**:
    - Summarize the key points discussed.
    - Reinforce the importance of reproducibility for advancing AI research and its applications.

By covering these areas, the presentation can provide a well-rounded view of the challenges and solutions related to the reproducibility of AI-based research.

What did Gemini say?

8

A presentation on the reproducibility of AI-based research should address a multifaceted issue, covering both the challenges and potential solutions. Here's a breakdown of key areas to consider:

1. Defining Reproducibility in the Context of AI:

* **What does reproducibility mean in AI?** Differentiate between:
    * **Replication:** Re-running the same code on the same data and getting the same results.
    * **Reproduction:** Re-implementing the method from scratch based on the paper and getting similar results.
    * **Robustness:** Evaluating the model's performance on different datasets or under different conditions.
* **Why is reproducibility important?** Highlight the benefits:
    * **Scientific validity:** Ensuring the reliability and trustworthiness of research findings.
    * **Progress and innovation:** Building upon existing work and accelerating advancements.
    * **Transparency and accountability:** Allowing for scrutiny and identification of potential biases or errors.

2. Challenges to Reproducibility in AI:

* **Code and Implementation:**
    * **Lack of code availability:** Papers often lack publicly available code or provide incomplete/unusable code.
    * **Poor code quality:** Unstructured, undocumented, or hard-to-understand code.
    * **Dependency issues:** Incompatible library versions, operating systems, or hardware.
    * **Hidden implementation details:** Crucial parameters or preprocessing steps not explicitly mentioned in the paper.
* **Data:**
    * **Data unavailability:** Datasets are often proprietary, sensitive, or difficult to access.
    * **Data preprocessing:** Inconsistent or undocumented data cleaning, transformation, or augmentation techniques.

3. Solutions and Best Practices for Improving Reproducibility:

* **Code Sharing and Version Control:**
    * **Public repositories:** Using platforms like GitHub, GitLab, or Bitbucket to share code.
    * **Version control:** Employing Git to track changes and manage different versions of the code.
    * **Clear documentation:** Providing comprehensive README files and comments within the code.
* **Data Management:**
    * **Public datasets:** Utilizing publicly available datasets whenever possible.
    * **Data sharing platforms:** Using platforms like Kaggle or Zenodo to share datasets.
    * **Data provenance:** Documenting the origin, processing, and versioning of the data.

4. Call to Action:

* **Encourage researchers to adopt best practices.**
* **Promote the use of open-source tools and platforms.**
* **Advocate for changes in the academic system to incentivize reproducible research.**
* **Raise awareness about the importance of reproducibility in AI.**

Presentation Structure Suggestions:

* **Start with a compelling example** of a reproducibility failure in AI research.
* **Clearly define the problem** and its implications.
* **Present the challenges** in a structured and understandable way.
* **Offer practical solutions** and best practices.
* **Conclude with a call to action** and a positive outlook for the future of reproducible AI research.

Visual Aids:

* **Use clear and concise slides.**
* **Include diagrams and illustrations to explain complex concepts.**
* **Show examples of good and bad code documentation.**
* **Use charts and graphs to visualize data and results.**

By addressing these key areas, your presentation will provide a comprehensive overview of the challenges and opportunities surrounding the reproducibility of AI-based research, ultimately contributing to a more robust and trustworthy field. Remember to tailor your presentation to your specific audience and their level of understanding.

References

Korinek, Anton. 2023a. "Data and Code for: Generative AI for Economic Research: Use Cases and Implications for Economists." Inter-university Consortium for Political and Social Research (ICPSR). https://doi.org/10.3886/E194623V1.

———. 2023b. "Generative AI for Economic Research: Use Cases and Implications for Economists." *Journal of Economic Literature* 61 (4): 1281–1317. https://doi.org/10.1257/jel.20231736.

———. 2024. "Generative AI for Economic Research: LLMs Learn to Collaborate and Reason." Working {{Paper}} 33198. National Bureau of Economic Research. https://doi.org/10.3386/w33198.

Footnotes

1.

Claude, queried on 2024-12-16, see `lars_query_claude.py` and `lars-prompt1.txt`

2.

Claude, queried on 2024-12-16, see `lars_query_claude.py` and `lars-prompt1.txt`

3. Source: Red Warning PNG Clipart, CC-BY.

4.

Results computed on Nov 26, 2023 based on a scan of replication packages conducted by Sebastian Kranz. 2023. "Economic Articles with Data". https://ejd.econ.mathematik.uni-ulm.de/, searching for the words `main`, `master`, `makefile`, `dockerfile`, `apptainer`, `singularity` in any of the program files in those replication packages. Code not yet integrated into this presentation.

5.