01. Getting started — Part 2

DSIER [/dɪˈzaɪər/] — Summer 2022

Julian Hinz

Bielefeld University

GOOD RESEARCH PRACTICE

Session Roadmap

- Reproducibility
- Git and Github
- Docker
- ChatGPT & Co.
- Setting up your machines

REPRODUCIBILITY

Reproducibility

- Mostly for your future self!
- but of course also: Science.

“Trying to replicate the estimates from an early draft of a paper, we discover that the code that produced the estimates no longer works because it calls files that have since been moved.

Now: No longer works.”

“Between regressions number of observations falling. After much sleuthing, we find that many observations were dropped in a merge because they had missing values for the county identifier we were merging on. When we correct the mistake and include the dropped observations, the results change dramatically.”

“Me and my coauthor write code that refers to a common set of data files stored on a shared folder. Our work is constantly interrupted because changes one of us makes to the data files causes the others’ code to break.”

8 building blocks of reproducibility

Code and Data in the Social Sciences (Gentzkow and Shapiro):

1. Automation
2. Version Control
3. Directories
4. Keys
5. Abstraction
6. Documentation
7. Management
8. Code Style

1 — Automation

Automation

1. Automate everything that can be automated.
2. Write a single script that executes all code from beginning to end.

→ Use a “master” file or, even better, use `make`

1 — Automation

Automation

1. Automate everything that can be automated.
2. Write a single script that executes all code from beginning to end.

$\rightarrow$ Use a “master” file or, even better, use `make`

2 — Version Control

Version Control

1. Store code and data under version control.
2. Run the whole directory before checking it back in.

→ Use Git

2 — Version Control

Version Control

1. Store code and data under version control.
2. Run the whole directory before checking it back in.

$\rightarrow$ Use Git

3 — Directories

Directories

1. Separate directories by function.
2. Separate files into inputs and outputs.
3. Make directories portable.

→ Use `code`, `input`, `output` and `temp` folders

3 — Directories

Directories

1. Separate directories by function.
2. Separate files into inputs and outputs.
3. Make directories portable.

→ Use `code`, `input`, `output` and `temp` folders

4 — Keys

Keys

1. Store cleaned data in tables with unique, non-missing keys.
2. Keep data normalized as far into your code pipeline as you can.

5 — Abstraction

Abstraction

1. Abstract to eliminate redundancy.
2. Abstract to improve clarity.
3. Otherwise, don’t abstract.

6 — Documentation

Documentation

1. Don’t write documentation you will not maintain.
2. Code should be self-documenting.

7 — Management

Management

1. Manage tasks with a task management system.
2. E-mail is not a task management system.

8 — Code Style

Code Style

1. Keep it short and purposeful.
2. Use descriptive names.
3. Be consistent.
4. Profile slow code relentlessly.
5. Store “too much” output from slow code.
6. (Stick to style guide)

Quick aside: Style guids

- Google: https://google.github.io/styleguide/Rguide.html
- tidyverse: https://style.tidyverse.org

GIT AND GITHUB

Git

- Git is a distributed version control system
  $\rightarrow$ “Dropbox and the ”Track changes” feature in MS Word have a baby: Git”
- Optimized for code (not data, actually)

GitHub

- Online hosting platform that provides services built on top of the Git system
  $\rightarrow$ Similar: Bitbucket and GitLab
- Makes Git *a lot* more user friendly
- Seamless integration into lots of other software: VSCode (and RStudio)

DOCKER

“Container” technology

- Docker containers are the software equivalent
  - → physical goods <-> software
  - → transport modes <-> operating systems

“Container” technology

- Standardized shape and form
- “If it runs on your machine, it will run on my machine.”
- Allows to always run code from a pristine, predictable state

How it works

- Stripped-down version of an operating system

  $\rightarrow$ Usually a Linux distro like Ubuntu

- Installs all of the programs and dependencies that are needed to run the code

  $\rightarrow$ + add any extra configurations that are needed/wanted

- Package everything up as a tarball (i.e. compressed file)

  $\rightarrow$ Containers are like mini, portable OS that contain everything needed to run some piece of software (but nothing more!)

Docker terminology

- Dockerfile: “The sheet music”

  → list of layers and instructions for building a Docker image

- Image: “The MP3 file”

  → tarball packages everything needed

- Container: “Song playing on my phone”

  → running instance of an image

Minimal working example

```
$ docker run --rm -it rocker/r-base
```

docker `run flags`

- `--rm` automatically removes the container once it exits (i.e. clean up)
- `-it` Launch with interactive (i) shell/terminal (t)

A bit more sophisticated working example

```
$ docker run -d -p 8787:8787 -e PASSWORD=verystrong rocker/tidyverse
```

docker `run flags`

- `-d` detach (i.e. run as background process)
- `-p 8787:8787` share a port with the host computer's browser
- `-e PASSWORD=pswd123` set password for logging on to RStudio Server
- `rocker/tidyverse:4.0.0` prepared `tidyverse` image built on top of R 4.0.0

CHATGPT & CO.

Introduction to LLMs: ChatGPT and GitHub Copilot

- Large Language Models (LLMs) are AI systems trained on massive text data
- ChatGPT: Powerful text generator, can be used for academic work
- GitHub Copilot: AI-based code generation, supports development tasks
- LLMs can improve productivity, but require caution in use
- Applications in data science, economic research, and academic writing

Benefits of LLMs in Academic Work

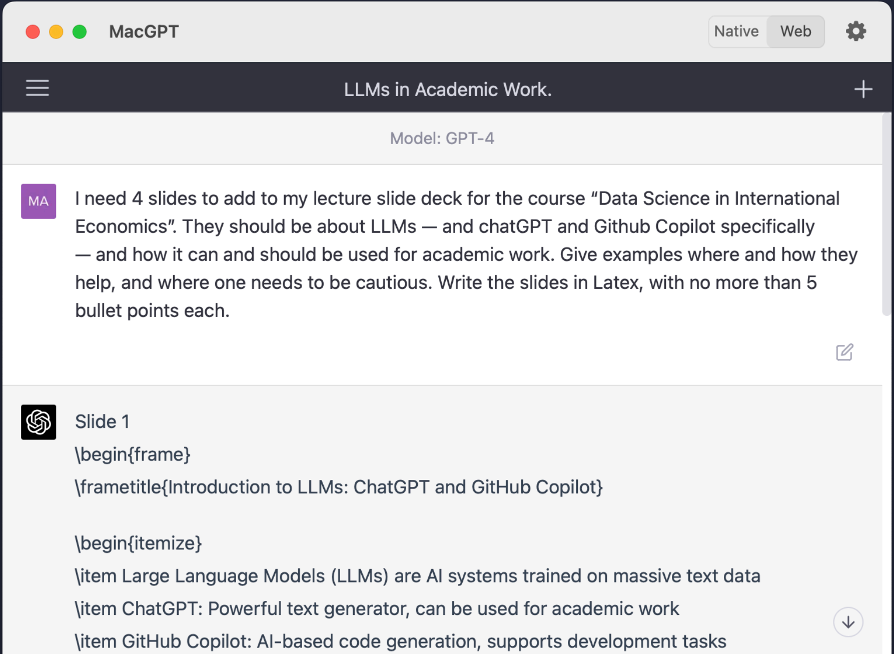
- Rapid literature review and summarization
- Guided brainstorming and idea exploration
- Assisting with code development and debugging
- Drafting and editing academic papers or presentations
- Automating repetitive tasks, enabling focus on high-level analysis

Cautions When Using LLMs

- Potential for biased or incorrect information
- Lack of understanding or context for complex topics
- Over-reliance may hinder critical thinking and originality
- Ethical concerns regarding data privacy and usage
- Proper citation and attribution to avoid plagiarism

Best Practices for LLMs in Academic Work

- Cross-validate information with trusted sources
- Use LLMs as a complementary tool, not a substitute for expertise
- Engage in regular peer review for insights and feedback
- Be transparent about LLM usage in research and publications
- Stay updated on LLM advancements and ethical guidelines



LLMs in Academic Work.

Model: GPT-4

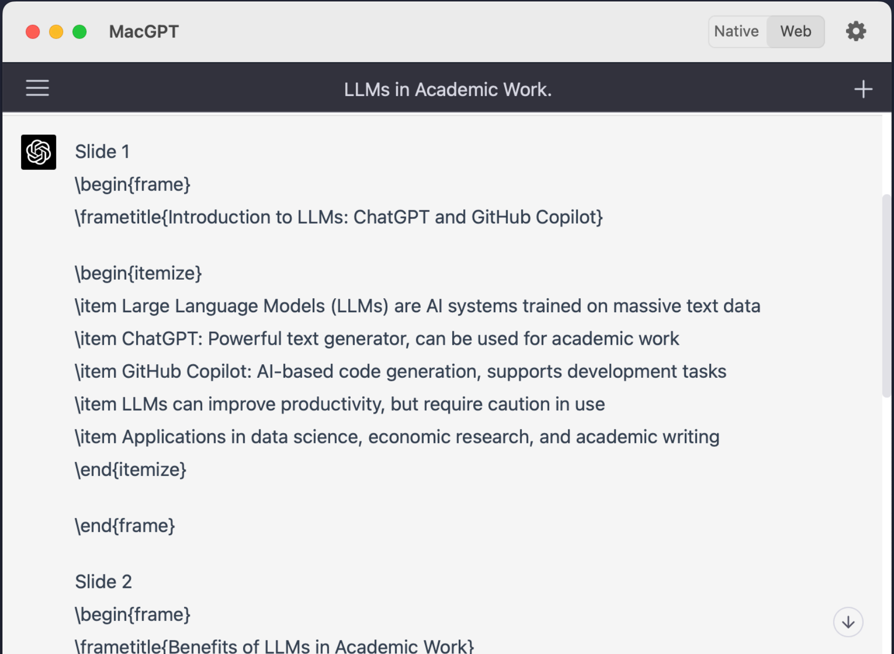
MA

I need 4 slides to add to my lecture slide deck for the course “Data Science in International Economics”. They should be about LLMs — and chatGPT and Github Copilot specifically — and how it can and should be used for academic work. Give examples where and how they help, and where one needs to be cautious. Write the slides in Latex, with no more than 5 bullet points each.

Slide 1

```
\begin{frame}
\frametitle{Introduction to LLMs: ChatGPT and GitHub Copilot}

\begin{itemize}
\item Large Language Models (LLMs) are AI systems trained on massive text data
\item ChatGPT: Powerful text generator, can be used for academic work
\item GitHub Copilot: AI-based code generation, supports development tasks
```



Slide 1

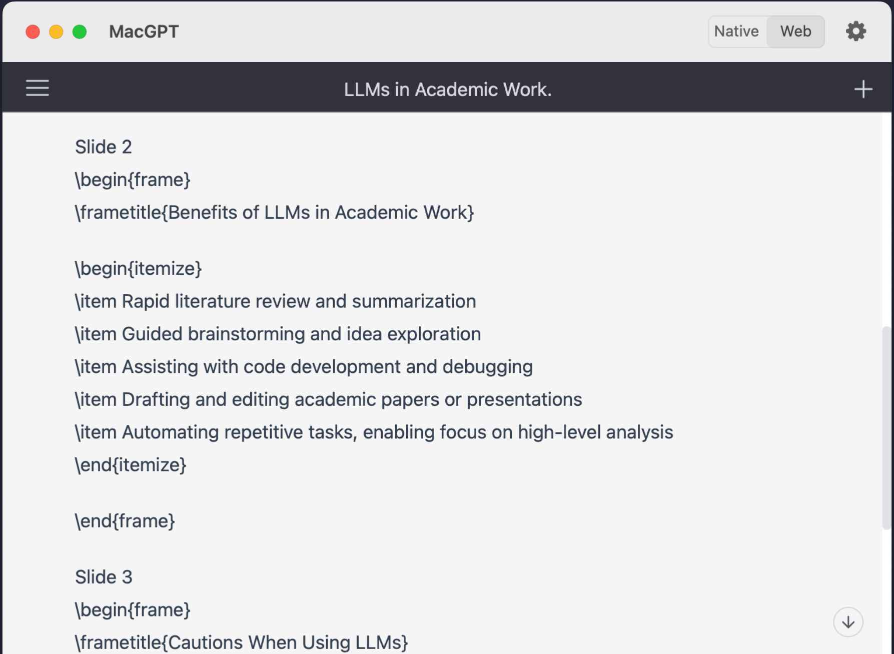
```
\begin{frame}
\frametitle{Introduction to LLMs: ChatGPT and GitHub Copilot}

\begin{itemize}
\item Large Language Models (LLMs) are AI systems trained on massive text data
\item ChatGPT: Powerful text generator, can be used for academic work
\item GitHub Copilot: AI-based code generation, supports development tasks
\item LLMs can improve productivity, but require caution in use
\item Applications in data science, economic research, and academic writing
\end{itemize}

\end{frame}
```

Slide 2

```
\begin{frame}
\frametitle{Benefits of LLMs in Academic Work}
```



Slide 2

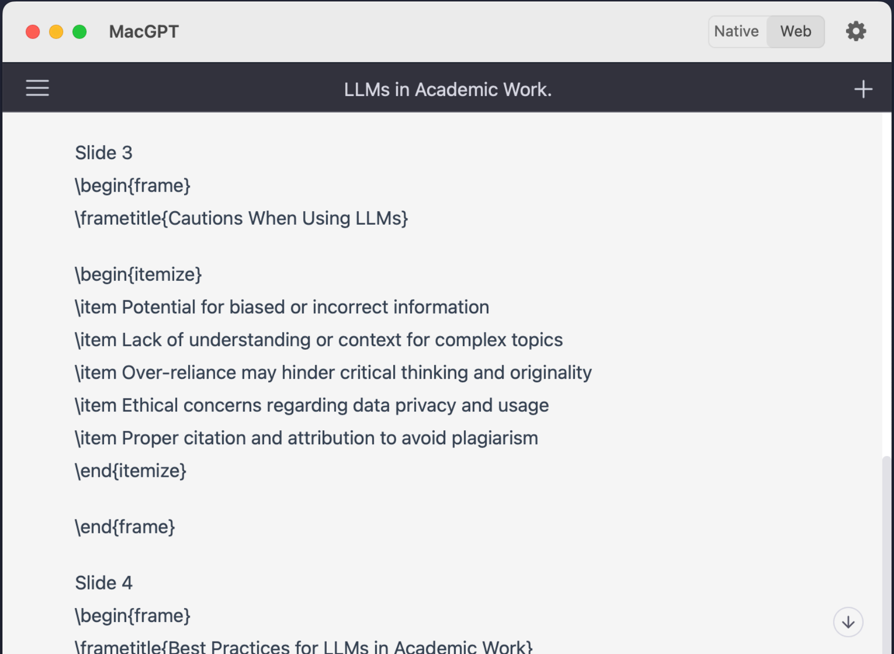
```
\begin{frame}
\frametitle{Benefits of LLMs in Academic Work}

\begin{itemize}
\item Rapid literature review and summarization
\item Guided brainstorming and idea exploration
\item Assisting with code development and debugging
\item Drafting and editing academic papers or presentations
\item Automating repetitive tasks, enabling focus on high-level analysis
\end{itemize}

\end{frame}
```

Slide 3

```
\begin{frame}
\frametitle{Cautions When Using LLMs}
```

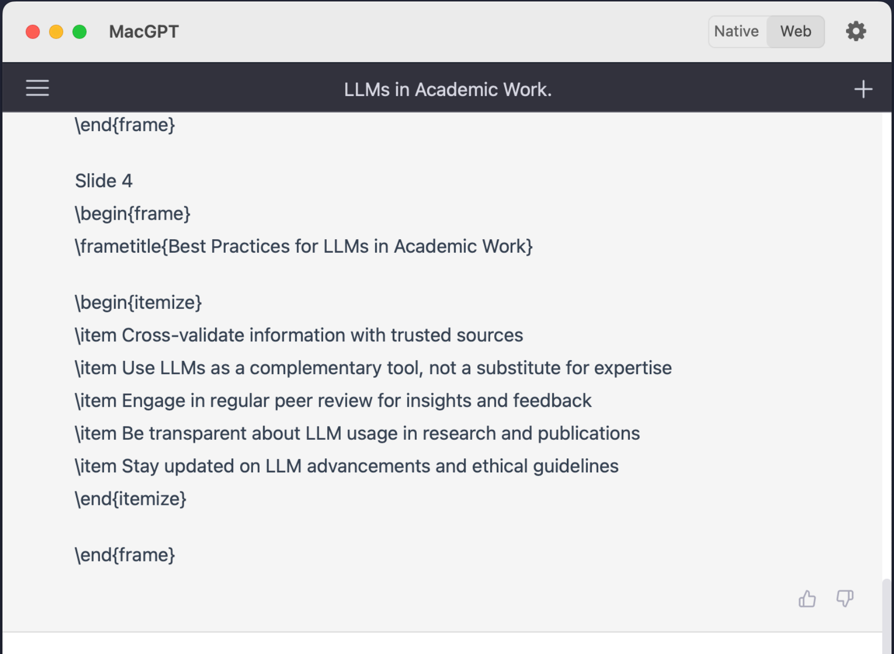


```
Slide 3
\begin{frame}
\frametitle{Cautions When Using LLMs}

\begin{itemize}
\item Potential for biased or incorrect information
\item Lack of understanding or context for complex topics
\item Over-reliance may hinder critical thinking and originality
\item Ethical concerns regarding data privacy and usage
\item Proper citation and attribution to avoid plagiarism
\end{itemize}

\end{frame}

Slide 4
\begin{frame}
\frametitle{Best Practices for LLMs in Academic Work}
```



```
\end{frame}

Slide 4
\begin{frame}
\frametitle{Best Practices for LLMs in Academic Work}

\begin{itemize}
\item Cross-validate information with trusted sources
\item Use LLMs as a complementary tool, not a substitute for expertise
\item Engage in regular peer review for insights and feedback
\item Be transparent about LLM usage in research and publications
\item Stay updated on LLM advancements and ethical guidelines
\end{itemize}

\end{frame}
```

SETTING UP YOUR MACHINES

01. Getting started — Part 2

DSIER [/dɪˈzaɪər/] — Summer 2022

Julian Hinz

Bielefeld University