Social Science PhD Student - Optimal Tech Stack, Sept 2024

Kevin Bryan, University of Toronto

October 24, 2024

Principles of Effective Research

- The goal: Research should be
  - Replicable: Others can reproduce your results
  - Accessible: Easy to understand and build upon
  - Quick: Efficient workflow and tools
- These tools and practices will help you achieve these goals
- The importance of this is **increasing** over time as the underlying technology improves

Replicability: Version Control

Why Version Control

"Not one piece of commercial software you have on your PC, your phone, your tablet, your car, or any other modern computing device was written with the 'date and initial' method." (Gentzkow and Shapiro)

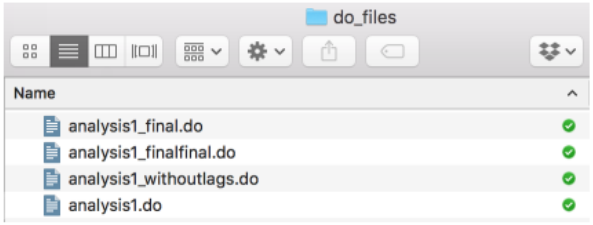
Version control is how you ensure that every change you make to your code, your writing, your data is tracked. It makes it *much* easier to fix mistakes later, and to collaborate!

Don’t Do This (from Battiston and Gars)

What we often do

Keep code and data in a Dropbox folder.

Apply “hacks” to deal with version control:



What we often do

Keep code and data in a Dropbox folder.

Apply “hacks” to deal with version control:

```
*SPECICATION IN THE DRAFT
*reg price weight i.year i.month, r

*TENTATIVE SPECIFICATION ADDED ON OCTOBER 3, 2021: CHECK BEFORE SUBMISSION
reg price weight i.year i.week, r
```

Setting Up Git

- The standard in software engineering is Git. To use this:
  - Windows: `https://git-scm.com/download/win`
  - macOS: Install via Homebrew with brew install git
  - Linux: Use package manager, e.g., sudo apt-get install git
- Configure Git from your command prompt window:
  - Set your name: git config –global user.name “Your Name”
  - Set your email: git config –global user.email “you@example.com”

Basic idea of Git

The idea of Github is to have a "repository" saving every version of every change you make to a project. GitHub is a service for hosting repositories online, while Git is the tool you use on your computer to track changes. After you make changes, you "stage" them to be uploaded, "commit" them with a note on what you did, then "push" them to Github. And when you start the day, "fetch" any changes your collaborators made, and "pull" them. Also, it's good practice to "PR" any changes made by others. Very easy to see exactly what changes you or your collaborators made!

My workflow (Windows)

- Initialize a repository: Set up a Github.com account, go in your command window to the folder for your project, type "git init" to initialize github, then type git remote add origin your-repo-url where the url of your Github.com project is
- When you start work: go in your command window to the project folder. Type "git pull origin master". You will now be on your 'master' branch.
- When you finish making changes to your code or writing and want to store it or send to coauthors, type "git add .", then "git commit -m "Describe what you did"", then "git push origin master"
- For small projects like ours, you probably don't need "branches" so don't worry about them for now
- You can now see on github every change you made, every time you commit. Easy!

More detail on Git

There are many guides to using Git (it has a small learning curve!), but I recommend:

- **Title**: Using Git for Social Scientists
- **Author**: Jesús Fernández-Villaverde
- **Link**: `https://www.sas.upenn.edu/~jesusfv/Chapter_HPC_5_Git.pdf`

Replicability: How to Write Code

Learn Python

My strong recommendation: Python plus an AI copilot (Cursor as of Sept 2024 is best). Julia is fine for certain computationally-intensive tasks. R is OK if you must. STATA/Matlab almost certainly never necessary.

Why? Python heavily used, so package mistakes found quickly. AI is very good at helping with Python. Python is free: `https://www.python.org/downloads/`

Pip is an easy way to install various libraries you need to perform special tasks, and is included when you download python this way.

Setting Up a Virtual Environment

- Why use virtual environments? Replicability! Especially across coauthors.
  - Isolate project dependencies
  - Avoid conflicts between packages
  - Easily share project requirements
- Create a virtual environment:
  - Navigate to your project directory in command window
  - "python -m venv myenv"
- Activate the environment:
  - Windows: "myenv\Scripts\activate"
  - macOS/Linux: "source myenv/bin/activate"
- If you use an IDE like Cursor, this can all be automated

Using Virtual Environments with pip

- In python, you "pip install x" for packages you need, from the command window
- The easiest way is to have a requirements.txt file listing the packages and their version in your base directory for each project
- Anyone who replicates in the future will replicate exactly!
  - e.g., "pip install numpy pandas matplotlib" from command line
- List installed packages (hopefully in your venv!)
  - "pip list"
- Create requirements file:
  - "pip freeze > requirements.txt"
- Install from requirements:
  - "pip install -r requirements.txt"

Replication Files

Every table, chart, text should replicate with **one** line of code, exactly

This is very easy if you use proper version control and python!

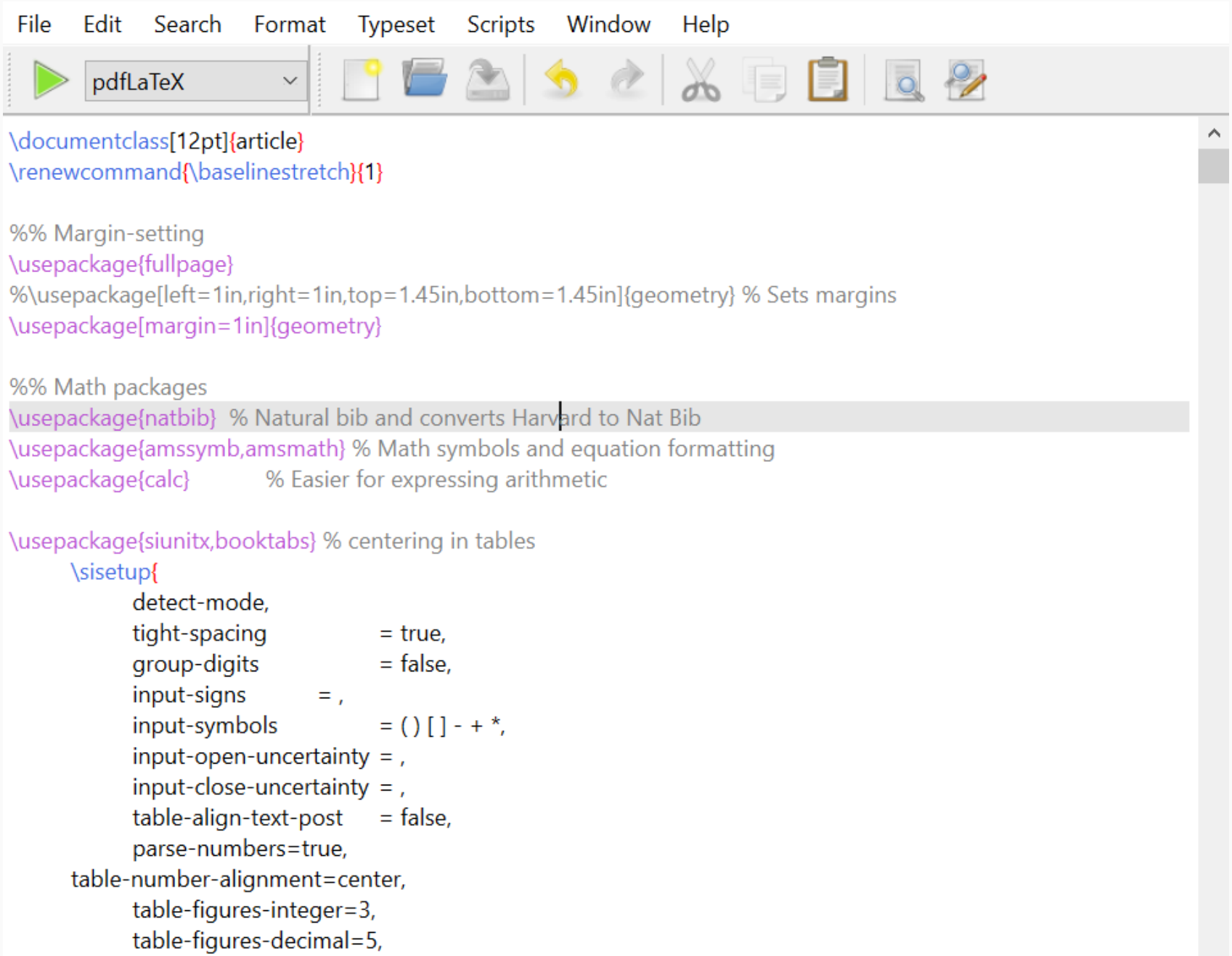
Accessibility

Accessibility

Your papers should be online, free-to-read, and available as soon as you feel comfortable sharing them. Because many readers will see a version of your paper which is not "publication-ready", you should make the document they read look nice.

The way to do this is with LaTeX.

LaTeX seems hard



But has many benefits

applying to more than 20 programs and forbids applying to more than 300. Our anecdotal understanding is that firms and workers in the RCT saw the 10-firm constraint as very natural.} As described in Section \ref{Section-EmpStrategy} below, we investigate four pre-registered outcomes using these rankings: the probability of applying at all, the probability of listing a firm in one's top N ranks (we use top rank and top three rank), and the absolute rank of the firm (with unranked firms treated as having been ranked $N+1$). As we report below, results are similar across all four outcomes, including top rank and top three rank, which are not affected by the ten application constraint, indicating that the constraint does not drive our findings. As discussed in Section \ref{Section-EmpStrategy} (``Worker selection into the RCT''), there is no evidence that workers found the ten application cap to be strange or that it limited who wished to participate. As seen in Table \ref{Number-Apps}, 45\% of applicants hit the cap, and the rate is similar across treatment arms (p-val of equality across four arms equals 0.77).

Math in $...$

Clickable "ref" links

\medskip

\textbf{Worker beliefs.} After submitting their ranked list of firms they wished to apply to, workers were asked to voluntarily evaluate three randomly selected ventures from the set of firms they had just considered applying to. This request was made only after the ranked true applications were sent, to limit Hawthorne effects. For each job seeker, the three firms in question remained the same for all belief questions. We asked candidates to estimate each firm's science quality, business quality, probability of raising capital at a valuation of \$1 million or more within a year, probability of having an IPO or being acquired for at least \$50 million within one year, and their interest in working for the company. Before answering the two questions on probabilities (i.e., the IPO and capital raise questions), subjects were given a standard ``explanation of probabilities'' as developed in the work of Charles Manski, and as used in many subsequent papers and large-scale surveys (see the review by \citet{BRUINEDEBRUIN20233}). The explanation gives examples of probabilities, and is designed to be simple and avoid leading subjects in any way (see \ref{Appendix-Screenshots} for the exact wording).

Citations very easy

The probability questions were incentivized with a risk-invariant quadratic scoring rule \citep{McKelvey&Page90}, under which applicants can win up to \$250 CAD ($\approx$ \$200 USD) on the basis of the accuracy of their predictions. Following past applied work using this scoring rule \citep{Hoffman-D4I}, we explain that the incentive system makes it optimal to state one's true beliefs, and we provide math formulas separately for people to look at if interested. \footnote{\label{FN-QSR}If a subject guesses correctly and states confidence level c, they get a lottery with a $2c-c^2$ probability of winning \$250 CAD and a $(1-c)^2$ probability of receiving zero. If they guess

And makes your work look professional

Information Frictions and Employee Sorting Between Startups*

Kevin A. Bryan† Mitchell Hoffman‡ Amir Sariri§

September 2024

Abstract

Would workers apply to better firms if they were more informed about firm quality? Collaborating with 26 science-based startups, we create a custom job board and invite business school alumni to apply. The job board randomizes across applicants to show coarse expert ratings of all startups' science and/or business model quality. Making ratings visible strongly reallocates applications toward higher-rated firms. This reallocation holds restricting to high-quality workers. Treatments operate in part by shifting worker beliefs about firms' right-tail outcomes. Despite these benefits, workers make post-treatment bets indicating highly overoptimistic beliefs about startup success, suggesting a problem of broader informational deficits.

Keywords: Hiring, job applications, startups, overconfidence
JEL Classifications: M50, M51

*We are grateful to Nick Bloom, Christian Dustmann, Claudine Gartenberg, Matt Gentzkow, Andrea

Table 2: Balance Table

	No Info	Science Info	Business Info	Science & Business Info	p-value
Panel A: Job Board RCT					
Male	0.77	0.72	0.79	0.69	0.51
City is SEP HQ	0.56	0.47	0.52	0.47	0.70
Graduation year	2012	2012	2013	2012	0.69
Startup founder	0.24	0.23	0.12	0.13	0.14
Startup employee	0.27	0.28	0.19	0.28	0.60
Employed	0.80	0.81	0.69	0.70	0.24
Yrs of exp	9.95	10.69	8.64	10.36	0.55
STEM	0.38	0.49	0.39	0.36	0.49
Worker Quality (1-10)	5.20	5.66	4.90	5.00	0.35
Num. Workers	66	53	67	64	
Panel B: MBA Student RCT					
Male	0.54	0.38	0.41	0.44	0.39
White	0.26	0.29	0.24	0.21	0.82
Hisp./Latino	0.09	0.08	0.06	0.06	0.94
Asian	0.30	0.33	0.49	0.35	0.25
Num. Workers	46	48	49	48	

Main notes: This table compares applicant characteristics across treatment groups for the Primary RCT (Panel A) and Secondary RCT (Panel B). "Science info" and "Business Info" in column headers refer to the subjects that received science and business scores.
Panel A: In the Primary RCT, randomization was stratified on gender, city, and year of graduation at the time potential applicants were contacted. Other variables were not observable prior to application. Worker Quality is based on a startup-focused HR expert's evaluation of resume quality. Of 259 resumes submitted in the job board, 9 were ineligible and were removed from all analysis. Ineligible candidates were forwarded the link to the job board from eligible candidates. The remaining 19,109 individuals contacted did not apply to any firm.
Panel B: In the Secondary RCT, randomization was not stratified.

The controls $\boldsymbol{X}_{nf}$ include firm fixed effects to control for underlying firm quality, as well as any strata dummies for RCTs when we do a stratified randomization. We cluster standard errors by worker, as our treatments are randomized across workers.

Our pre-analysis plan (PAP) also states that worker beliefs and perceptions of firm quality will be secondary outcomes. These are analyzed in the same way as our results on worker application rankings. Finally, the PAP specifies that we will run heterogeneity analyses based on whether workers have a STEM degree or not.[17]

Randomization check. Table 2 shows that the four treatment groups of the Primary RCT (Panel A) and Secondary RCT (Panel B) are balanced on observables.

And citations are really easy

```
@article{anton2002sale,
  title={The sale of ideas: Strategic disclosure, property ri
  author={Anton, James J and Yao, Dennis A},
  journal={Review of Economic Studies},
  volume={69},
  number={3},
  pages={513--531},
  year={2002},
}

@article{aran2021equity,
  title={Equity Illusions},
  author={Aran, Yifat and Murciano-Goroff, Raviv},
  journal={Journal of Law, Economics, and Organization},
  year={2023},
  month={Forthcoming},
}

@Article{Arcidiacono04,
  author={Arcidiacono, Peter},
  title={Ability Sorting and the Returns to College Major},
  journal={Journal of Econometrics},
  year=2004,
  volume={121},
  number={1-2},
  pages={343-375},
}
```

This is .bib citation format

To add this paper

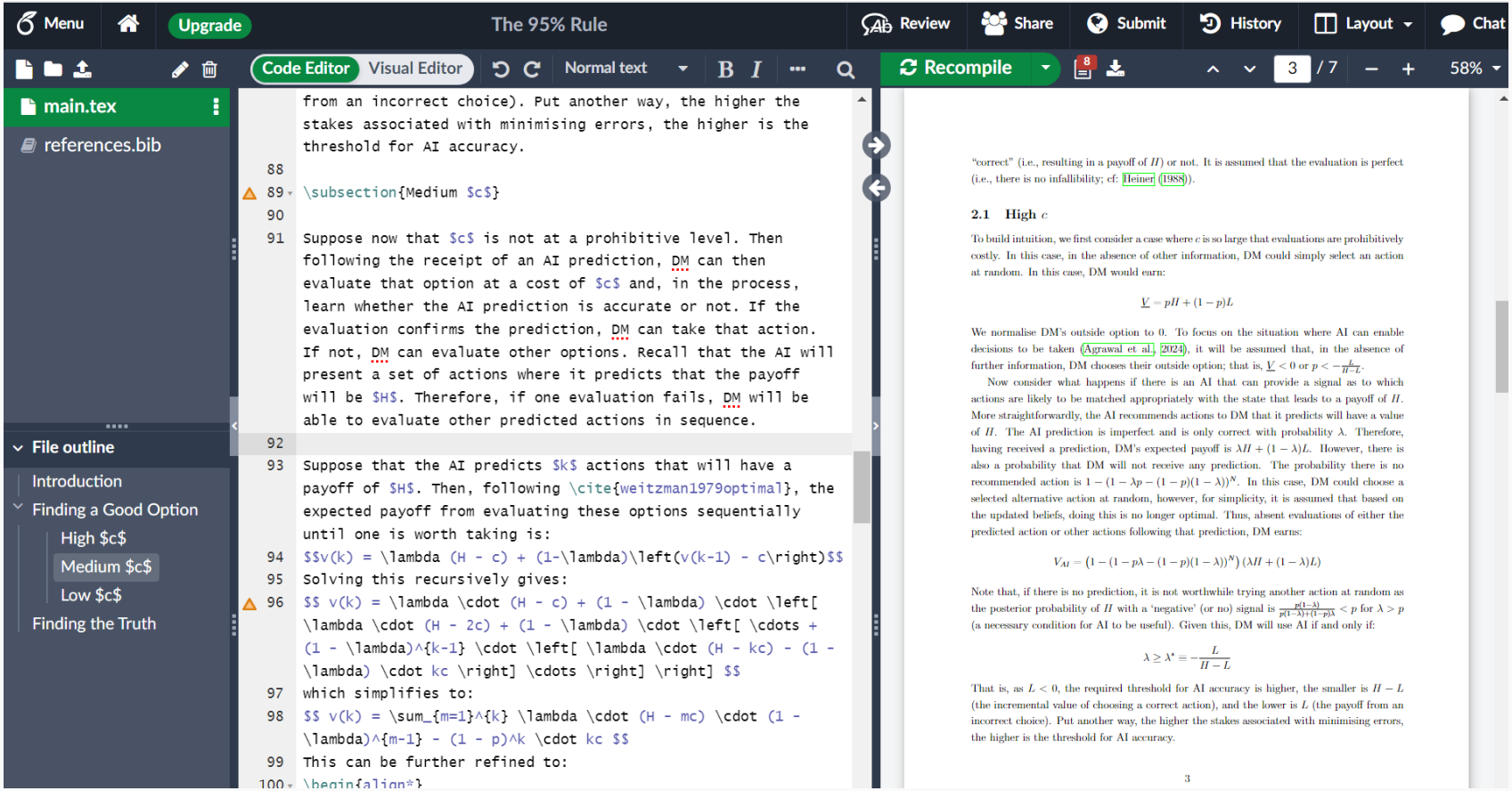
Why LaTeX

- Your math looks pretty
- Your tables are pretty, auto-rotate on pdf if you set things up right, and can be exported in the right format directly from your coding language of choice
- Typesetting looks good
- Bibliography style is changed by changing one line, and references auto-added and clickable
- Even paper style can be changed with one line. When Journal X accepts, you just get their latex style and change one line on your paper - done!

Writing in LaTeX with Overleaf

- What: Online LaTeX editor with collaboration features
- Why:
  - No local LaTeX installation needed (can be annoying)
  - Real-time collaboration
  - Even for solo projects, probably easier than TeXworks or other local files
- How to use:
  - Sign up at `https://www.overleaf.com`
  - Create a new project or use a template
  - Write your LaTeX code in the editor
  - Compile and view output in real-time

Overleaf is very easy to use!



Overleaf and Github

You can even integrate Overleaf into your Github repository for a given project. See Git and Overleaf at https://medium.com/codex/git-for-economists-a-practical-guide-7d10faf4224f

LaTeX and Beamer

By the way, you can also make slides with LaTeX using the "Beamer" style. It's just a few lines of code to add at the start, and you can easily import all of your tables, figures, etc. as before. These slides are Beamer! This is optional depending on your preference - I sometimes use ppt anyway.

I also recommend reading, before you start making slides and figures and tables, a short book on graphic design. The go-to here is *The Visual Display of Quantitative Information* by Tufte. Go to the library now and look at it - if you have not considered how an academic must care about design, it will blow your mind.

Speed

How to write code, theory, experiments quickly

It is 2024. The answer to "how should I do X quickly" is **Use AI to help you!**

When coding, I recommend Cursor.ai. The free version is a wonderful VSCode based interface for writing your python, but it also integrates frontier AI to assist you. I would guess I physically type 20% of the code I write; I'm mainly just directing AI and proofreading what it gives me. The paid version is $20/month and well worth it.

If you get an error in your code, Cursor and/or just pasting the file plus the error message into Claude or GPT will find the mistake 99% of the time. Don't bang your head against the wall!

How to write code, theory, experiments quickly

If you do theory, you can get help on proofs, or even double-check them, with frontier models like GPT-o1. These are not perfect, but at this point are good enough to be useful on a daily basis for me.

If you are running a survey or experiment, pasting your instructions into GPT or Claude and asking for "hypothetical answers" can help you find mistakes or misleading language in the same way a pilot survey might. The software tool Expected Parrot does this at scale and is specifically for social science experiments.

How to write code, theory, experiments quickly

You should always "proofread" anything you do with a frontier LLM. They are better than any spell or grammar check. Grammarly is very good if you have an account. I have specific stylistic things I ask the LLMs to look at using a custom prompt. Indeed, I give my papers to LLMs before submission and ask for hypothetical referee reports - the LLMs often spot things I'd missed!

Efficient Note-Taking Strategies

You should have a folder with every paper you find interesting, organized by topic.

I also have .txt files with "General paper ideas", notes for each project I'm working on, and "General notes on papers I've read". These documents are more and more readable every year with AI. For instance, I used the GPT API to code a searchable database of all the papers in my folder alongside my own notes. Google's NotebookLM is 99% as good and free. These tools are getting better all the time. You should add notes to your "lab book" as you read the newspaper, attend seminars, and so on!

My AI toolkit

I have GPT, Claude, Cursor and Perplexity (essentially, Google search with AI) subscriptions, plus I use APIs from the major LLMs in my research (a topic which goes beyond these slides). This costs over 100 CAD per month. For students, I would recommend one LLM (either Claude or GPT) and Cursor if you code. It will be very much worth it!

For teaching, I also use AI heavily. Joshua and I even started a company called AllDayTA to use these tools to generate question banks, spin up a virtual TA for the students, and so on, all based on very specific class content. When you start teaching, let me know and I'll set you up!

This is a starting point

AI is getting better *rapidly*. I use these tools every day

They will be better next year, and better still the year after.

What I've shown you in "step one". For your research, esp. if you use unstructured data like historical text, audio, or video, there are wild research possibilities that have become feasible. I recommend Anton Korinek's AI series for the Journal of Economic Literature (updated every six months) and Melissa Dell's article in the same article alongside her EconDL website of deep learning tools for economists.

Bringing It All Together

- Replicable work: version control using a modern system + tracked environments for your coding language noting every package you use + "PR" your collaborator's changes (Git + Python)
- Accessible work: good graphic design + typography from LaTeX, with your work on your website or ArXiv/SocArXiv as soon as possible
- Quick work: Use modern AI tools for coding, survey testing, brainstorming, copyediting, proof checking, and note taking