Causal Inference

Mixtape Session

Roadmap

- Hidden curriculum
  - Background
  - Empirical workflow
  - Hierarchical folder structure
  - Naming conventions
  - Version control
  - Soft skills

What this course is about

- This course will cover topics in “causal inference”, a field within statistics and the quantitative social sciences
- I will use lectures, simulations, replications, discussion of papers, and coding together to teach this material
- My goal is to help you become competent so that you feel confident enough to apply this material in your own research

Foundations of scientific knowledge

- Scientific methodologies are the epistemological foundation of **scientific knowledge**, which is a particular kind of knowledge
- Science **does not** collect evidence in order to “prove” what people already believe or want others to believe.
- Science is **process oriented**, not **outcome oriented**.
- Good science allows us to accept unexpected and sometimes even undesirable answers.
- Causal inference is one of science’s cornerstones

What is causal inference

- Don’t think of causal inference as a synonym for statistics or even econometrics
- Causal inference is a sub-field within statistics and econometrics that focuses on *estimation* of *causal effects* using either experimental or non-experimental data
- Causal effects are the consequences of actions and interventions like government policy (e.g., Medicaid), firm marketing (e.g., Black Friday sales) and other actions (e.g., raising prices), or even personal choices (e.g., marriage; having a child; going to college)
- Awarded the Nobel Prize in economics in 2021 (David Card, Joshua Angrist and Guido Imbens)

Causal inference is necessary

- Causal inference is not a substitute for good sense or theoretical knowledge
- Causal inference provides *a priori* information to decision makers about what to expect with policy changes
- Policy makers are everywhere – in households, firms, government agencies
- Policies are actions chosen; causal effects are the outcomes of those actions
- The task of causal inference is to separate the causal effects from spurious accidents that often happen with but which are technically not the result of the actions chosen
- Failure to distinguish between causality and correlation can lead to fatal mistakes
- But causal inference als is not a substitute for the hard part of research

School of thought

- I will tend to focus on a specific school of thought within the broader stream of work in statistics and econometrics focused on causal inference associated with the 1970s and 1980s labor group at Princeton
- Key architects include Orley Ashenfelter, David Card (2021 Nobel Prize winner), Josh Angrist (2021 Nobel Prize winner), Guido Imbens (2021 Nobel Prize Winner), Bob LaLonde, Alan Krueger, Don Rubin, Alberto Abadie and many more
- Usually associated with the “potential outcomes” model, randomization and natural experiment methodologies

Gaps in coverage

- Because it tended to come from the Princeton group, it followed as a body of work *applied micro* researchers, and less so econometrics, and even less so data science
- Incorporation into traditional econometrics curriculum, including textbooks, lagged its adoption in applied social sciences
- Adoption tended to follow academic job placements, empirically oriented field classes (labor, public, health, development), editorial positions at high ranked journals
- My goal with this course is to “fill in” the spots it hasn’t moved into with “explainer” style workshops, books and other writings

Biographical sketch

- Scott Cunningham, Professor of economics at Baylor (Waco Texas)
- Graduated from University of Georgia in 2007 with fields in econometrics, industrial organization, public and labor
- Research focus on sex work, mental healthcare, drug policy, abortion policy and crime
- Author of one book, co-editor of another, 19 peer reviewed articles, 4 book chapters
- Co-editor at the Journal of Human Resources

Mission: “to democratize causal inference”

- **Book**: Causal Inference: the Mixtape (Yale University Press) at https://mixtape.scunning.com
- **Substack**: Causal Inference: the Remix at https://causalinf.substack.com
- **Mixtape Sessions**: Platform hosting on-site and online workshops on causal inference at https://www.mixtapesessions.io
- **Mixtape Tracks**: Self-paced courses at https://mixtape.thinkific.com
- **Twitter**: @causalinf
- **Universities**: Baylor University, UT-Austin, University of Oxford, Universidad Catolica del Uruguay and more

My journey into causal inference

- In grad school, I knew I was going to be an empiricist, so I made econometrics my main field – passed field exam on second attempt
- Research topics have included risky sex, drug policy, abortion, mental healthcare and health topics more generally
- I developed courses on causal inference, as well as wrote a book on the subject, because many students do not know it even if they have studied econometrics
- This is because causal inference isn't taught historically in traditional econometrics

My pedagogy

- I am not an econometrician, although it was my field in grad school and I still love the field
- I am a consumer of econometrics – I use these designs in my work when I need to, and don't when I don't need to
- If I can understand this material well enough to teach it, anyone can learn it
- I want researchers to feel empowered and knowledgeable enough to implement these methods profitably
- Tend to emphasize the art as well as the science as well as the coding

Some reading resources

1. Causal Inference: the Mixtape. Cunningham. Free online version at `mixtape.scunning.com`. Introductory to intermediate
2. Mostly Harmless Econometrics. Angrist and Pischke. Classic, advanced
3. Mastering Metrics. Angrist and Pischke. Classic, introductory
4. The Effect. Nick Huntingon-Klein. Excellent, introductory
5. Counterfactuals and Causal Inference. Morgan and Winship. Excellent, introductory to intermediate
6. Causal Inference for Statistics, Social and Biomedical Sciences. Imbens and Rubin. Advanced
7. Book of Why. MacKenzie and Pearl. Introduction to graphs.

Data Scarcity vs Data Abundance

- Think about it: for most of the history of science, and really all the social sciences, **there was practically no data** – only theories about data
- Today we drown in data. Ordinary workers have more data than they know what to do with
- As such causal inference cannot be easily separated from the work itself, which means programming skill – both in the analysis stage, but maybe more importantly the data wrangling and cleaning stage

Code and Software

- For causal inference we need:
  - → data
  - → software for data
  - → understanding of statistics and causality
  - → skill using software, cleaning and analyzing data, applying our models and interpreting our results
- But which software? SAS, SPSS, Eviews, **R, Stata, python**, julia and more
- Language agnostic programming principles that are necessary but not covered in econometrics courses (“hidden curriculum”)

Making mistakes

- Once upon a time there was a boy who wrote a job market paper using the NLSY97.
- This boy presented the findings a half dozen times, spoke to the media a few times, got 17 interviews at the ASSA, 7 flyouts, and an offer from Baylor
- He submitted the job market paper to the *Journal of Human Resources*, a top field journal in labor, and received a “revise and resubmit” request from the editor (woo hoo!)

Coding error

- But then digging into his one directory, he found countless versions of his do file and hundreds of files with random names
- And once he finally was able to get the code running again, he found a critical coding error that when corrected (“destroyed”) his results
- The young boy was devastated and never resubmitted which he does not recommend (but he was sad!)

All competent empirical work is a mousetrap

“Happy families are all alike; every unhappy family is unhappy in its own way.” - Leo Tolstoy, Anna Karenina

“Good empirical work is all alike; every bad empirical work is bad in its own way.” - Scott Cunningham, This slide

Cunningham Empirical Workflow Conjecture

- The cause of most of your errors is **not** due to insufficient knowledge of syntax in your chosen programming language
- The cause of most of your errors is due to a poorly designed empirical workflow

Workflow

Wikipedia definition:

> *“A workflow consists of an orchestrated and repeatable pattern of activity, enabled by the systematic organization of resources into processes that transform materials, provide services, or process information.”*

Dictionary definition:

> *“the sequence of industrial, administrative, or other processes through which a piece of work passes from initiation to completion.”*

Empirical workflow

- Workflow is a fixed set of routines you bind yourself to which when followed identifies the most common errors
  - → Think of it as your morning routine: alarm goes off, go to wash up, make your coffee, check Twitter, repeat *ad infinitum*
- Finding the outlier errors is a different task; empirical workflows catch typical and common errors created by the modal data generating processes

Why do we use checklists?

- Before going on a trip, you use a checklist to make sure you have everything you need
  - → Charger (check), underwear (check), toothbrush (check), passport (oops), . . .
- The empirical checklist is solely referring to the intermediate step between “getting the data” and “analyzing the data”
- It largely focuses on ensuring data quality for the most common, easiest to identify, situations you’ll find yourself in

The Checklist

- Empirical workflows are really just a checklist of actions you will take before analyzing your data
- It is imperative that you do not analyze your data (e.g., “explore the data”, “run some regressions”) until you have checked everything off
- Your checklist should be a few simple, yet non-negotiable, programming commands and exercises to check for coding errors
- These are some of mine – feel free to add your own

Empirical workflows require scarce time inputs

- All of us are living at the edge of our resource constraints, and our most scarce resources is *time*
- To do anything, we must sacrifice something else because all activities use time – including cleaning and arranging our data
- Just like running a marathon involves far far more time training than you ever spend running the marathon, doing empirical research involves far far more time doing tedious, repetitive tasks
- Since you do the tedious tasks repeatedly, they have the *most* potential for error which can be catastrophic ("anything that can happen will happen with enough trials")
- However long you think cleaning and organizing the data will take, **multiply it by 10** – prepare for it by managing your expectations

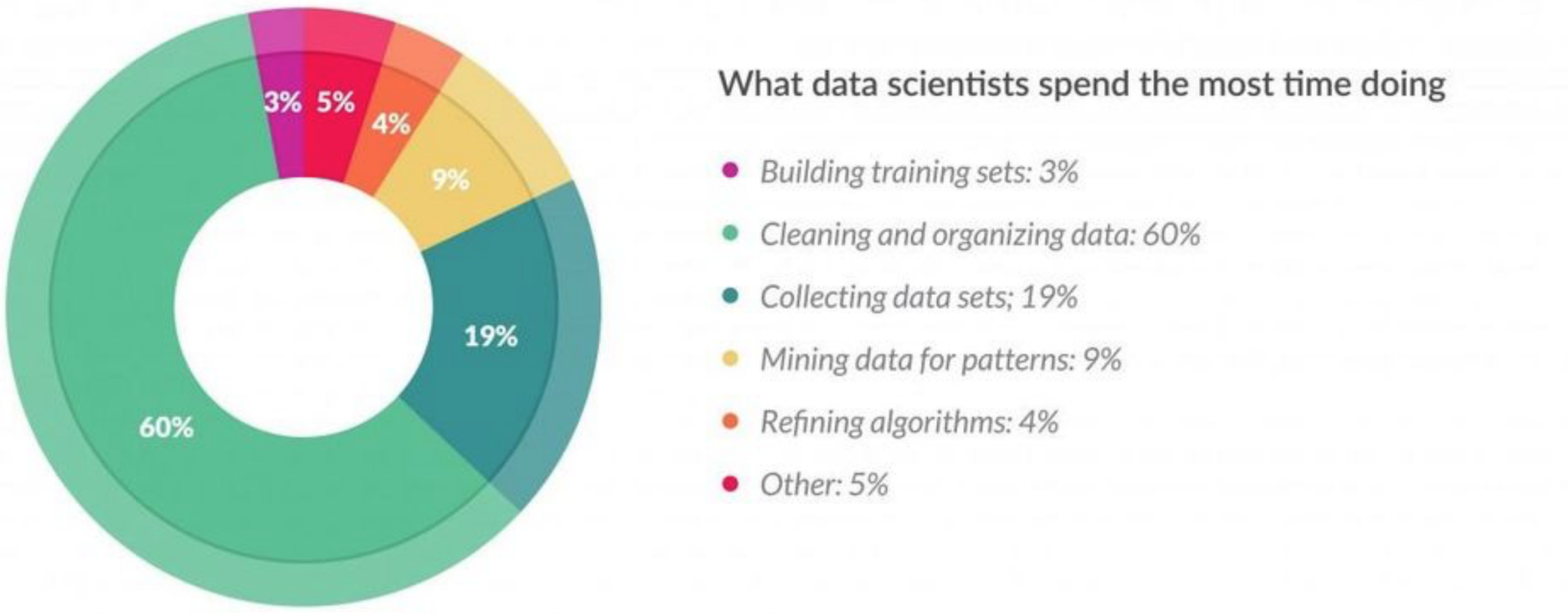


Figure: Image from Wenfei Xu at Columbia

Read the codebook

- Datasets often come with very large documents (either physical or digital) describing in detail the data production and arrangement
- You must become as much of an expert on the codebook as you are on your own research topic
- The codebook explains how to interpret the data you have acquired and it is not a step you can skip
- Set aside time to study it, and have it in a place where you can regularly return to it
- This goes for the `readme` that accompanies some datasets, too.

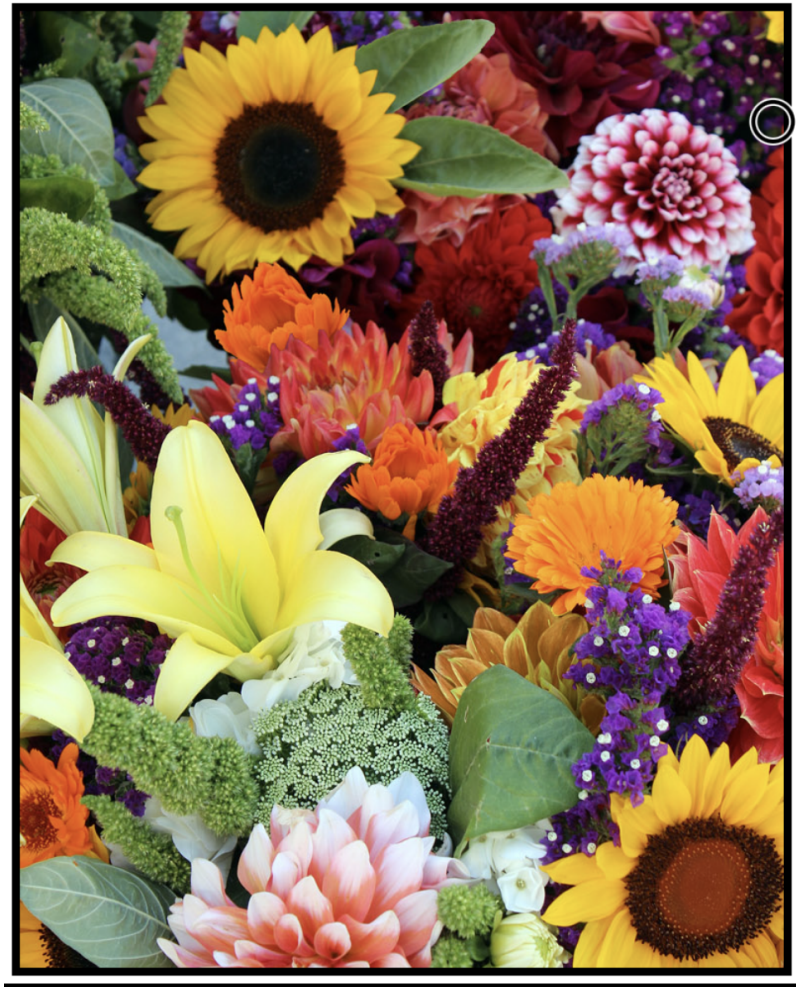

© Robert Del Tredici

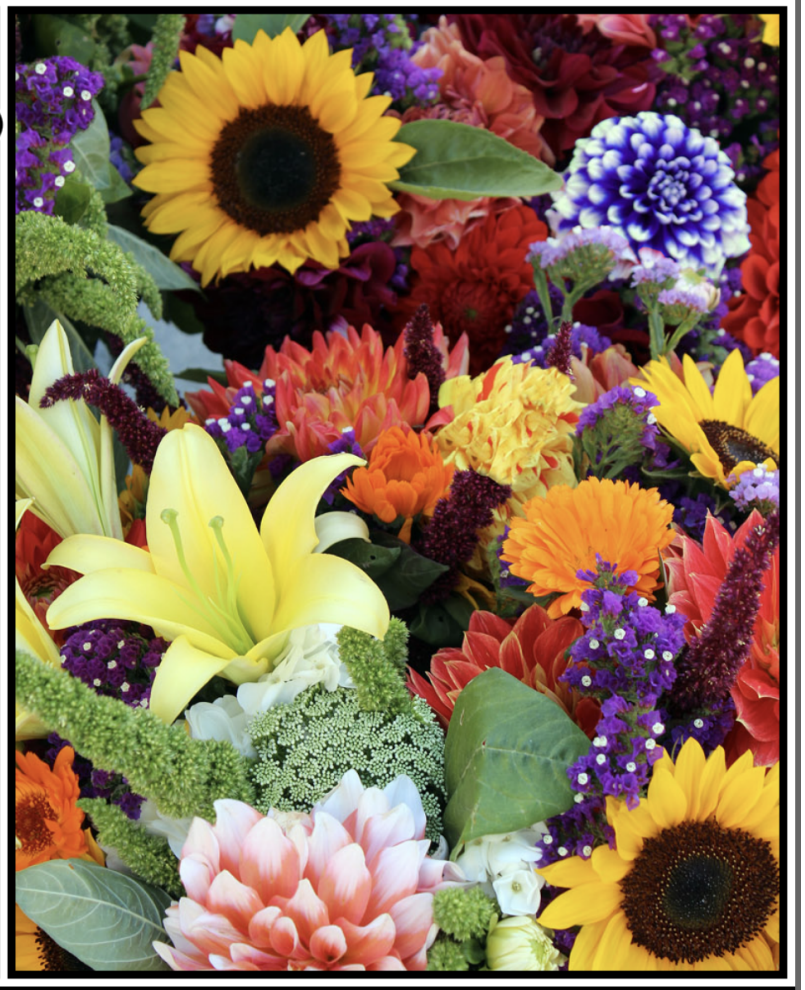
Do Not Touch the Original Data

- Empirical workflows require data manipulation
- It is **imperative** that you always and only work with **copies**
- Never save over the original dataset – be careful of Excel which may do it automatically
- Avoid this by storing the raw data separate from your copies
- Do not alter the raw data – you will ruin it and may not get it back

SPOT THE DIFFERENCE DAILY

January 2, 2022





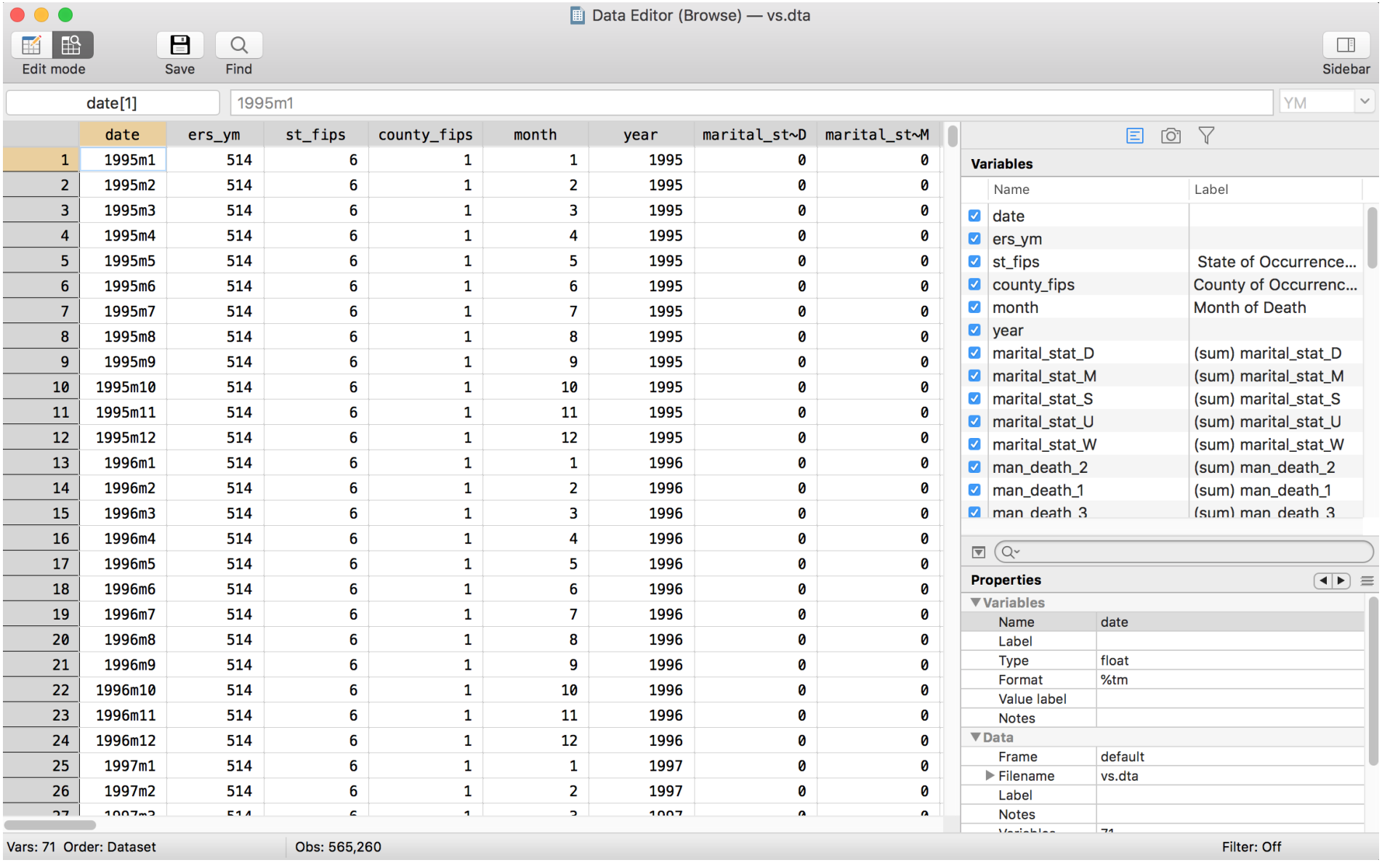
HINT ?

Spot the 10 differences

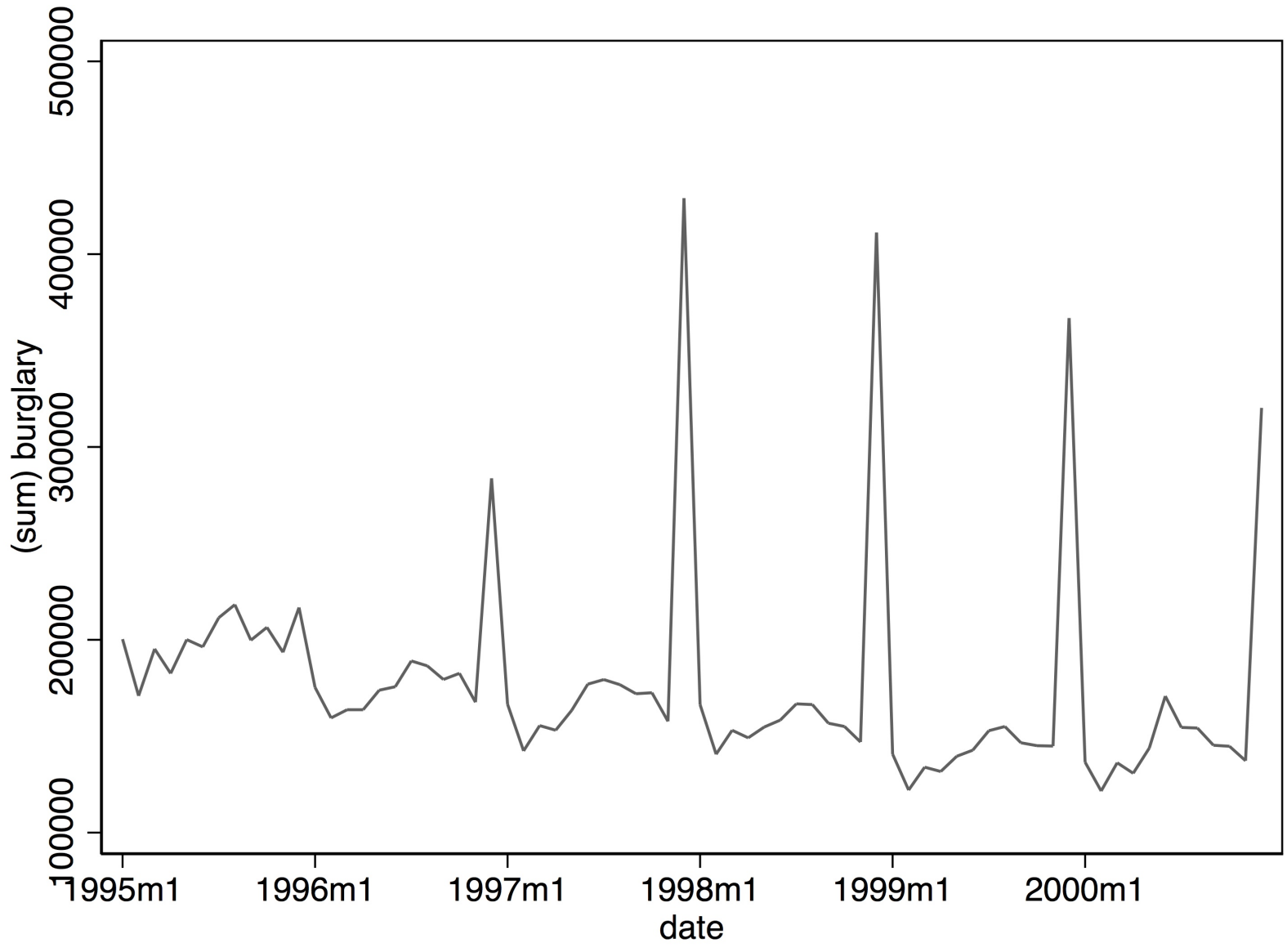
00:03

Try and spot the problems

- Our eyeballs evolved to spot patterns
- We can therefore use it to find things that belong but also things that don't
- First just scan the data in its spreadsheet form – get comfortable with what you're going to bre using
- Use `browse` or excel to just read the spreadsheet with your eyes.
- See if anything jumps out



	date	ers_ym	st_fips	county_fips	month	year	marital_st~D	marital_st~M
1	1995m1	514	6	1	1	1995	0	0
2	1995m2	514	6	1	2	1995	0	0
3	1995m3	514	6	1	3	1995	0	0
4	1995m4	514	6	1	4	1995	0	0
5	1995m5	514	6	1	5	1995	0	0
6	1995m6	514	6	1	6	1995	0	0
7	1995m7	514	6	1	7	1995	0	0
8	1995m8	514	6	1	8	1995	0	0
9	1995m9	514	6	1	9	1995	0	0
10	1995m10	514	6	1	10	1995	0	0
11	1995m11	514	6	1	11	1995	0	0
12	1995m12	514	6	1	12	1995	0	0
13	1996m1	514	6	1	1	1996	0	0
14	1996m2	514	6	1	2	1996	0	0
15	1996m3	514	6	1	3	1996	0	0
16	1996m4	514	6	1	4	1996	0	0
17	1996m5	514	6	1	5	1996	0	0
18	1996m6	514	6	1	6	1996	0	0
19	1996m7	514	6	1	7	1996	0	0
20	1996m8	514	6	1	8	1996	0	0
21	1996m9	514	6	1	9	1996	0	0
22	1996m10	514	6	1	10	1996	0	0
23	1996m11	514	6	1	11	1996	0	0
24	1996m12	514	6	1	12	1996	0	0
25	1997m1	514	6	1	1	1997	0	0
26	1997m2	514	6	1	2	1997	0	0


(sum) burglary
100000
200000
300000
400000
500000
1995m1
1996m1
1997m1
1998m1
1999m1
2000m1
date

Missing observations

- Check the size of your dataset in Stata using `count`
- Check the number of observations per variable in Stata using `summarize`
  - → String variables will always report zero observations under `summarize` so `count if X==""` will work
- Use `tabulate` also because oftentimes missing observations are recorded with a -9 or some other illogical negative value

Missing years

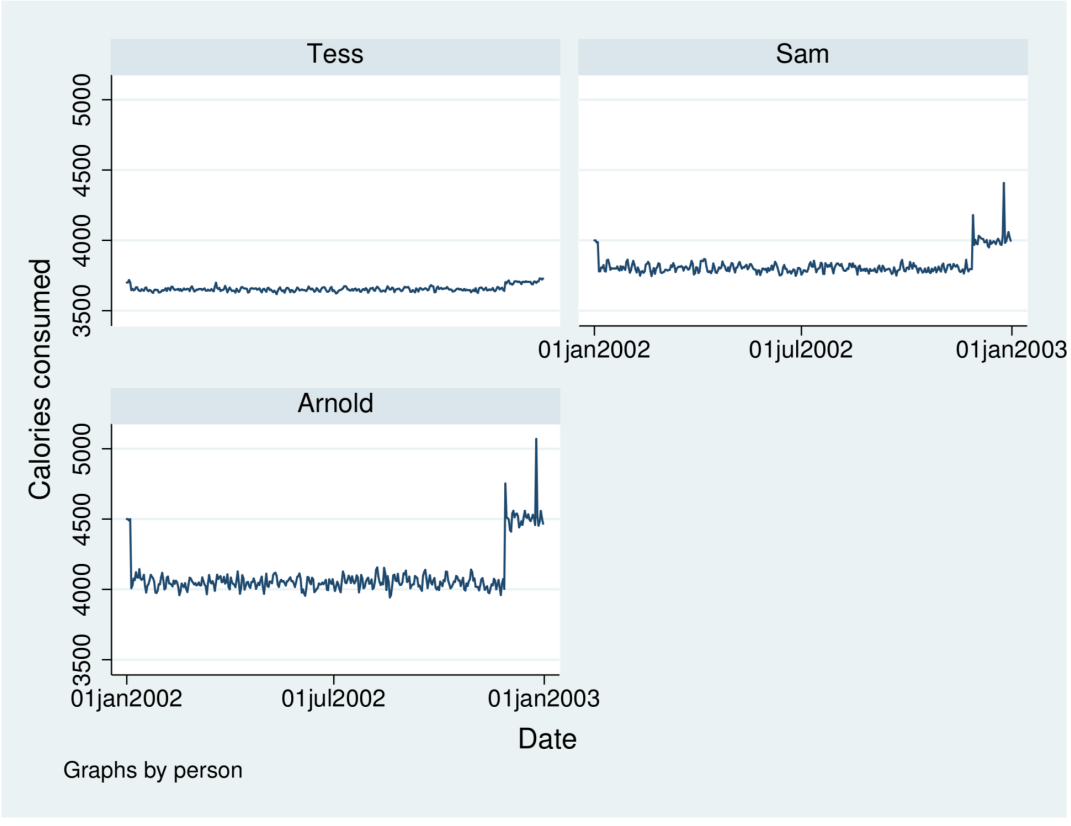
- Panel data can be overwhelming bc looking at each state/city/firm/county borders on the impossible
- Start with `collapse` to the national level by year and simply `list` to see if anything looks strange
  - → What's "strange" look like?
  - → Well wouldn't it be strange if national unemployment rates were zero in any year?
- You can use `xtline` to see time series for panel identifiers, with or without the subcommand of `overlay`

```
. collapse (sum) male_homicide female_homicide, by(year)

. list
```

	year	male_h~e	female~e
1.	1995	0	0
2.	1996	0	0
3.	1997	0	0
4.	1998	0	0
5.	1999	0	0
6.	2000	0	0
7.	2001	0	0
8.	2002	0	0
9.	2003	4474	910
10.	2004	4270	900
11.	2005	4450	895
12.	2006	4479	889
13.	2007	4480	895
14.	2008	4228	893
15.	2009	3857	866

```
. xtline calories, tlabel(#3)
```



```
. xtline calories, overlay
```

Panel observations are $N \times T$

- Say you have 51 state units (50 states plus DC) and 10 years
- $51 \times 10 = 510$ observations
- If you do not have 510 observations, then you have an unbalanced panel; if you have 510 observations you have a balanced panel
- Check the patterns using `xtdescribe` and simple counting tricks

Results

```
.
. tsset id date
       panel variable:  id (unbalanced)
        time variable:  date, 1995m1 to 2009m12
                delta:  1 month

.
.
end of do-file

. xtdescribe

      id:  4, 5, ..., 270                                    n =         69
    date:  1995m1, 1995m2, ..., 2009m12                      T =        180
           Delta(date) = 1 month
           Span(date)  = 180 periods
           (id*date uniquely identifies each observation)

Distribution of T_i:   min      5%     25%       50%       75%     95%     max
                        24      48     180       180       180     180     180

     Freq.  Percent    Cum. |  Pattern
 ---------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
> ---------------
       58     84.06   84.06 |  111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
> 111111111111111
        3      4.35   88.41 |  ............................................................111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
> 111111111111111
        2      2.90   91.30 |  ............................................................................................................................................................111111111
> 111111111111111
        2      2.90   94.20 |  ........................111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
> 111111111111111
        1      1.45   95.65 |  ................................................................................................................................................111111111111111111111
> 111111111111111
        1      1.45   97.10 |  ....................................................................................................................................111111111111111111111111111111111
> 111111111111111
        1      1.45   98.55 |  ....................................................................................111111111111111111111111111111111111111111111111111111111111111111111111111111111
> 111111111111111
        1      1.45  100.00 |  111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111.............................................
> ...............
 ---------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
> ---------------
       69    100.00         |  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
> XXXXXXXXXXXXXXX

.
```

log on (text)

```
. gen one = 1

. bysort county_group: egen count=sum(one)

. ta count
```

count	Freq.	Percent	Cum.
24	48	0.42	0.42
36	36	0.31	0.73
48	48	0.42	1.15
96	96	0.84	1.99
120	480	4.19	6.18
156	312	2.72	8.90
180	10,440	91.10	100.00
Total	11,460	100.00	

Merge

- During a stage of arranging datasets, you will likely merge – oftentimes a lot
- Make sure you count before and after you merge so you can figure out what went wrong, if anything
- Also make sure you're using the contemporary 1:1 (etc.) syntax as many an excellent empiricists have been hurt by merge syntax errors

```
. count
  48,600

. do "/Users/scott_cunningham/Dropbox/Indy/Do/.tm-stata-55642.do"

. merge 1:1 id date using ../data/seer.dta
(note: variable month was byte, now float to accommodate using data's values)

    Result                           # of obs.
    -----------------------------------------
    not matched                       517,044
        from master                       384  (_merge==1)
        from using                    516,660  (_merge==2)

    matched                            48,216  (_merge==3)
    -----------------------------------------

. ta _merge

                _merge |      Freq.     Percent        Cum.
-----------------------+-----------------------------------
       master only (1) |        384        0.07        0.07
        using only (2) |    516,660       91.40       91.47
           matched (3) |     48,216        8.53      100.00
-----------------------+-----------------------------------
                 Total |    565,260      100.00

.
.
end of do-file

. count
  565,260
```

Folder structure and directories

- After my coding error with my job market paper, I researched how problems like these tended to happen
  1. **hierarchical folder structure**
  2. automation
  3. naming conventions
  4. version control
- Also see Gentzkow and Shapiro's 2014 "Code and Data for the Social Sciences: A Practitioner's Guide" `https://web.stanford.edu/~gentzkow/research/CodeAndData.pdf`

REPORT

FILE NOT FOUND

A generation that grew up with Google is forcing professors to rethink their lesson plans

By Monica Chin | @mcsquared96 | Sep 22, 2021, 8:00am EDT

Illustrations by Micha Huigen

Catherine Garland, an astrophysicist, started seeing the problem in 2017. She was teaching an engineering course, and her students were using simulation software to model turbines for jet engines. She'd laid out the assignment clearly, but student after student was calling her over for help. They were all getting the same error message: The program couldn't find their files.

Garland thought it would be an easy fix. She asked each student where they'd saved their project. Could they be on the desktop? Perhaps in the shared drive? But over and over, she was met with confusion. "What are you talking about?" multiple students inquired. Not only did they not know where their files were saved — they didn't understand the question.

Gradually, Garland came to the same realization that many of her fellow educators have reached in the past four years: the concept of file folders and directories, essential to previous generations' understanding of computers, is gibberish to many modern students.

Helping your future self find what she needs

- Remember your future self is operating at the edge of her production possibilities frontier and has no more time left to screw around
- The typical applied micro project may have hundreds of files of various type and will take *years* just to finish not including time to publication
- So simply finding the files you need becomes more difficult if everything is stored in the same place
- Goal is to help her find what she needs quickly, accurately and without breaking anything else
- Hierarchical folder structures are the main way we do this

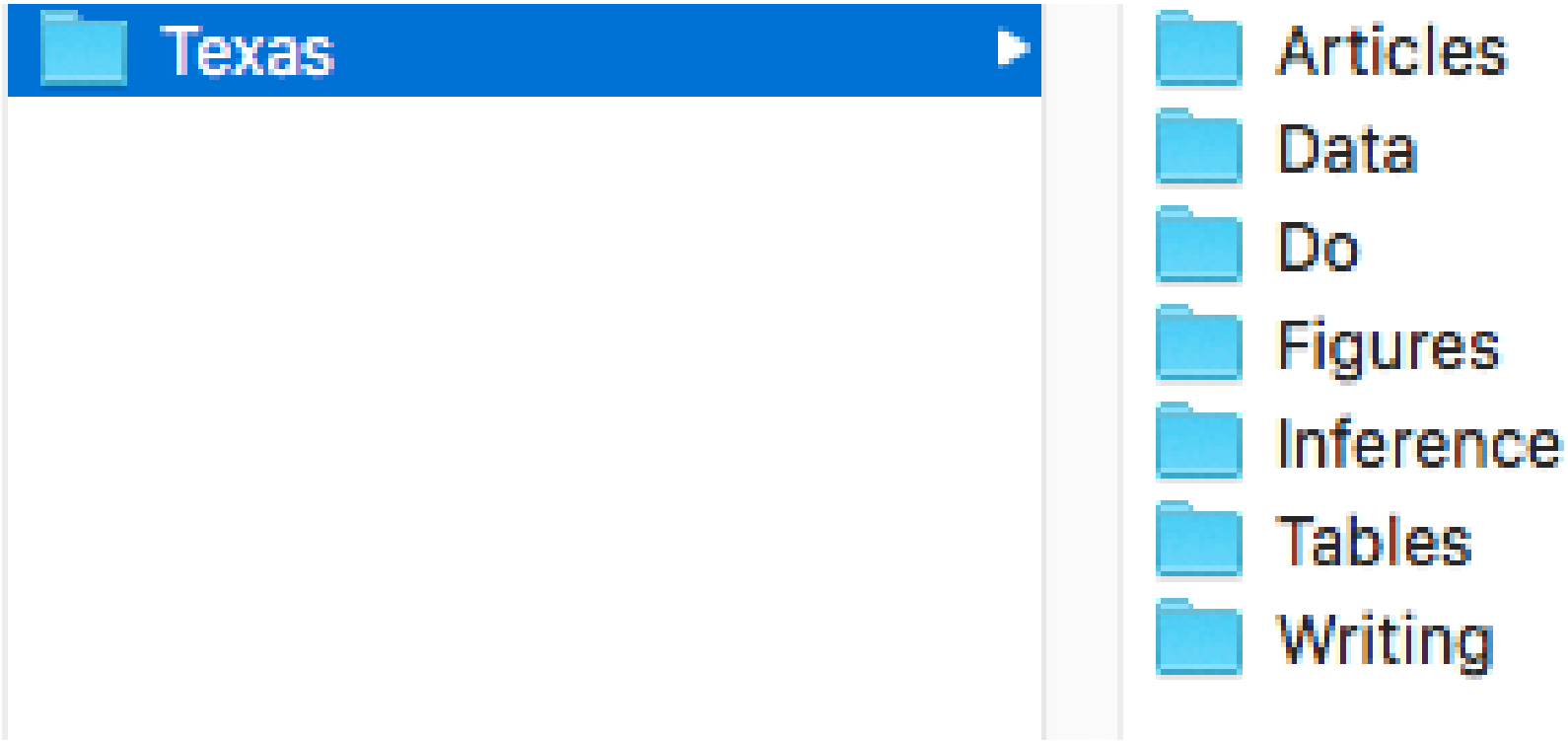
Finding files using folder structure

- Directory structure is a “hierarchical system of folders that modern computer operating systems use to arrange files” – Chin (2021)
- Directory structure connotes physical placement as in a file stored on a computer is *somewhere* on that computer in a specific and discrete location
- But it’s very easy in projects that takes years to accumulate so many files that you just give up and everything ends up in one directory
- Mental models have been changing (educators noticed it in 2017) as newer students tended to use more cloud-based storage, smartphone apps, and various forms of search, rather than hierarchical folders for organization
- Need to stick with hierarchical folder structure as much as possible

Directories

- But even well designed structure can be undermined by:
  1. Time: Multi-year projects lead to more files, and weird ones that don't quite fit in your pre-designed folders
  2. Files: More data you collect, the harder it becomes to store in logical places
  3. Coauthors: Because everyone tends to do things differently, shared dropbox with collaboration can create messes
- Hierarchical folder structures require constant vigilance because efficiency and accuracy both decrease with time, files and collaboration
- Consider the following relatively simple structure as a conceptual model only

Subdirectory organization

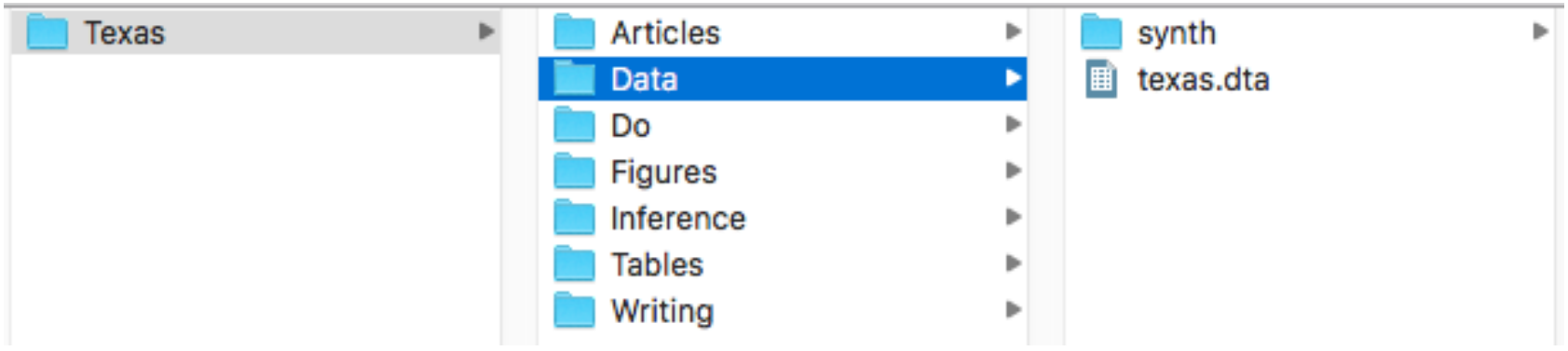


1) Name the project (“Texas”)

Subdirectory organization

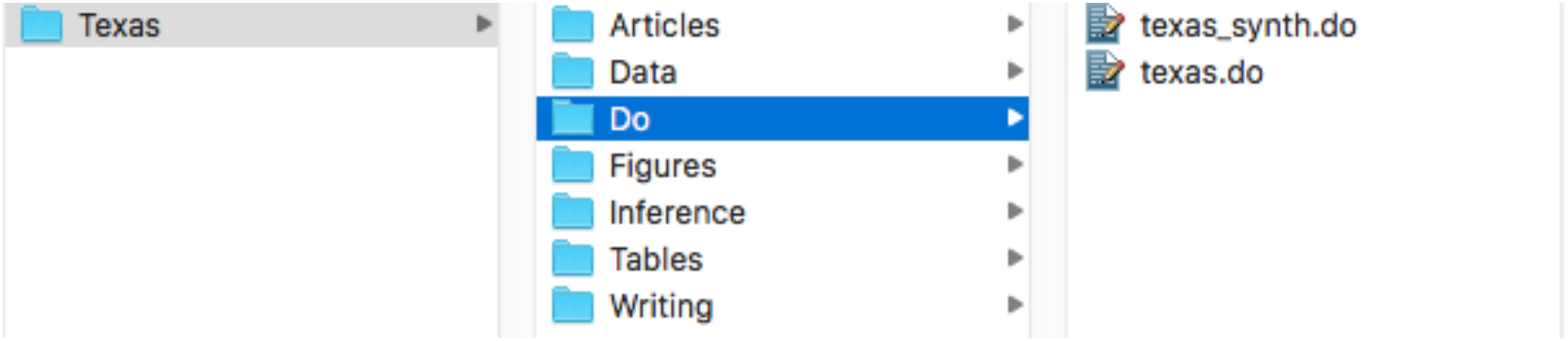
2) A subdirectory for all articles you cite in the paper

Subdirectory organization



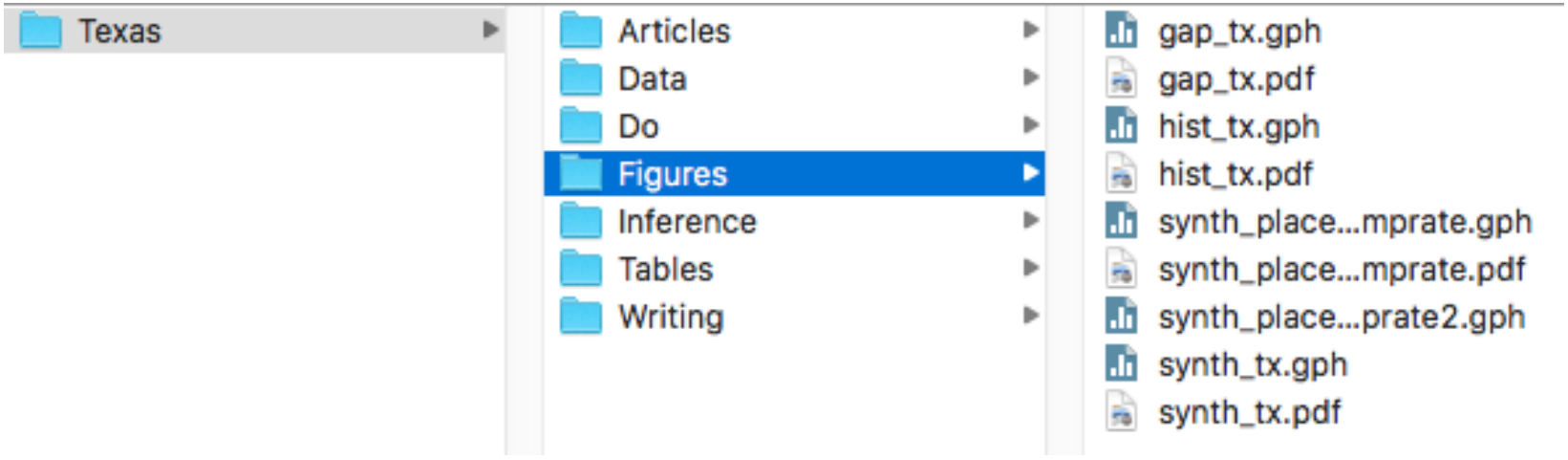
3) Data subdirectory containing all datasets

Subdirectory organization



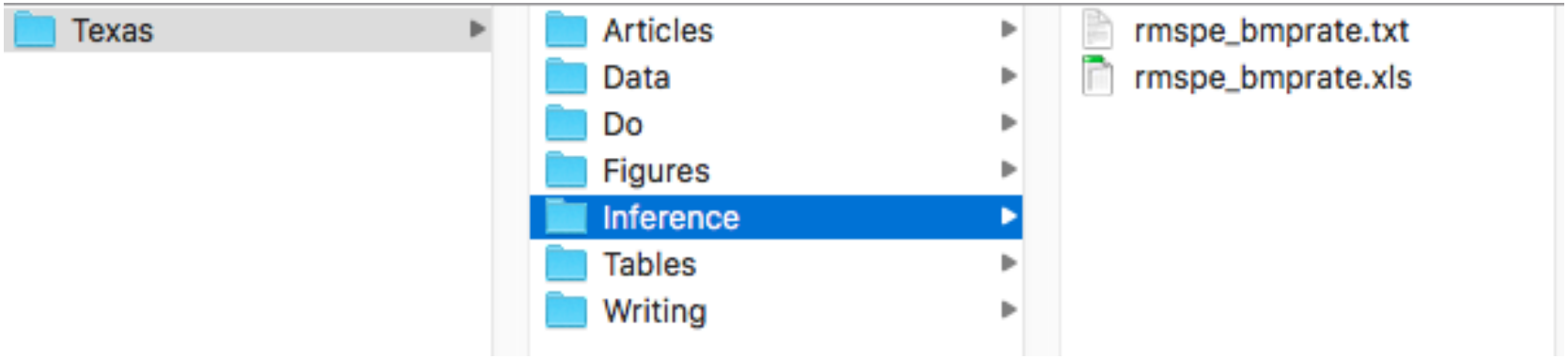
4) A subdirectory for all do files and log files

Subdirectory organization



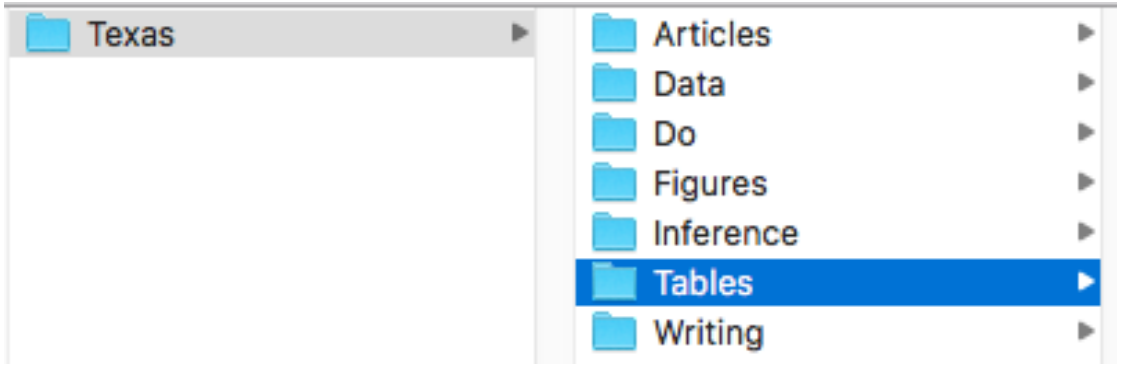
5) All figures produced by Stata or image files

Subdirectory organization



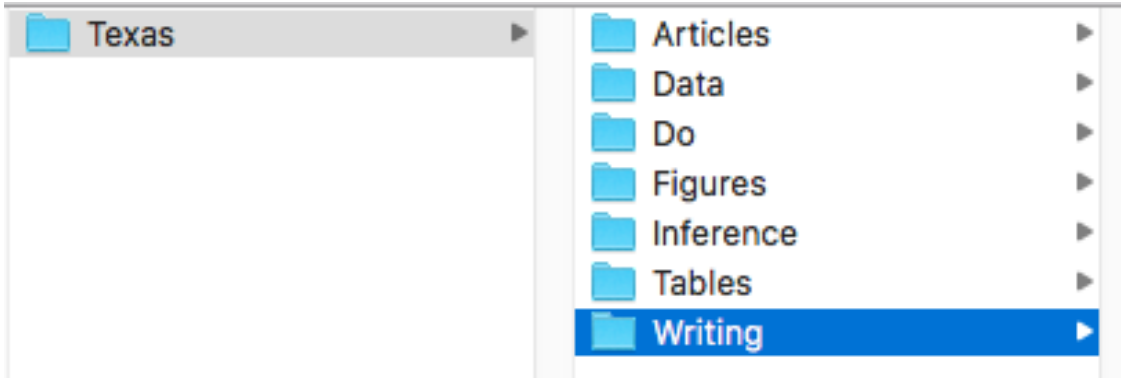
6) Project-specific heterogeneity (e.g., “Inference”, “Grants”, “Interview notes”, “Presentations”, “Misc”)

Subdirectory organization



7) All tables generated by Stata (e.g., .tex tables produced by -estout-)

Subdirectory organization



8) A subdirectory reserved only for writing

Automation

- A second problem I had was that I was copying and pasting information
  1. hierarchical folder structure
  2. **automation**
  3. naming conventions
  4. version control
- Automation does not allow errors to creep in, whereas hand copying does

Code not Command Line

- Your future self doesn't remember making any of these tables or figures
- All tables and figures must be replicable using a scripting file, not the user interface or command line
  - → It's fine to use the command line
  - → But it must eventually go into the program
- Final output must be produced by running the entire code, not just chunks

Beautiful code

- Your ideal goal is to make beautiful code which is easier for some than others
- At minimum, don't make it ugly – if your future self can't read it, it's ugly or it's confusing but both are bad
- Consider a new text editor like Visual Studio Code which allows for colored syntax, indentation, column editing and bundles
- Stata and Rstudio also come with built-in text editors which are fine

Headers

```
*************************************************************
* name: texas.do
* author: scott cunningham (baylor university)
* description: estimates the causal effect of prison capacity
*              expansion on incarceration rates using synth
* date: march 19, 2018
*************************************************************
```

Speak slowly in your programs

“Be conservative in what you do; be liberal in what you accept from others.” - Jon Postel

- Smart sounding quote about both programming and relationships
- Your future self is time constrained, so explain *everything* to her as well as write clear code
- Optimally document your programs
- But speak your future self’s love language so she understands

Automating tables

- Your goal is to make “beautiful tables” that are never edited post-production as well as readable on their own
- Large fixed costs learning commands like -estout- or -outreg2-: incur them bc marginal costs are zero
- I use -estout- because Jann has written an excellent help file at `http://repec.org/bocode/e/estout/hlp_esttab.html` but many like -outreg2-
- I’ve uploaded code in Stata that will automate some simple tables

Automate figures

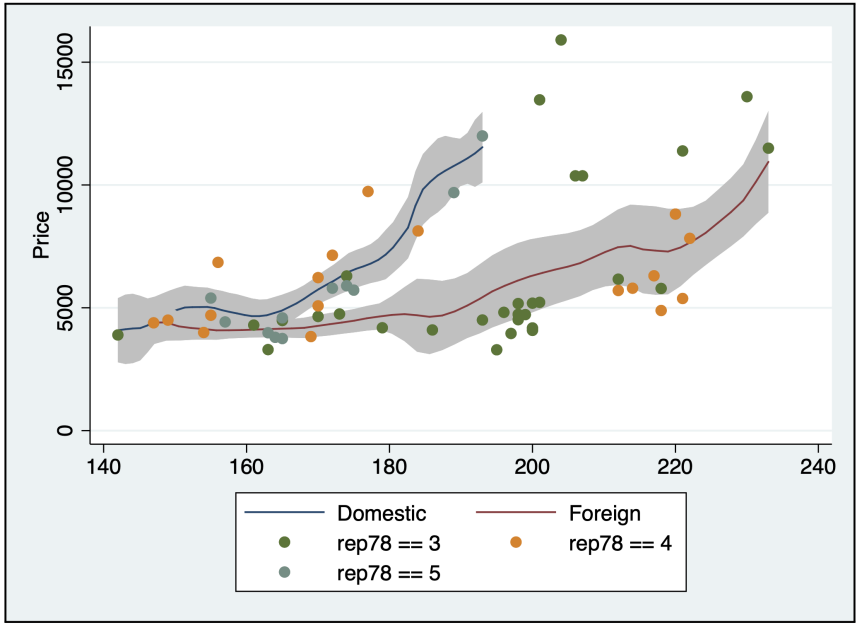
- Again goal is replication, accuracy and efficiency
- If you're doing something a few times, learn to automate it
- Learn how to make customizable graphs using automation, not by tinkering in post-production
- Examples include Stata's -twoway-, R's -ggplot2-, Python's -matplotlib- and its various wrappers

Data visualization resources

Study other people's pictures and get help from experts

1. Kieran Healy's 2018 Visualization: A Practical Introduction (Princeton University Press); free version is `http://socviz.co/index.html#preface`.
2. Ed Tufte's book Visual display of quantitative information is classic, but more a coffee table book plus no programming assistance.
3. Ben Jann deck for making beautiful graphs in Stata `https://www.stata.com/meeting/uk18/slides/uk18_Jann.pdf`

```
. set scheme s2color
. two (lpolyci pfor length, clstyle(p1line))
>     (lpolyci pdom length, clstyle(p2line))
>     (scatter price? length, pstyle(p3 p4 p5))
>     , ytitle(Price) legend(order(2 "Domestic"
>         4 "Foreign" 5 6 7))
```



Name stuff well

1. hierarchical folder structure
2. automation
3. **naming conventions**
4. version control

Different elements

- Three different things you must name:
  1. variables,
  2. datasets
  3. programs
- You do these three things repeatedly therefore you need a system otherwise errors will occur due to the repetition
- Anything that can happen will happen with enough trials. – Murphy's law

Naming conventions for variables

- Variables should be readable to a stranger
  - → Say that you want to create the product of two variables. Name it the two variables with an underscore
  - → `gen price_mpg = price * mpg`
- Otherwise name the variable exactly what it is
  - → gen bmi = weight / (height^2 * 703)
- Avoid meaningless words (e.g., `lmb2`), dating (e.g., `temp05012020`) and numbering (e.g., `outcome25`) as your future self will be confused

Naming datasets and do files

- The overarching goal is always to name things so that a stranger seeing them can know what they are
- One day you will be the stranger on your own project! Make it easy on your future self!
- Choose some combination of simplicity and clarity but whatever you do, be consistent
- Avoid numbering datasets unless the numbers correspond to some meaningful thing, like randomization inference where each file is a set of coefficients and numbered according to FIPS index

Version control

1. hierarchical folder structure
2. automation
3. naming conventions
4. **version control**

Version control

- “Version control” is the practice of tracking and managing changes to software code and documentation
- Old school version control was to use a suffix attached to the filename, maybe with date and author abbreviation (`Paper_Revision_V3_SC_May052021.tex`)
- Creates dozens and maybe more local versions which depending on the project may make management difficult and therefore allow for errors
- As software production became more collaborative and complex, new version control systems were created so that we didn’t have to use the date/author suffix system
- Git is the most popular version control sytem

What is git?

- Open source project developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system (also open source)
- Since 2005, Junio Hamano has been the core maintainer
- Many many many projects rely on git for version control, including commercial projects
- Git is “distributed architecture”, like “deep” software (e.g., operating system, programming language), installed on your local machine
- Used to coordinate production, editing and maintenance of software
- Nice because it allows for simultaneous editing by multiple sources, and reconciling differing versions, as well as distributing those versions back to people

Git and Github

- Git and Github are related but distinct
- **Git** is a mashup between Dropbox and Microsoft “track changes” (credit: Grant McDermott)
- **GitHub** is a company that provides services that are built on git
- **Imperfect analogy**: Github is to git what beer bottles are to beer (not exactly, but almost)

Github repository

- Think of it for now as like your Dropbox folder in that it’s like a folder stored in the GitHub servers
- A repository contains all of your project’s files and each file’s revision history.
- You can discuss and manage your project’s work within the repository.
- It tracks all changes made to files in your project, building a history over time.
- If you delete the .git/ folder, you have deleted your project’s history.

Getting started

- Set up git (steps not shown): `https://docs.github.com/en/get-started/quickstart/set-up-git`
- Create a free account at github (steps not shown): `https://github.com`
- Download github desktop (next)
- Set local path (next) – do *not* put it in dropbox or cloud based directory. Put it in something like /Documents or /Desktop
- Clone a repo (next)
- Go through git operations (next)

The Four Git Operations

1. **Staging**: tell git you want to add changes to the repo history
2. **Commit**: tell git you are sure that you want this particular thing you did to be part of the repo history
3. **Pull**: tell git to get any new changes made on the github repo and put it on your local machine (hence “pull”)
4. **Push**: push any local stuff in your .git/folder to the github repo which allows then anyone else to pull it

Create your own repo

- Use Github Desktop to create a tutorial repo called /articles
- Put the Iyengar 2009 article in this directory
- Stage, commit and push
- Go to github and find your repo and upload Chin and Cunningham 2019
- Pull from Github Desktop

Good and bad projects take the same amount of time

- The goal is not to explore the data; the goal is to create useful knowledge
- "All you can do is write the best paper on the question you're studying" – Mark Hoekstra
  - → Note he didn't say "Write the best paper you're capable of writing"
  - → He said **the best paper**
  - → Important therefore to choose the right questions ahead of time
- Slow down, think big picture, figure out exactly what your question is, and let that guide who is in your sample (and importantly who won't be), what time periods you'll pull, etc.

Selling your work

- If you don’t advocate for your work, *no one will*.
- Network, network, network
- You will need to become an expert in 1.5 areas, and you will need experts in those 1.5 areas to agree
- Study the effective of rhetoric of successful economists who expertly communicate their work to others both in their writing of the actual manuscript, as well as the presentation and promotion of their work

Find your mentors and sponsors

- Working with senior people at some point becomes necessary
- Good news: many senior people want to help you
- Bad news: they don’t know who you are and can’t find you
- It’s a two sided matching problem

Finding them

- Introduce yourself in socially appropriate ways!
- This is likely a friction for URM and females given their under-representation in the profession, and prejudices more generally
- Find allies; work networks; I'll be a resource if you need me

Al Roth story

- I wrote Al Roth in 2007 and like Robert Browning to Elizabeth Barrett introduced myself by saying “I love your book on twosided matching with Sotomayor with all my heart.”
- We became pen pals and then he won the Nobel Prize
- Scared, I wrote to congratulate him on the day he won and he immediately asked to help me
- “Interpersonal favors are meant to be paid forward not backwards” - Roth to me after a *second* favor!
- Nobody can help you if you don’t know them bc help, sponsorship and mentoring is a two sided matching problem

More readings

- I've put several deck of slides and helpful articles for you in the dropbox folder
- Jesse Shapiro's "How to Present an Applied Micro Paper"
- Gentzkow and Shapiro's coding practices manual
- Rachael Meager on presenting as an academic
- Ljubica "LJ" Ristovska's language agnostic guide to programming for economists
- Grant McDermott on Version Control using Github
  https://raw.githack.com/uo-ec607/lectures/master/02-git/02-Git.html#1