Practical Data Analysis Tips

ECON 490

Taylor Mackay || Email: tmackay@fullerton.edu

Don’t Reinvent the Wheel

Data analysis activities and R handouts are all written very intentionally

- Goal of these activities is to prepare you for capstone analysis!
- Most coding issues/questions I get deal with material covered in activities

Review old coding activities to remind yourself what you know how to do!

If you’re using R (and not proficient with it) pull out copies of the following:

- Introduction to Tidyverse handout
- Regression Review Pt. II

A Gentle Bit of Advice (Pt. 1)

Students often ask if they can use Excel/ Python/whatever for their project

- In general, yes, you can (especially if you use it for work)
- Doing so means I can’t directly share code (but tradeoffs might be worth it)

In my experience, students who have difficulty using R also tend to have difficulty with other programs

Disclaimer: If you can regularly/confidently use PivotTables or Excel macros, this doesn’t apply to you! More generally, if you’ve had a “Neo in the Matrix” moment

A Gentle Bit of Advice (Pt. 2)

We are using R at a relatively basic, entry level

It is very likely that if you are having trouble, the issue isn't with using R specifically

- Instead, it's often a conceptual issue – can you precisely state your goal?
- Does the regression you're trying to run make sense?
- Does the data set you're trying to put together let you run that regression?

If your goal isn't clear, and you swap to Python/Excel… you still don't have a clear goal

My advice: 1) get clear on your goals and 2) make sure you're leveraging class resources

Handling Data (and Avoiding Headaches)

General project workflow looks something like:

- Find some data set(s) online, download them – this is your ***raw data***
- Clean up/process your raw data to create your ***working data***

KEY POINT: You should always retain an ***unmodified*** version of your raw data

If you need to change your project in the future, and you can't recreate your working data set from raw data, you can wind up totally stuck

- If you're using Excel, don't just start editing data you've downloaded!
- Generally, this is less of a concern when using R/Python

Data Processing Workflow

Processing/cleaning your data means filtering rows, selecting columns, etc.

- You should have a clear record of how you're doing this
- Use .R script/code files – don't just run things in the console!
- If you're using Excel, use comments or a separate text file describing process

KEY QUESTION: *Ask yourself as you're working, "If my computer died/rebooted right now, how screwed would I be?"*

If you'd lose all your project progress if your computer died, something is wrong

- All code/data should be backed up remotely via Dropbox/iCloud/Drive, etc.

Where to Find Data

Easy places to start:

1) Capstone Data Resources page on ECON 490 Canvas page
    - Scroll down to External Data Sets for links sorted by topic area

2) IPUMS ACS/CPS websites for individual-level survey data
    - Use variable search feature to get a sense of what's available

As you look at new data, ask yourself, "What correlations/regressions are interesting?"

- If you can't think of something (relatively) quickly, move to another data source

***WARNING:** ChatGPT is horrible with questions like, "What data sets should I use?"*

- 5.0 combines overconfidence with high error rate = lots of potential time lost
- Claude/Gemini are better (more conservative) but this one area where AI still struggles

Build the Plane While You're Flying

If a project idea isn't going to work, you want to know sooner rather than later

- Try to define and run a "minimum viable" regression to test your ideas
- Start by doing just enough data collection to run that test regression

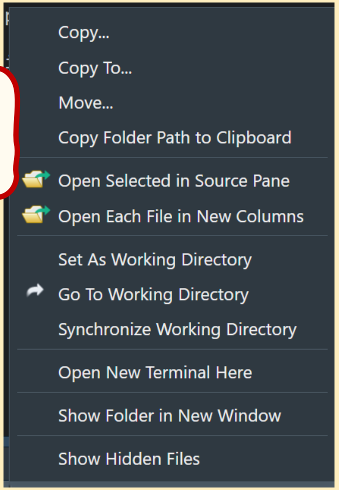
E.g., do you plan to collect 10 years of data? Try running a regression with 2 years first.

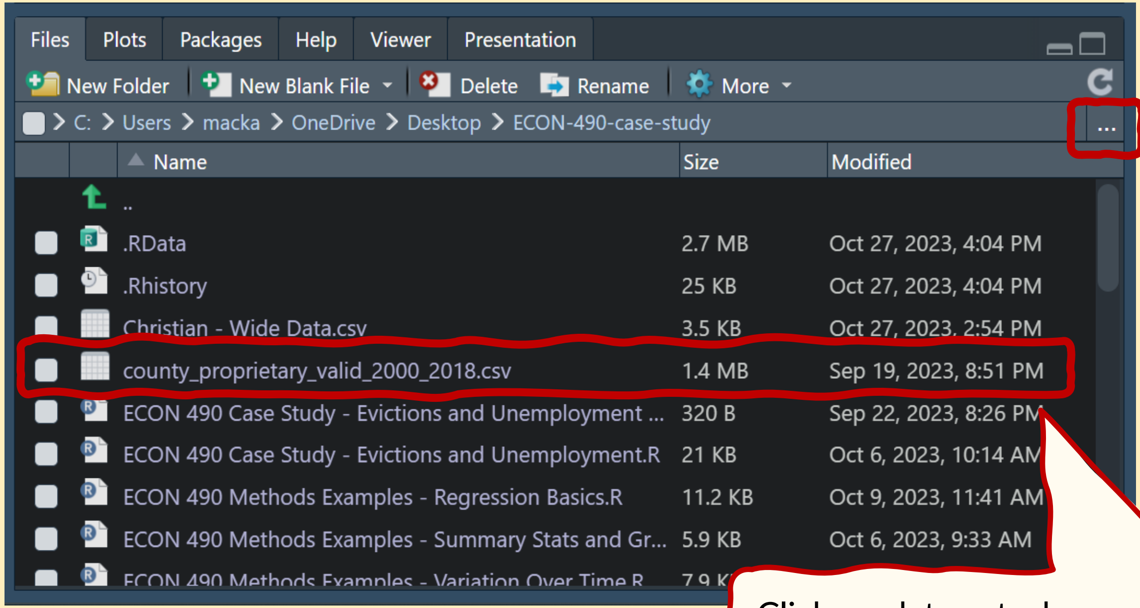
If test models don't work, you can adjust goals as you go

- Once you've test things, you can then collect more data
- ***Rule of thumb:*** *More data is better… once you have a clear plan/goal*

Loading Data into R

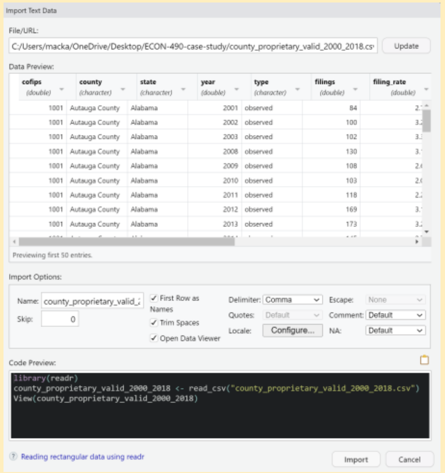
Click …, load folder with your data and set working directory


Copy…
Copy To…
Move…
Copy Folder Path to Clipboard
Open Selected in Source Pane
Open Each File in New Columns
Set As Working Directory
Go To Working Directory
Synchronize Working Directory
Open New Terminal Here
Show Folder in New Window
Show Hidden Files



C: > Users > macka > OneDrive > Desktop > ECON-490-case-study

Name	Size	Modified
..		
.RData	2.7 MB	Oct 27, 2023, 4:04 PM
.Rhistory	25 KB	Oct 27, 2023, 4:04 PM
Christian - Wide Data.csv	3.5 KB	Oct 27, 2023, 2:54 PM
county_proprietary_valid_2000_2018.csv	1.4 MB	Sep 19, 2023, 8:51 PM
ECON 490 Case Study - Evictions and Unemployment ...	320 B	Sep 22, 2023, 8:26 PM
ECON 490 Case Study - Evictions and Unemployment.R	21 KB	Oct 6, 2023, 10:14 AM
ECON 490 Methods Examples - Regression Basics.R	11.2 KB	Oct 9, 2023, 11:41 AM
ECON 490 Methods Examples - Summary Stats and Gr...	5.9 KB	Oct 6, 2023, 9:33 AM
ECON 490 Methods Examples - Variation Over Time.R	7.9 K	


Import Text Data

File/URL: C:/Users/macka/OneDrive/Desktop/ECON-490-case-study/county_proprietary_valid_2000_2018.csv

Data Preview:

cofips (double)	county (character)	state (character)	year (double)	type (character)	filings (double)	filing_rate (double)
1001	Autauga County	Alabama	2001	observed	84	2.
1001	Autauga County	Alabama	2002	observed	100	3.
1001	Autauga County	Alabama	2003	observed	102	3.
1001	Autauga County	Alabama	2008	observed	130	3.
1001	Autauga County	Alabama	2009	observed	108	2.
1001	Autauga County	Alabama	2010	observed	103	2.
1001	Autauga County	Alabama	2011	observed	118	2.
1001	Autauga County	Alabama	2012	observed	169	3.
1001	Autauga County	Alabama	2013	observed	173	3.

Previewing first 50 entries.

Import Options:

Name: county_proprietary_valid_2
Skip: 0
✓ First Row as Names
✓ Trim Spaces
✓ Open Data Viewer
Delimiter: Comma
Quotes: Default
Locale: Configure...
Escape: None
Comment: Default
NA: Default

Code Preview:

```
library(readr)
county_proprietary_valid_2000_2018 <- read_csv("county_proprietary_valid_2000_2018.csv")
View(county_proprietary_valid_2000_2018)
```

Click on data set, choose “Import Data,” to preview & load data

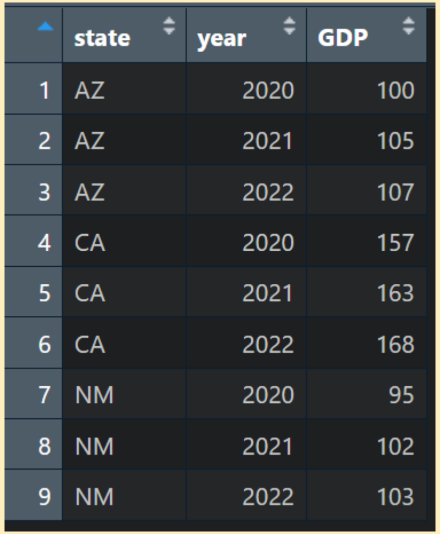
Wide vs. Long-Formatted Data

Two types of data structures:

- ***Wide*** data has same variable in multiple columns (here, GDP is split by year)
- ***Long*** data has each variable in one column

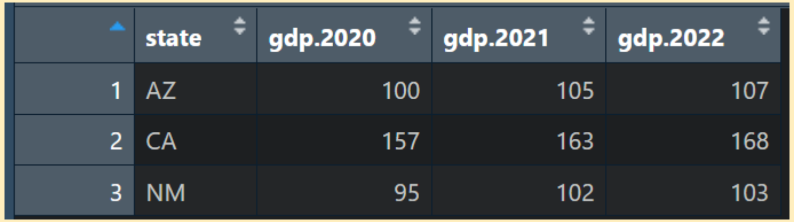
In almost all cases, data should be in long format for running regressions

- In R, use `pivot_longer()`



	state	year	GDP
1	AZ	2020	100
2	AZ	2021	105
3	AZ	2022	107
4	CA	2020	157
5	CA	2021	163
6	CA	2022	168
7	NM	2020	95
8	NM	2021	102
9	NM	2022	103

Long-formatted data



	state	gdp.2020	gdp.2021	gdp.2022
1	AZ	100	105	107
2	CA	157	163	168
3	NM	95	102	103

Wide-formatted data

Combining Data Sets

Merging data == combining data sets – why do we need to do this?

- For your projects (and most data analysis), you'll get data from ***multiple*** sources
- You combine data sets by ***merging or joining*** them together

Examples include combining:

- State-level data with individual-level survey responses
- Location-based crime data with local area demographic data
- NBA player performance data with salary and contract records

The Shape of Data Sets

When you combine two data sets, your data expands ***horizontally***

- You started with some set of columns/variables, then added new columns
- Different variables across data sets

Sometimes, you have data that you want to ***stack or append***

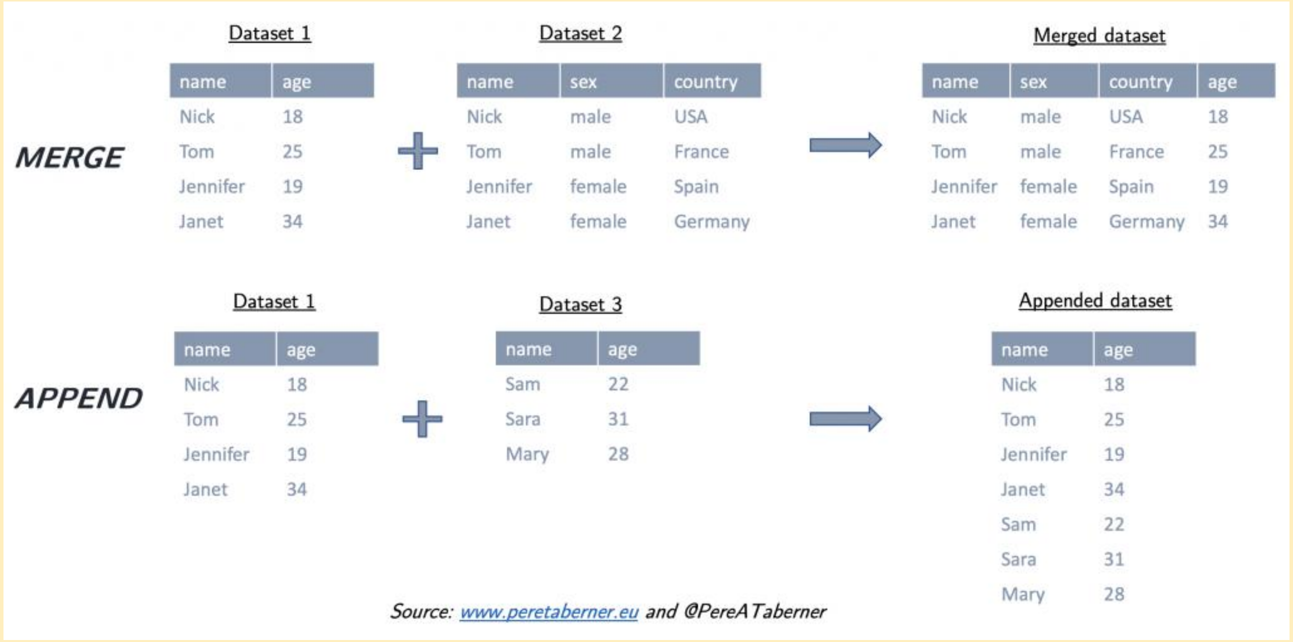
- Example: county-level pollution data for CA in 2020 and 2021
- ***Same*** variables in **both** data sets → append data to get panel with both years

Appending data expands your data ***vertically***

- Depending on context, merges might also, but main goal = new columns

Appending Data in R

In screenshot below, use `bind_rows()` function to ***append*** data sets 1 & 3



name	age
Nick	18
Tom	25
Jennifer	19
Janet	34

name	sex	country
Nick	male	USA
Tom	male	France
Jennifer	female	Spain
Janet	female	Germany

name	sex	country	age
Nick	male	USA	18
Tom	male	France	25
Jennifer	female	Spain	19
Janet	female	Germany	34

name	age
Nick	18
Tom	25
Jennifer	19
Janet	34

name	age
Sam	22
Sara	31
Mary	28

name	age
Nick	18
Tom	25
Jennifer	19
Janet	34
Sam	22
Sara	31
Mary	28

Source: www.peretaberner.eu and @PereATaberner

Before You Combine Data

Before you combine data sets, you need to understand ***data structure***

- Within both data sets, what variables uniquely identify each row?

Across both data sets, what variables will you use to link data?

- These are your “key” or “by” variables
- What is the shared structure of both data sets? Defined by ***less*** granular data

Example: Combining individual-level CPS with state-level home price data

- While CPS has individual identifiers, the ***shared structure*** is state and year, so these are the key or by variables (state-level data is less granular)

Preparing Data for Merging

Once you've identified your key variables, make sure they're ***consistent***

- Across both data sets, is each variable ***same format*** (i.e., factor, string, etc.?)
- Are spellings consistent? I.e., check "California" vs. "california" vs. "CA"

If necessary, create new, consistent versions of key variables using `mutate()`

Check you don't have duplicates across key variables

- E.g., in state-by-year ACS data, suppose you found 2 Arizona-in-2010 rows
- Drop extra row using `filter()` or combine via `group_by() + summarize()`

Join Functions in R

Given two data sets, you'll have one of the following types of merges or joins:

- ***1-to-1:*** Each observation in 1st dataset matches exactly one row in 2nd
- ***Many-to-1:*** Multiple observations in 1st dataset match to same row in 2nd
- ***Many-to-Many:*** Multiple rows match to multiple rows → avoid doing this!

Most useful join functions in R for your projects:

- `inner_join():` Keep ***only*** matched rows in ***both*** data sets
- `left_join():` Keep all rows from 1st data set and all matches from 2nd
- `full_join():` Keep all rows from ***both*** data sets

After Merging Data

The most important thing to do is “sanity check” your new merged data set

- Check the number of rows in the matched data set using `nrow()`
- Does it match what you expected?

Check NAs and summary statistics for important variables using `summary()`

Example: Combine state-level crime data with state-level ACS via `inner_join()`

- This keeps matched rows, so you should have rows for 51 states (counting DC) multiplied by 5 years; if you don’t, something went wrong!

Join Examples

Inputs

Join Functions

Outputs

DF1

ID	Value
A	123
B	769
C	475
D	978

DF2

ID	Level
A	Red
B	Blue
E	Green
F	Yellow

```
inner_join(DF1, DF2,
     by = “ID”)
```

ID	Value	Level
A	123	Red
B	769	Blue

```
left_join(DF1, DF2,
    by = “ID”)
```

ID	Value	Level
A	123	Red
B	769	Blue
C	475	*NA*
D	978	*NA*

Who’s in my Regression Sample?

Key question before running any regression – who gets included?

- Sometimes, this answer is straightforward (esp. with state or county data)
- Matters a lot with individual data like the CPS (or business-level data, etc.)

Data sets like the CPS have kids/retired folks included

- You might not want them in a regression exploring union membership!
- Use the `filter()` function to restrict sample to observations you want

Describing sample restrictions is a ***key*** part of interpreting regression output

Outliers and “Weird” Observations

Always check your data with `summary()` and / or `table()`

- This lets you catch values that don’t look right
- Do you see outliers or repeated instances of the same random value?

Dealing with these situations requires background knowledge about your data

- Are extreme values feasible or plausible for a given variable?
- Do you have unusually coded missing values (e.g., -99, 999, 1000)?

As a general rule, be careful removing outliers if you think they’re real

- If you remove values or rows, how does this change regression interpretation?

R Code Example

R code file shows examples of using summary() and table() to check variables, then describes top-coding for income data

Splitting Your Data

In general, your working data should be a ***single*** data set

- You can use `filter()` to explore a subsample of people in your data…
- But always make sure you're clear on ***why*** you're doing this!

General rule – splitting your data and running "symmetric" regressions probably shouldn't be your first approach

Let's say you have state-year data on gas prices and car sales for 2023 and 2024

- Your default strategy should be to use ***all*** data in `lm(sales ~ prices)`
- Worried about trends in both variables? Start by including `as.factor(year)`

R Code Example

R code file shows an example of splitting your sample using filter(), then shows how to run an interaction regression as an alternative to splitting your data by group (in the example, by state)

R-Squared (aka R^2)

Economic outcomes are high-dimensional (lots of stuff matters!)

R^2 tells us how much of the variation in Y we can explain with our Xs

- This matters if we want to predict Y…
- But less important if we're interested in assessing how a specific X impacts Y

For most descriptive projects, it's easy to increase R^2 without doing anything substantively interesting

- Try this yourself – add FEs, other X variables, etc. and see what happens
- Your R^2 might go up… does the answer to your research question change?

R Code Example

R code file shows an example of how adding FE's can change both your regression interpretation and R2

“Regression-Level” Things to Include in Interpretation

We’ve talked about interpreting βs – “regression-level” context matters too

Things to include when interpreting regression output:

- Sample restrictions *(Is anyone excluded?)*
- Missing values *(Are there people you couldn’t include? Why are they missing?)*
- Outliers *(Did you remove anyone for being an outlier? Why were they an outlier?)*

Things you don’t need to worry about (unless you’re doing prediction):

- R^2, other model-level “goodness of fit” tests

Is Doing X Okay?

A common question I get is, “Is it okay if I just do X for my 2nd stage”

- This is very hard to know in advance
- General theme from prior slides – explore variation/find patterns/etc.

These are all ways of saying, “Explore your data”

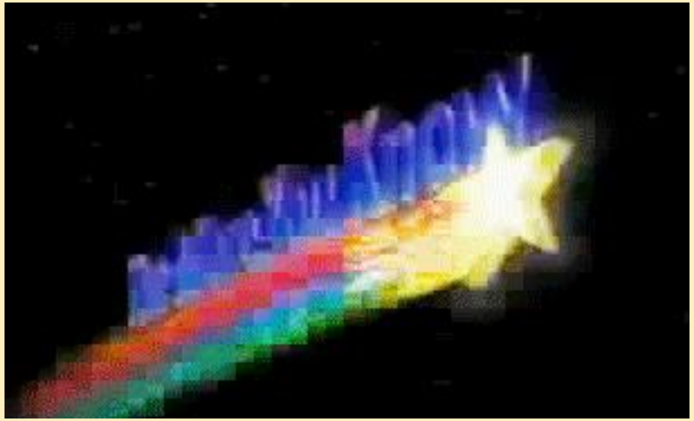
- If you’re doing a descriptive project, you should be looking at a lot of results
- Try different conditional summary stats, regressions, graphs, etc.

Try running at least 3-4 variations of any regression – what differences matter?

Disclaimer: Technically, in some contexts (esp. causal inference), we want to avoid this (it’s known as multiple testing). For this class, however, the goal is to learn how to explore data and use regression. The best way to do that is (not surprisingly) exploring data and running lots of regressions!

The Next Slide is IMPORTANT

Is everyone paying attention?



Things You Should Absolutely Remember

Even if you forget everything else from tonight, remember the following:

1. Keep clean, separate, ***unedited*** versions of ***all*** raw data
2. Write code somewhere that you can save/back up (e.g., R scripts, Colab, etc.)
3. Check variable definitions for missing values
4. Always visually/manually check output when you combine/merge data

ASK YOURSELF: *What happens if my computer reboots right now? Am I okay?*

- Back all data and code up via Google Drive/Dropbox/OneDrive

Bonus Tips: Make sure your data is long-formatted before running regressions. Also, make sure your R code runs start-to-finish (don't just run chunks randomly)