Big Data in Spatial Economics

Lecture 2: Working with Big Data

Victor Couture (UC Berkeley) and Allison Green (Princeton)

with special thanks to Jonathan Dingel and Nick Tsivanidis

Prepared for PhD289 at UC Berkeley and ECON720 at Yale University.

Fall 2019

Lecture Plan

- Lecture 1: Overview of Current and Future Research
  1. Literature review
  2. A detailed example: Smartphone GPS data in Couture, Dingel, Green, and Handbury.
  3. How to find a big data research question.
  4. Ethics of big data

- Lecture 2: Working with Big Data
  1. Basic principles
  2. `Stata` for big data
  3. Alternatives to `Stata` for really big data (`SQL`, `Python`, `Julia`)
  4. Machine learning for high dimensional/complex data

Lecture 2: Objectives

- Learn some general principles of working with big data
  - E.g., Break down your big data into smaller chunks
- Learn how and when to use `Stata` with big data.
  - Some commands and packages are faster than others.
- Recognize when alternatives to `Stata` are desirable.
  - e.g., `SQL` to clean and assemble smartphone data
- Recognize when alternatives to our traditional econometric framework are desirable
  - e.g., Machine learning for imagine recognition

Working with Big Data: Basic Principles

1. Take time to organize your work flow
2. Store data efficiently (skinny & relational)
3. Break down big data into smaller chunks
4. Analyze data efficiently (use server & parallelize)
5. Test your code before running large data tasks
6. Run code efficiently (build tools)

Working with Big Data: Basic Principles

Take time to organize your work flow

- Gentzkow and Shapiro, *Code and Data for the Social Sciences. A Practitioner's Guide.*
  - Must read for empirical economists and their RAs.
  - Brings some best practices from data science and computer science to economics.

- https://github.com/jdingel/projecttemplate
  - Full research template that we use for smartphone project, from Jonathan Dingel.
  - Lots of links to sources to train yourself or your RA.
  - Focuses on tools we use (Bitbucket for version control, Asana for task management, etc.)

Working with Big Data: Basic Principles

Store data efficiently (skinny + relational)

- Keep data skinny:
  - Import only variables and observations needed
  - Use efficient data format (e.g., bytes instead of double for dummies.)
  - Clean before generating new variables.

- Keep a relational database:
  - Each dataset has a unique identifer.
  - E.g., smartphone relational database consists of:
    1. device dataset with unique device id
    2. place dataset with unique place id
    3. visit dataset relating device and place

Working with Big Data: Basic Principles

Break down your big data into smaller chunks

- Most important rule of big data.
  - High computation times are especially unproductive when code testing.
  - *Test your code on a small sample*.

- Choose a natural partition for the type of operations you are trying to do
  - Spatial data can often be broken down in smaller geographical units.
  - 200 city level estimations often faster than 1 national estimation.
  - Many statistical procedures are non linear in sample size.

Working with Big Data: Basic Principles

Analyze data efficiently (use server + parallelize)

- Your university likely has a server that is faster than your laptop.
- Some languages and commands are faster than others (more on that below.)
- Parallelize whenever possible
  - Bash scripts can automatically submit jobs when queues become available
  - Think carefully about the best dimension to split your code on for parallelizing
  - E.g., Your server has 5 nodes, run 40 city-level regressions on each $\rightarrow$ 5X faster.
  - Most universities have a limit on the amount of jobs you can run simultaneously
  - Check your university's documentation for example scripts to run parallel bash scripts.

Working with Big Data: Basic Principles

Test your code before running large data tasks:

- When working on servers it is best practice to know how much computing resources large data tasks will take up
- Likely you are sharing resources with your colleagues!
- Benchmark different approaches to the same problem against each other
- Check how your computational efficiency scales as data size increases

Working with Big Data: Basic Principles

Run code efficiently (build tools) *Special thanks to Jonathan Dingel for slide material*

- Research projects are collections of code executed in sequence to produce output.
  - Your code has a first step (download raw data) and a last step (generate paper PDF).
  - A "master" do file in `Stata` is a simple way to automate this.
  - If a project involves A through Z, master.do executes A, B, . . . , Y, Z in order.
- The "run everything" master is inefficient: if you edit Y, you only need to run Y and Z.
  - Do not run time-consuming data cleaning steps for every new estimation.
- Software build tools automate the choice of what to run.
  - Build tools use a dependency graph and information about file changes (e.g., timestamps) to produce output using (all and only) necessary steps.
  - In an appendix, we describe `Make`, the build tool that we use for the smartphone project.

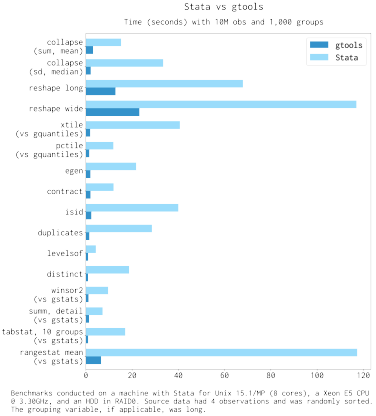
Working with Big Data in Stata

- Stata is the most common language in empirical economics
  - Good for co-authoring, all our co-authors know Stata
  - Stata has lots of built in commands for econometrics.
  - Stata MP on good server can handle big (not huge) data.

- Avoid computationally heavy commands on large datasets
  - Many popular commands in Stata are slow because they rely on underlying sort.
  - E.g., “collapse”, ”reshape”, ”sort” , and ”egen”
  - User-written Stata packages such as gtools offer faster alternatives

- Choose well-suited regression commands
  - ”reghdfe” by Sergio Correia, is often the best reg option
  - ”asclogit” for logit models

Gtools vs. Generic Stata Commands



Source: https://gtools.readthedocs.io/en/latest/

Alternatives to Stata

- `Stata` is not well-suited to perform certain tasks.
- `SQL` for cleaning and assembling *really* big data.
- `Python` for ArcGis or machine learning or scraping or anything you can't do in `Stata`.
- `Matlab` or `Julia` for coding estimators or structural models.
- Many other languages that we haven't used for our projects (e.g., `R` can do ArcGis, machine learning, estimation, etc.)

Working with *Really* Big Data: Data Cleaning and Assembling

Problem: Smartphone data has over a trillion raw visits.

- `Stata MP` on best current computer can take 10 to 20 Billion obs.
- Breaking down data is not enough.

Solution:

- `SQL` well-suited for data cleaning and assembling.
- `Python` offer similar options through Panda library.
- In smartphone project we use `SQL` to clean, merge and create estimation tables:
  - E.g., assign home location to each device
  - E.g., remove near simultaneous visits to same location
  - E.g., aggregate data at monthly level
- `SQL` has limitations, and cannot produce regressions or figures.
  - Need to export to another language for analysis.

Working with *Really* Big Data: Data Cleaning and Assembling

Caveat: Your university (likely) lacks the expertise to set up an efficent `SQL` warehouse.

- Cloud data warehouses like Snowflake and Amazon Redshift offer that service.
- Our smartphone data is on Snowflake.
  - Snowflake is a software-as-a-service (SaaS).
  - You pay for efficient storage and computing power ($>$\$40/hour)
  - Handles huge data with clever "partitioning" (break down big table into smaller chunks!)
  - Repeated operations that rely on similar structures become faster.

Case Study: SQL and Smartphone Data

- Lots of variation in speed across different types of operations
- Subsetting/aggregating operations are fast
- Join/merge operation also relatively fast
- Window functions - e.g., lags/leads, min/max in group, etc - are slow

Selected Queries on tables with 20 billion smartphone visits and 5 variables

SQL Operation	Stata Analog	Time to Run in SQL	Query
where	Keep if	23 sec	select * from visits_all where local_date = '2019-01-01'
group by	collapse	2 min 42 sec	select sum(*) as num_visits, local_date from visits_all group by local_date
join	joinby/merge	9 min 58 sec	select v1.device_key from visits_all as v1 inner join visits_all as v2 on v1.device_key = v2.device_key
lag/lead	xtset/tsset	35 min 13 sec	select lag(unix_time) over(partition by device_key order by unix_time) as lag_time from visits_all

Working with *Really* Big Data: Estimation

Problem: Discrete choice model has huge choice set of venues in large cities.

Possible Solution:

1. Break down data into smaller geography: Smallest reasonable unit (city) still too large.
2. Alternative programming language: `Stata`'s asclogit already a fast command.
3. Alternative estimation approach: Machine learning less familiar to econ audience.
4. Use discrete choice theory to reduce dimensionality: Worked well so far!
   - Venue FE estimation is particularly time-consuming
   - One solution is to exploit IIA and randomly drop venues.
   - Other solution is to derive venue FE analytically from utility max problem to concentrate them out (we coded this estimator in `Julia`.)

Machine Learning (ML) and Big Data

- Limited objective: Give you a sense of the kind of problems and data ML can help with.
- Many tools seen so far are useful when data is big in the sense of billions of observations.
- ML is useful when data is big in the sense of being complex and high dimensional.
  - E.g. Image recognition: Predict face from set of pixel .
  - E.g. Text analysis: Classsifying industry using 10-K fillings
  - E.g. Regression with tons of regressors, discrete choice with huge/rich choice sets.
- Slides drawn from: Mullainathan and Spiess, *Machine Learning: An Applied Econometric Approach*, Journal of Economic Perspective (2017)
- Slides also drawn from more technical: Athey and Imbens, *Machine Learning Methods Economists Should Know About*, March 2019

Machine Learning vs Traditional Econometrics

- Traditional Econometrics:
  - Write statistcal model relating variables as function of parameters.
  - Estimate best fit parameters using random data sample (e.g., minimize squared error.)
  - Focused on property of estimator (asymptotic), causality, and confidence intervals.
- Machine Learning:
  - Write an algorithm that predicts (or classify) x based on set of y.
  - Focus on out-of-sample performance and finding general features from complex patterns.
  - Less focused on property of estimator, causality, and confidence intervals.
    - Could hurt in econ journal.

Machine Learning: An Example of Application

- Example in Mullainathan and Spiess (2017): Predict house prices
- Can do it with OLS, such hedonic regressions are popular.
  - But what if 150 house characteristics (e.g, # bedroom), and 1000s of interactions?
  - High R^2 in OLS could just mean overfitting with poor out-of-sample performance.
- ML tools well-suited to find variable and interactions with predictive power.
  - LASSO: sometimes used in economics.
  - Regression tree, random forest, etc: rarely used in economics.
- Principal Component Analysis: not covered in MS2017 but often used by economists to reduce dimensionality and avoid overfitting.

Machine Learning vs Deep Learning.

- Image recognition is an exciting ML application for spatial economists:
  - Measure local poverty
  - Measure neighborhood characteristics
  - Identify slums
- Deep Learning (DL) tools (e.g., neural networks) more powerful than traditional ML, which matters for video, audio, and image recognition.
- Introductory DL text: Goodfellow, I., Y. Bengio, and A. Courville (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org
- In the next slides, I reproduce two slides from Tsivanidis and Gechter (2019), that compare ML with DL in the context of identifying slums from pictures in Mumbai.

Approach 1: ML

Method 1: ML Approach

- Summary: Machine "learns" mapping between a representation of the data and the outcome
- Step 1: Extract N features $\{X_{1i}, X_{2i}, \ldots, X_{Ni}\}$ for each pixel i.
- Step 2: Estimate the function $Y_i = f(X_{1i}, X_{2i}, \ldots, X_{Ni})$ using your favorite method (e.g. Random Forest)
- Key characteristics: Researcher chooses representation of the data.

Example: Predicting whether an image contains a person or not

- Extract features predictive of being a person, e.g. arms, legs
- But may be difficult: arms and legs look different based on angle/light etc, may be hard to hand-pick best way to extract features...

Approach 2: Deep Convolutional Neural Network

Summary: Machine "learns"

1. Mapping between a representation of the data and the outcome, AND
2. Representation of the data itself

Method:

- Network is a number of layers consisting of many nodes
- A node within a layer applies convolutions (i.e matrix transformation) to inputs from the previous layers, and aggregates up using a non-linear transformation
- Researcher chooses: hyperparameters (e.g. number of layers), the machine "learns" the rest...

Example: Predicting whether an image contains a person or not

- Layer 1: Take raw image in, apply a convolution to isolate edges
- Layer 2: Take edges as inputs, apply a convolution to aggregate into contours
- Layer 3: Take contours as inputs, aggregate these into objects (arm, leg, etc)
- Output Layer: Predict whether image is person or not based on objects

Additional Reading

- Varian, *Big Data: New Tricks for Econometrics*, Journal of Economic Perspective, 2014
  - From chief economist at Google and author of popular micro text.
  - Google tools for manipulating big data, and an introduction to ML.

- Einav and Levin, *Economics in the Age of Big Data*, Science, 2014
  - Nice overview of recent advances in big data research in economics.

- https://tradediversion.net/2018/09/17/why-i-encourage-econ-phd-students-to-learn-julia/
  - Why Dingel thinks you should learn Julia.

Appendix: Build automation for research projects

- Guest slides from Jonathan Dingel.
- Your research code input-output structure is a directed graph (dependency graph).
- Software build tools automate compiling source code into executable binaries.
  - If you've installed Linux packages, you've likely used `Make`.
- Build tools use a dependency graph and information about file changes (e.g., timestamps) to produce output using (all and only) necessary steps.

Appendix: Build automation with Make

- Build automation is valuable for any non-trivial research project.
  - Build automation can be particularly valuable for big data.
  - E.g., To process data for 100 cities, we shouldn't manually track which cities are up-to-date and which need to run the latest code.
  - Define the dependencies and let the build tool track everything.
- `Make` is an old, widely used build tool.
  - Should be available on every Linux box by default (e.g., it's available in Census RDCs)
  - For Mac users, it's in OS X's developer tools.
- Other build tools: Gentzkow and Shapiro use SCons (`Python`-based).
- If all your code in `Stata` consider `project` by Robert Picard.

Appendix: Makefiles

A Makefile consists of a dependency graph and a recipe for each graph node.

- Define dependencies by writing a target before the colon and that target's prerequisites after the colon.
- The next line gives the recipe that translates those inputs into output. Make can execute any recipe you can write on the command line.

```
Y.tex: Y.do X.dta
   stata-se Y.do
Z.pdf: Y.tex
   pdflatex -jobname=Z Y.tex
```

- When you type `make`, it will run `Y.do` in `StataSE` only if `Y.tex` does not exist or if `Y.tex` is older than `Y.do` or `X.dta`.
- `Make` will compile `Z.pdf` if `Y.tex` has changed since you last compiled `Z.pdf`. Changes to `X.dta` or `Y.do` trigger this.