Nuts and Bolts

Matthew Gentzkow
Jesse M. Shapiro

Chicago Booth and NBER

Introduction

- We have focused on the statistical / econometric issues that arise with big data
- In the time that remains, we want to spend a little time on the *practical* issues...

Introduction

- We have focused on the statistical / econometric issues that arise with big data
- In the time that remains, we want to spend a little time on the *practical* issues...
  - E.g., where do you actually put a 2 TB dataset?

Introduction

- We have focused on the statistical / econometric issues that arise with big data
- In the time that remains, we want to spend a little time on the *practical* issues...
  - E.g., where do you actually put a 2 TB dataset?
- Goal: Sketch some basic computing ideas relevant to working with large datasets.

Introduction

- We have focused on the statistical / econometric issues that arise with big data
- In the time that remains, we want to spend a little time on the *practical* issues...
  - E.g., where do you actually put a 2 TB dataset?
- Goal: Sketch some basic computing ideas relevant to working with large datasets.
- Caveat: We are all amateurs

The Good News

- Much of what we've talked about here you can do on your laptop
  - Your OS knows how to do parallel computing (multiple processors, multiple cores)
  - Many "big" datasets are < 5 GB
  - Save the data to local disk, fire up Stata or R, and off you go...

How Big is Big?

Congressional record text (1870-2010)	≈50 GB
Congressional record pdfs (1870-2010)	≈500 GB
Nielsen scanner data (34k stores, 2004-2010)	≈5 TB
Wikipedia (2013)	≈6 TB
20% Medicare claims data (1997-2009)	≈10 TB
Facebook (2013)	≈100,000 TB
All data in the world	≈2.7 billion TB

Outline

- Software engineering for economists

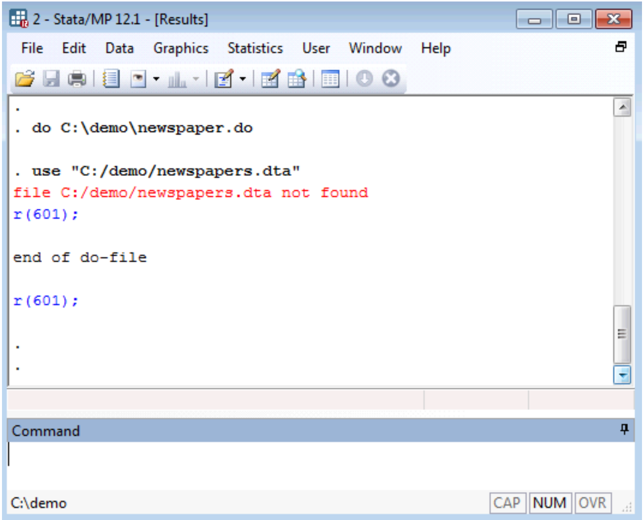
- Databases
- Cluster computing
- Scenarios

Software Engineering for Economists

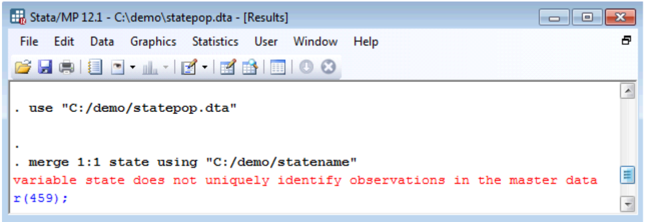
Motivation

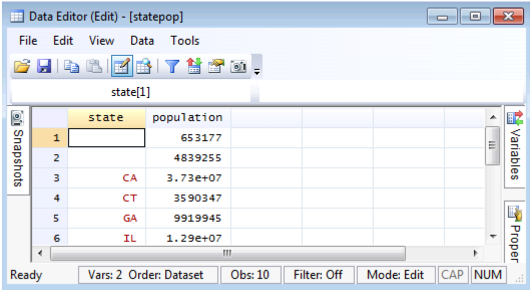
- A lot of the time spent in empirical research is writing, reading, and debugging code.
- Common situations...

Broken Code


2 - Stata/MP 12.1 - [Results]
File Edit Data Graphics Statistics User Window Help
.
. do C:\demo\newspaper.do
. use "C:/demo/newspapers.dta"
file C:/demo/newspapers.dta not found
r(601);
end of do-file
r(601);
.
.
Command
C:\demo
CAP NUM OVR

Incoherent Data



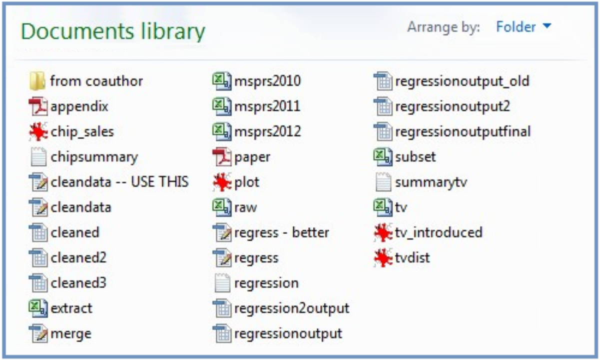


Rampant Duplication

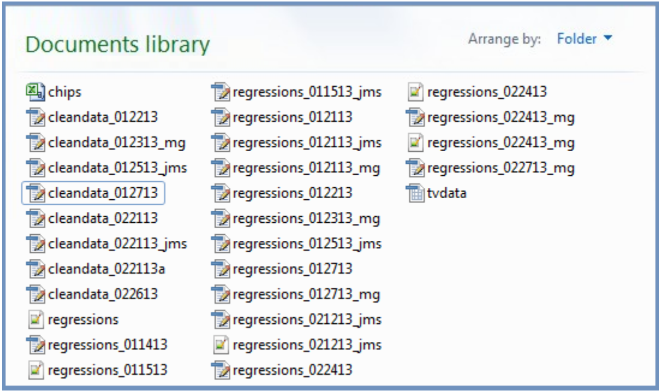
Do-file Editor - reg_duplication

```
use "C:/demo/data.dta", clear

regress probgen coll pharm hlthworker hlthmajor scimajor i.market
regress probgen coll pharm hlthworker hlthmajor scimajor i.year
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.year
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.year age
regress probgen coll pharm hlthworker hlthmajor scimajor i.market income
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.product
regress probgen coll pharm hlthworker hlthmajor scimajor i.year age income
regress probgen coll pharm hlthworker hlthmajor scimajor i.market age income
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.product i.year age income
regress probgen coll pharm hlthworker hlthmajor scimajor i.product i.year age income
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.year income
```

Replication Impossible



Tons of Versions



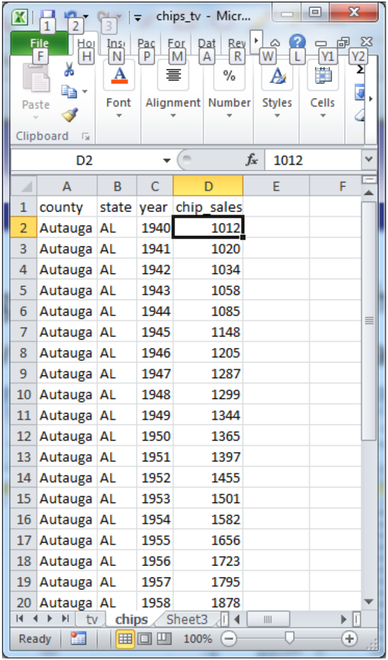
This Talk

- We are not software engineers or computer scientists.
- But we have learned that most common problems in social sciences have analogues in these fields and there are standard solutions.
- Goal is to highlight a few of these that we think are especially valuable to researchers.
- Focus on incremental changes: one step away from common practice.

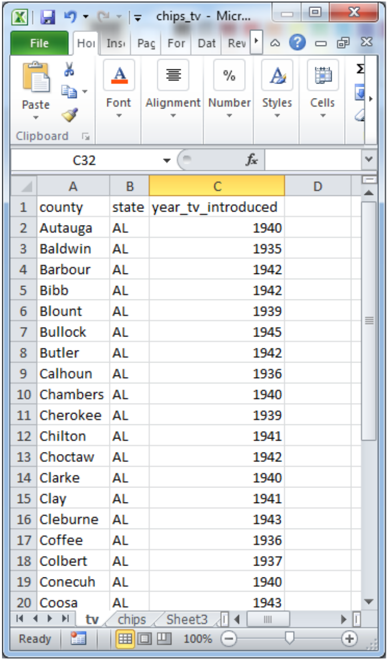
Automation

Raw Data

Data from original source...



	A	B	C	D
1	county	state	year	chip_sales
2	Autauga	AL	1940	1012
3	Autauga	AL	1941	1020
4	Autauga	AL	1942	1034
5	Autauga	AL	1943	1058
6	Autauga	AL	1944	1085
7	Autauga	AL	1945	1148
8	Autauga	AL	1946	1205
9	Autauga	AL	1947	1287
10	Autauga	AL	1948	1299
11	Autauga	AL	1949	1344
12	Autauga	AL	1950	1365
13	Autauga	AL	1951	1397
14	Autauga	AL	1952	1455
15	Autauga	AL	1953	1501
16	Autauga	AL	1954	1582
17	Autauga	AL	1955	1656
18	Autauga	AL	1956	1723
19	Autauga	AL	1957	1795
20	Autauga	AL	1958	1878



	A	B	C
1	county	state	year_tv_introduced
2	Autauga	AL	1940
3	Baldwin	AL	1935
4	Barbour	AL	1942
5	Bibb	AL	1942
6	Blount	AL	1939
7	Bullock	AL	1945
8	Butler	AL	1942
9	Calhoun	AL	1936
10	Chambers	AL	1940
11	Cherokee	AL	1939
12	Chilton	AL	1941
13	Choctaw	AL	1942
14	Clarke	AL	1940
15	Clay	AL	1941
16	Cleburne	AL	1943
17	Coffee	AL	1936
18	Colbert	AL	1937
19	Conecuh	AL	1940
20	Coosa	AL	1943

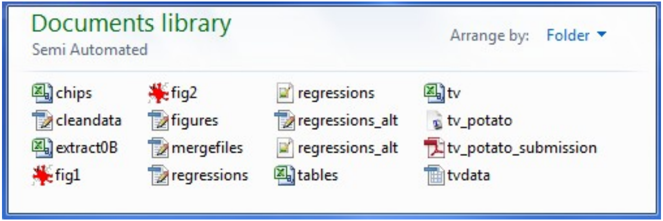
Manual Approach

- Open spreadsheet
- Output to text files
- Open Stata
- Load data, merge files
- Compute log(chip sales)
- Run regression
- Copy results to MS Word and save

Manual Approach

- Two main problems with this approach
  - Replication: how can we be sure we'll find our way back to the exact same numbers?
  - Efficiency: what happens if we change our mind about the right specification?

Semi-automated Approach



- Problems
  - Which file does what?
  - In what order?

Fully Automated Approach

```
File:  rundirectory.bat
stattransfer export_to_csv.stc
statase -b mergefiles.do
statase -b cleandata.do
statase -b regressions.do
statase -b figures.do
pdflatex tv_potato.tex
```

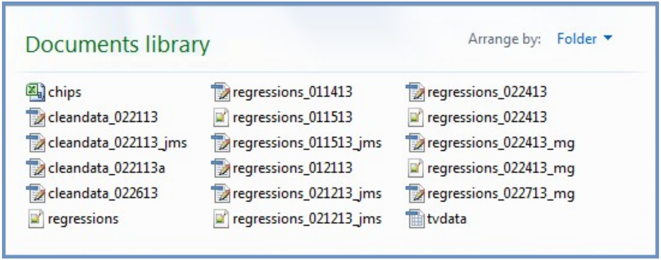
- All steps controlled by a shell script
- Order of steps unambiguous
- Easy to call commands from different packages

Make

- Framework to go from source to target
- Tracks dependencies and revisions
- Avoids rebuilding components that are up to date
- Used to build executable files

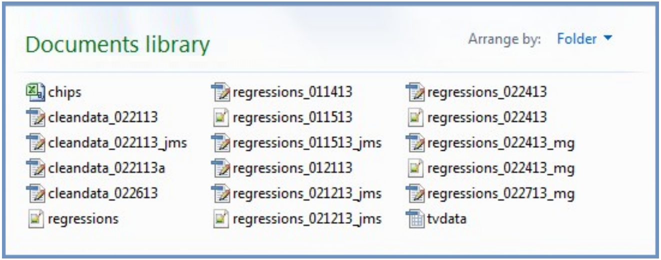
Version Control

After Some Editing



- Dates demarcate versions, initials demarcate authors
- Why do this?
  - Facilitates comparison
  - Facilitates "undo"

What's Wrong with the Approach?

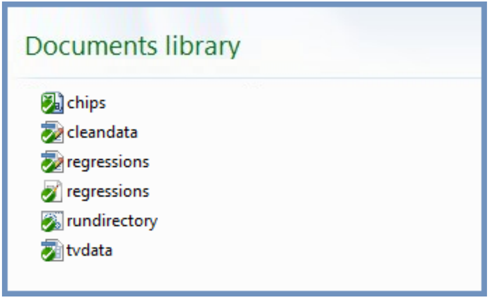


- Why not do this?
  - It's a pain: always have to remember to "tag" every new file
  - It's confusing:
    - Which log file came from `regressions_022713_mg.do`?
    - Which version of cleandata.do makes the data used by `regressions_022413.do`?
  - It fails the market test: No software firm does it this way

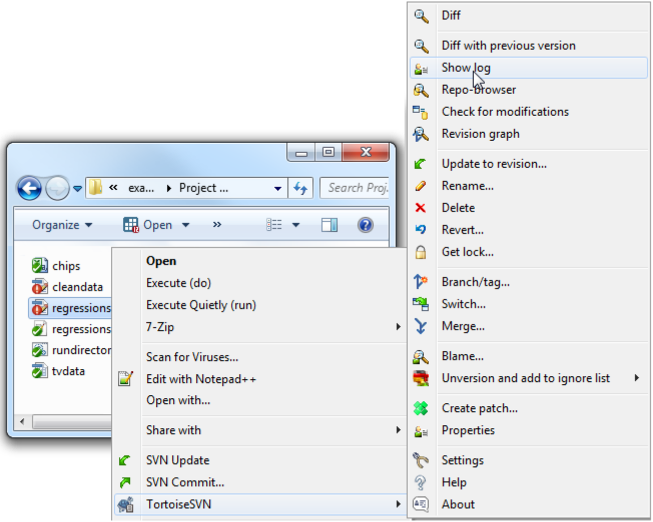
Version Control

- Software that sits "on top" of your filesystem
  - Keeps track of multiple versions of the same file
  - Records date, authorship
  - Manages conflicts
- Benefits
  - Single authoritative version of the directory
  - Edit without fear: an undo command for everything

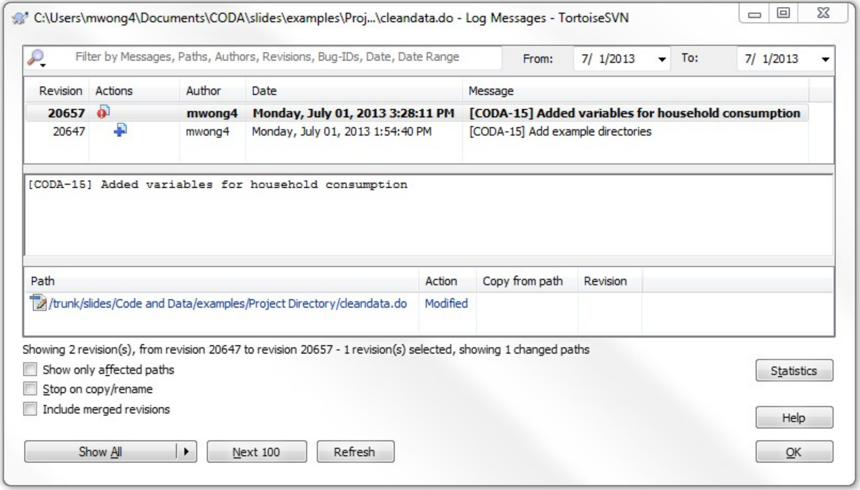
Life After Version Control



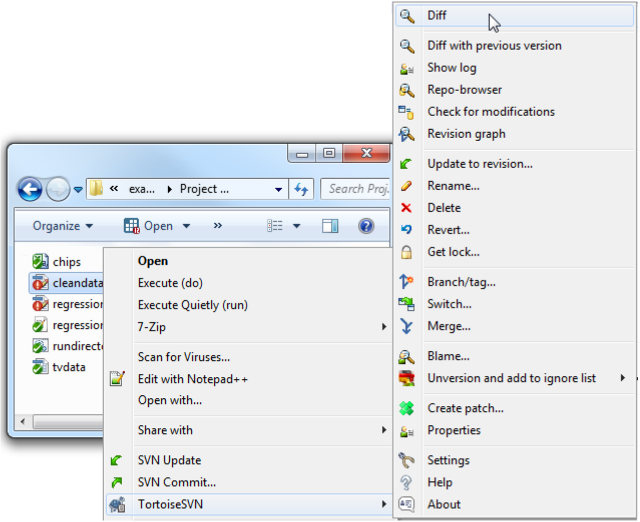
Life After Version Control



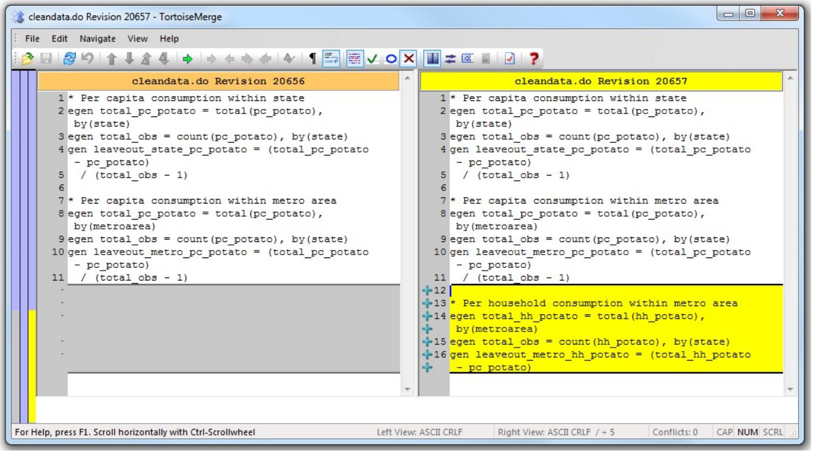
Life After Version Control



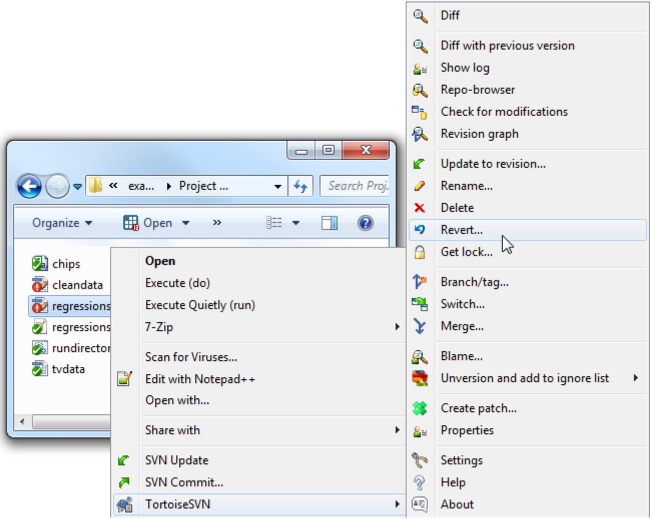
Life After Version Control



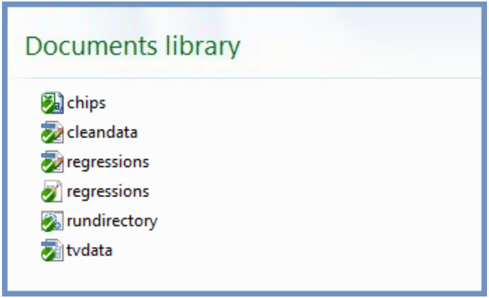
Life After Version Control



Life After Version Control



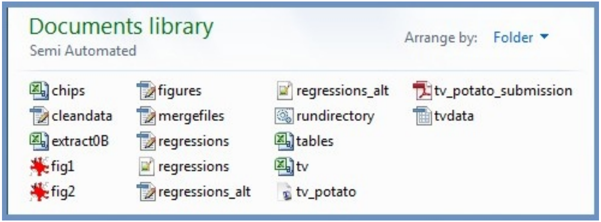
Life After Version Control



- Aside: If you always run rundirectory.bat before you commit, you guarantee replicability.

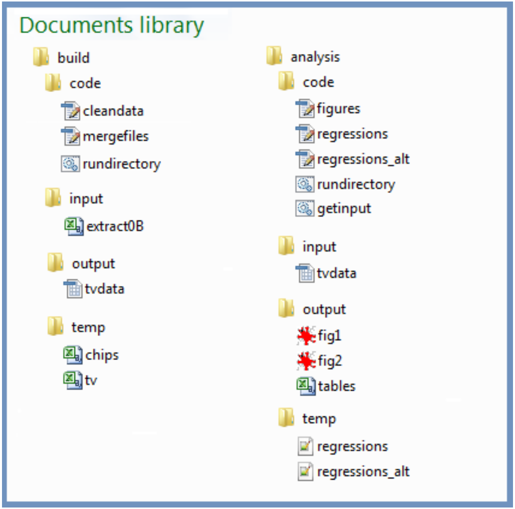
Directories

One Directory Does Everything

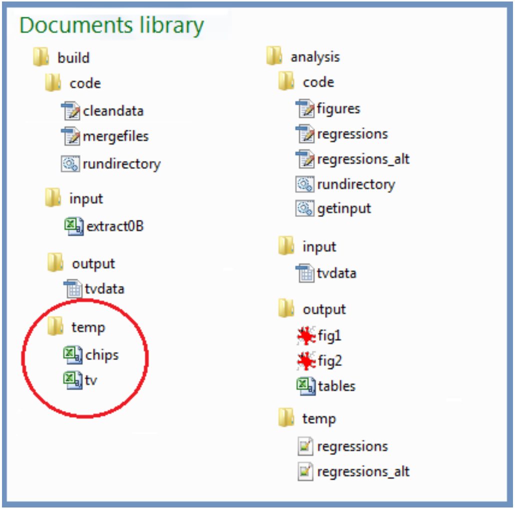


- Pros: Self-contained, simple
- Cons:
    - Have to rerun everything for every change
    - Hard to figure out dependencies

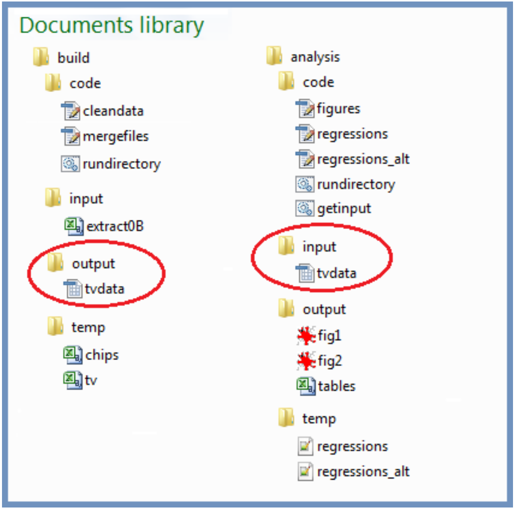
Functional Directories



Dependencies Obvious



One Resource, Many Projects



Keys

Research Assistant Output

county	state	cnty_pop	state_pop	region
36037	NY	3817735	43320903	1
36038	NY	422999	43320903	1
36039	NY	324920	.	1
36040	.	143432	43320903	1
.	NY	.	43320903	1
37001	VA	3228290	7173000	3
37002	VA	449499	7173000	3
37003	VA	383888	7173000	4
37004	VA	483829	7173000	3

Causes for Concern

county	state	cnty_pop	state_pop	region
36037	NY	3817735	43320903	1
36038	NY	422999	43320903	1
36039	NY	324920	.	1
36040	.	143432	43320903	1
.	NY	.	43320903	1
37001	VA	3228290	7173000	3
37002	VA	449499	7173000	3
37003	VA	383888	7173000	4
37004	VA	483829	7173000	3

Relational Databases

county	state	population
36037	NY	3817735
36038	NY	422999
36039	NY	324920
36040	NY	143432
37001	VA	3228290
37002	VA	449499
37003	VA	383888
37004	VA	483829

state	population	region
NY	43320903	1
VA	7173000	3

- Each *variable* is an attribute of an *element* of the table
- Each table has a *key*
- Tables are connected by *foreign keys* (state field in the county table)

Steps

- Store data in normalized format as above
  - Can use flat files, doesn't have to be fancy relational database software
- Construct a second set of files with key transformations
  - e.g., log population
- Merge data together and run analysis

To Come

- What to do with enormous databases?

Abstraction

Rampant Duplication

```
use "C:/demo/data.dta", clear

regress probgen coll pharm hlthworker hlthmajor scimajor i.market
regress probgen coll pharm hlthworker hlthmajor scimajor i.year
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.year
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.year age
regress probgen coll pharm hlthworker hlthmajor scimajor i.market income
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.product
regress probgen coll pharm hlthworker hlthmajor scimajor i.year age income
regress probgen coll pharm hlthworker hlthmajor scimajor i.market age income
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.product i.year age income
regress probgen coll pharm hlthworker hlthmajor scimajor i.product i.year age income
regress probgen coll pharm hlthworker hlthmajor scimajor i.market i.year income
```

Abstracted

```
use "C:/demo/data.dta", clear

local dep_var       "probgen"
local info_proxies  "coll pharm hlthworker hlthmajor scimajor"

regress `dep_var' `info_proxies' i.market
regress `dep_var' `info_proxies' i.year
regress `dep_var' `info_proxies' i.market i.year
regress `dep_var' `info_proxies' i.market i.year age
regress `dep_var' `info_proxies' i.market income
regress `dep_var' `info_proxies' i.market i.product
regress `dep_var' `info_proxies' i.year age income
regress `dep_var' `info_proxies' i.market age income
regress `dep_var' `info_proxies' i.market i.product i.year age income
regress `dep_var' `info_proxies' i.product i.year age income
regress `dep_var' `info_proxies' i.market i.year income
```

Three Leave-Out Means

```
* Per capita consumption within state
egen total_pc_potato = total(pc_potato), by(state)
egen total_obs = count(pc_potato), by(state)
gen leaveout_state_pc_potato = (total_pc_potato - pc_potato)/(total_obs - 1)

* Per capita consumption within metro area
egen total_pc_potato = total(pc_potato), by(metroarea)
egen total_obs = count(pc_potato), by( state )
gen leaveout_metro_pc_potato = (total_pc_potato - pc_potato)/(total_obs - 1)

* Per household consumption within metro area
egen total_hh_potato = total(hh_potato), by(metroarea)
egen total_obs = count(hh_potato), by( state )
gen leaveout_metro_hh_potato = (total_hh_potato - pc_potato )
```

Copy and Paste Errors

Do-file Editor - copy_paste_errors

File Edit Tools View

copy_paste_errors

```
* Per capita consumption within state
egen total_pc_potato = total(pc_potato), by(state)
egen total_obs = count(pc_potato), by(state)
gen leaveout_state_pc_potato = (total_pc_potato - pc_potato)/(total_obs - 1)

* Per capita consumption within metro area
egen total_pc_potato = total(pc_potato), by(metroarea)
egen total_obs = count(pc_potato), by(state)
gen leaveout_metro_pc_potato = (total_pc_potato - pc_potato)/(total_obs - 1)

* Per household consumption within metro area
egen total_hh_potato = total(hh_potato), by(metroarea)
egen total_obs = count(hh_potato), by(state)
gen leaveout_metro_hh_potato = (total_hh_potato - pc_potato)

```

Ready | Line: 1, Col: 37 | CAP | NUM | OVR

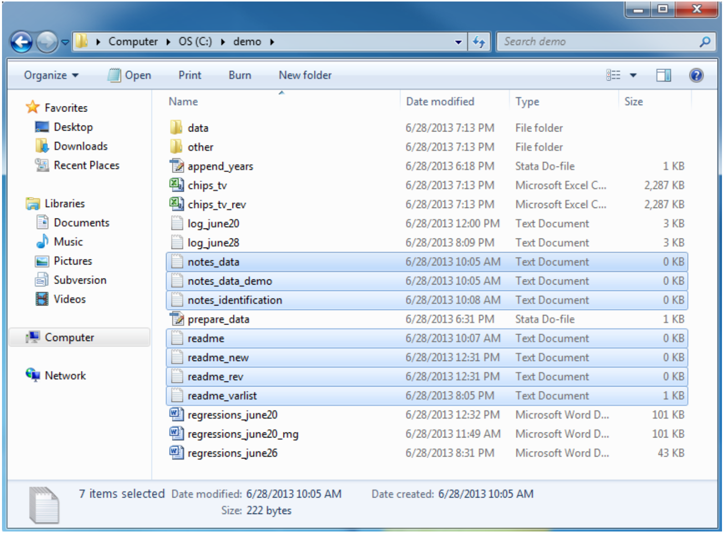
Abstracted

Do-file Editor - abstracted

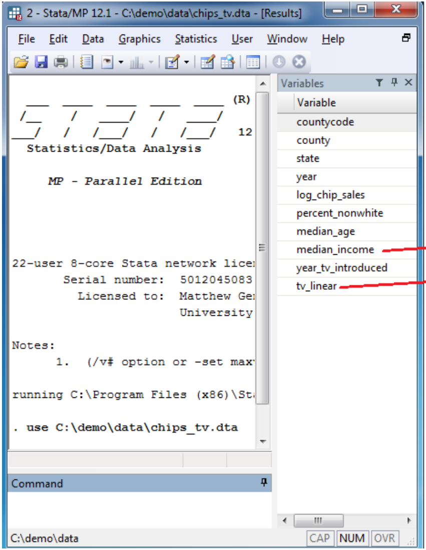
```
program leaveout_mean
    syntax, invar(varname) outvar(name) byvar(varname)
    tempvar tot_invar count_invar
    egen `tot_invar' = total(`invar'), by(`byvar')
    egen `count_invar' = count(`invar'), by(`byvar')
    gen `outvar' = (`tot_invar' - `invar') / (`count_invar' - 1)
end

leaveout_mean, invar(pc_potato) outvar(leaveout_state_pc_potato) byvar(state)
leaveout_mean, invar(pc_potato) outvar(leaveout_metro_pc_potato) byvar(metro)
leaveout_mean, invar(hh_potato) outvar(leaveout_metro_hh_potato) byvar(metro)
```

Documentation

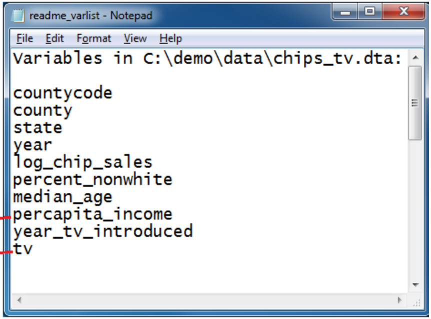
Too Much Documentation



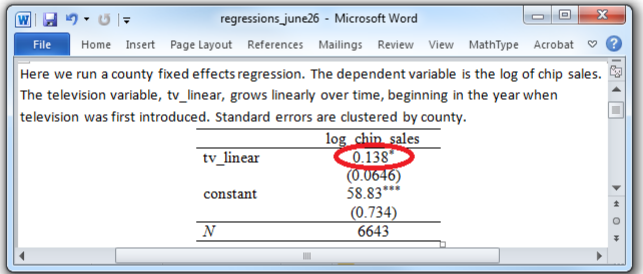
Too Much Documentation



?

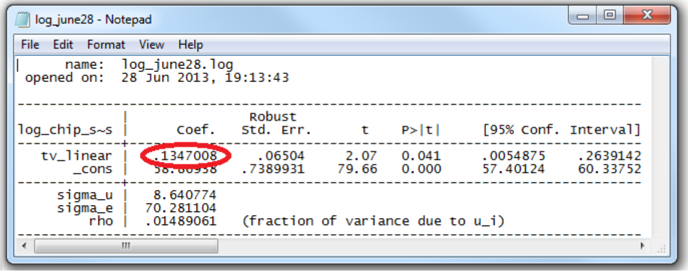


Too Much Documentation



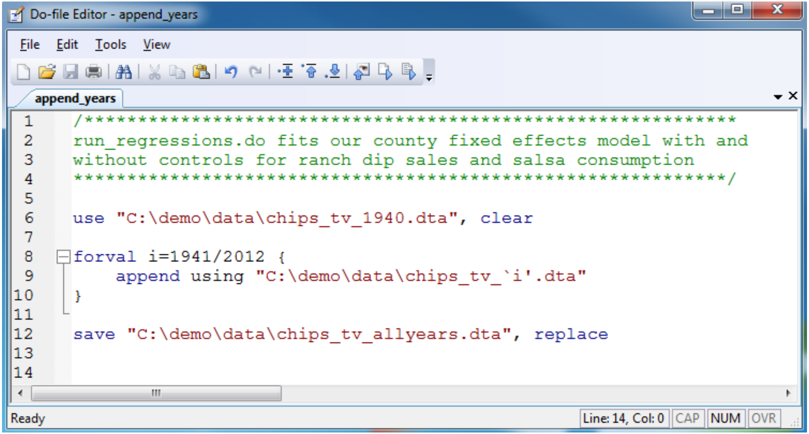
Here we run a county fixed effects regression. The dependent variable is the log of chip sales. The television variable, tv_linear, grows linearly over time, beginning in the year when television was first introduced. Standard errors are clustered by county.

	log_chip_sales
tv_linear	0.138^{*}
	(0.0646)
constant	58.83^{***}
	(0.734)
N	6643



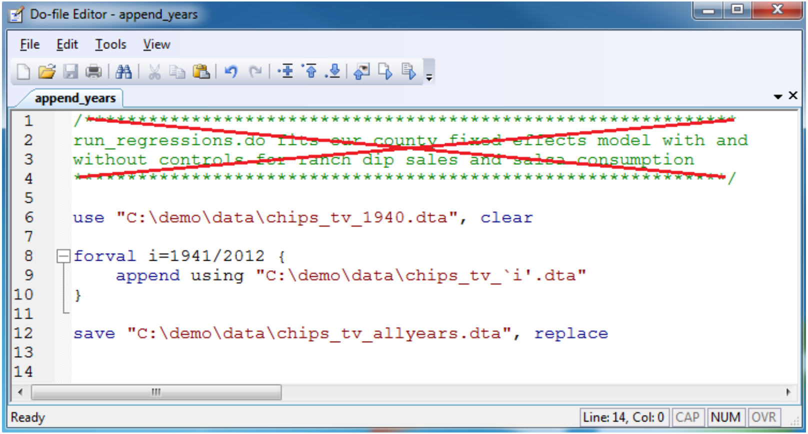

```
       name:  log_june28.log
  opened on:  28 Jun 2013, 19:13:43

------------------------------------------------------------------------------
             |               Robust
log_chip_s~s |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   tv_linear |   .1347008     .06504     2.07   0.041     .0054875    .2639142
       _cons |   58.80938   .7389931    79.66   0.000     57.40124    60.33752
-------------+----------------------------------------------------------------
     sigma_u |   8.640774
     sigma_e |  70.281104
         rho |  .01489061   (fraction of variance due to u_i)
------------------------------------------------------------------------------
```

Too Much Documentation

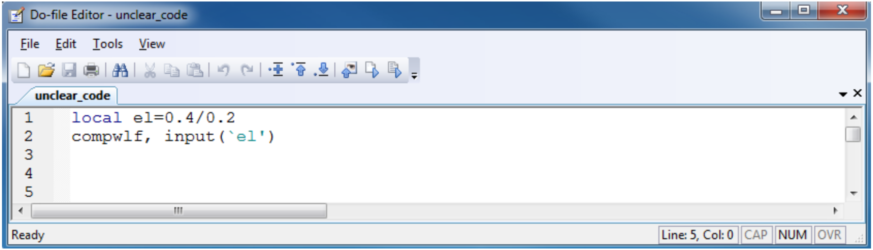



```
/******************************************************************
run_regressions.do fits our county fixed effects model with and
without controls for ranch dip sales and salsa consumption
*****************************************************************/

use "C:\demo\data\chips_tv_1940.dta", clear

forval i=1941/2012 {
    append using "C:\demo\data\chips_tv_`i'.dta"
}

save "C:\demo\data\chips_tv_allyears.dta", replace
```

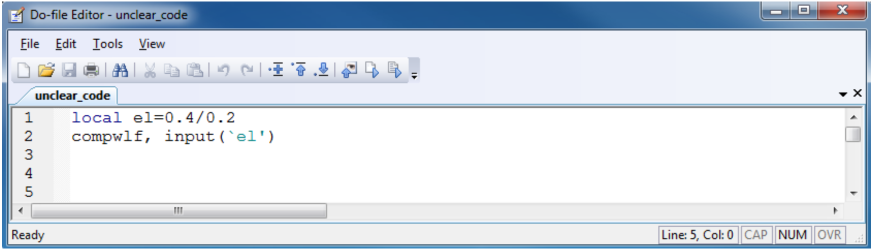
Too Much Documentation

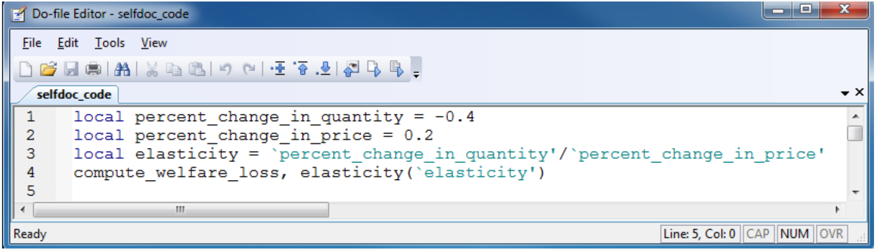


Unclear Code



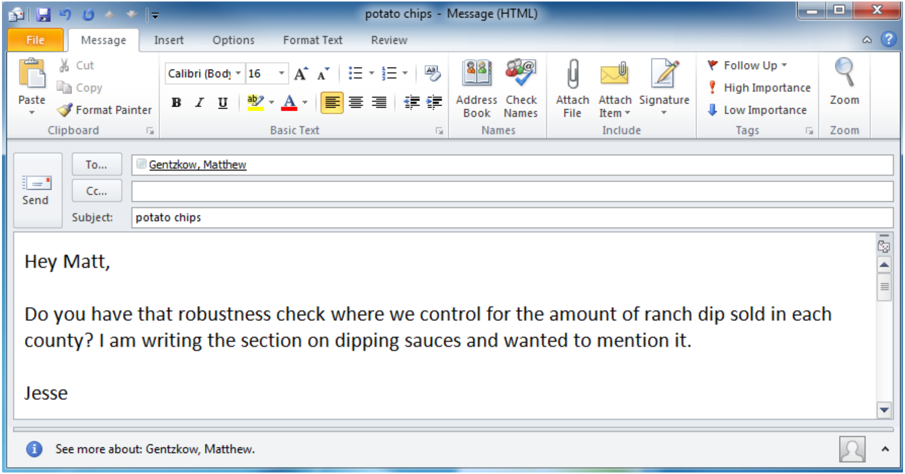
Self-Documenting Code



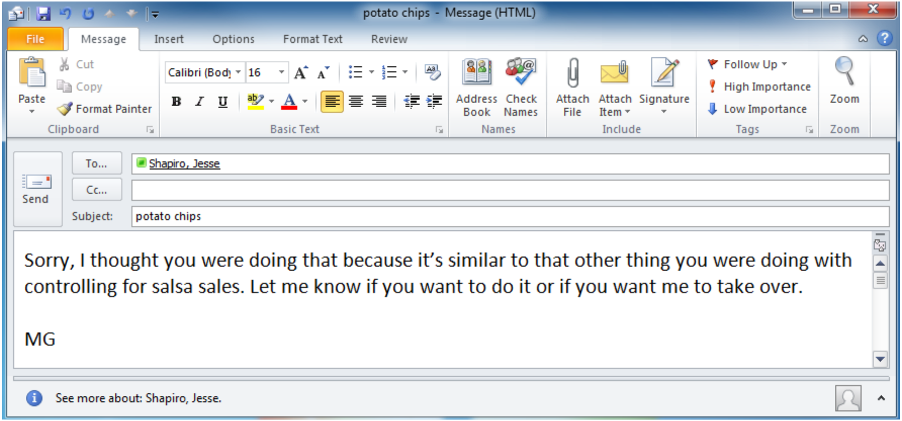


Management

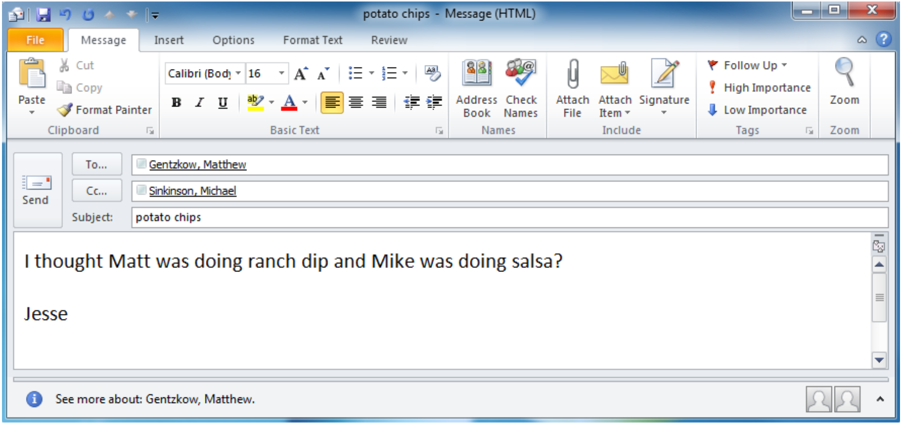
A Friendly Chat



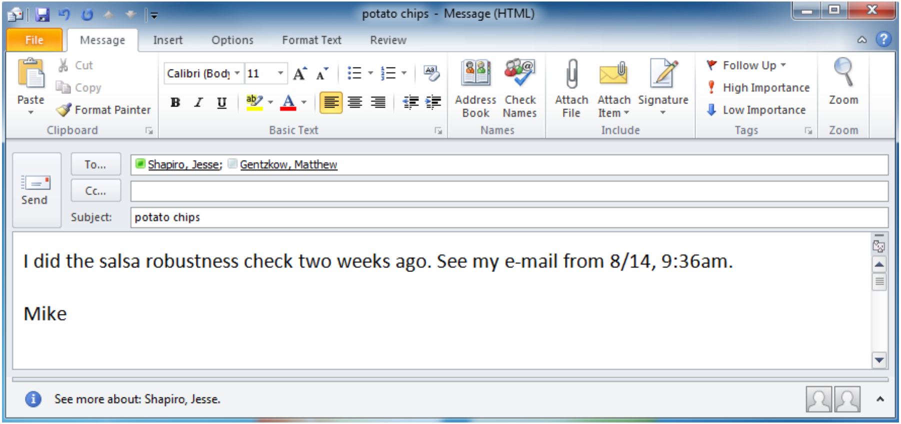
A Friendly Chat



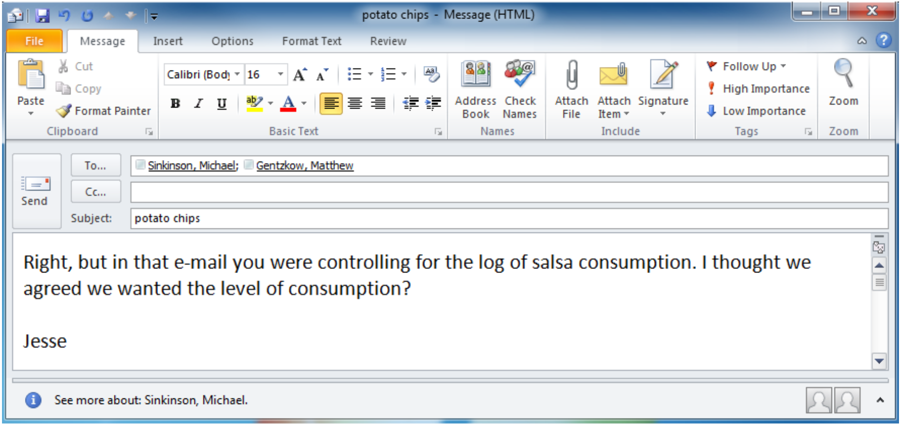
A Friendly Chat



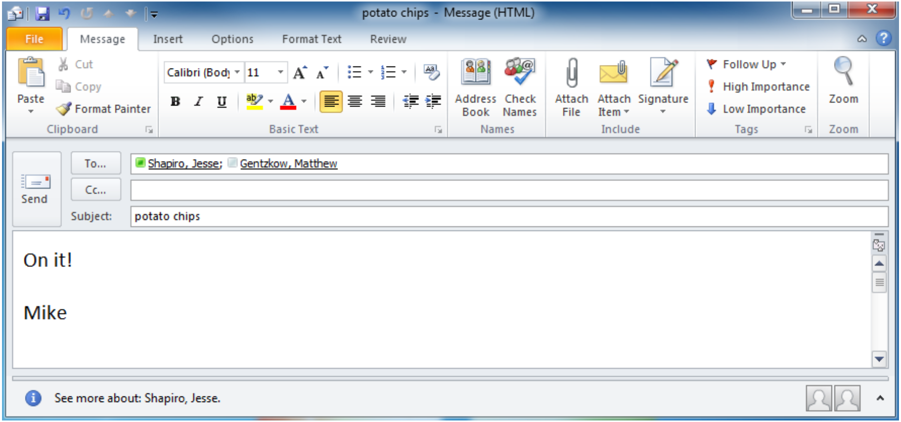
A Friendly Chat



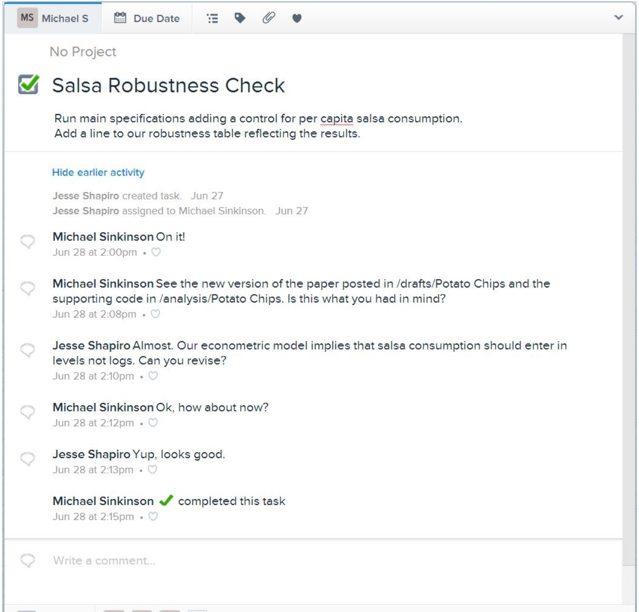
A Friendly Chat



A Friendly Chat



Task Management



Parting Thoughts

Code and Data

- Data are getting larger
- Research is getting more collaborative
- Need to manage code and data responsibly for collaboration and replicability
- Learn from the pros, not from us

Databases

What is a Database?

- Database Theory
  - Principles for how to store / organize / retrieve data efficiently (normalization, indexing, optimization, etc.)
- Database Software
  - Manages storage / organization / retrieval of data (SQL, Oracle, Access, etc.)
  - Economists rarely use this software because we typically store data in flat files & interact with them using statistical programs
  - When we receive extracts from large datasets (the census, Medicare claims, etc.) someone else often interacts with the database on the back end

Normalization

- *"Database Normalization* is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them."

Benefits of Normalization

- Efficient storage
- Efficient modification
- **Guarantees coherence**
- **Makes logical structure of data clear**

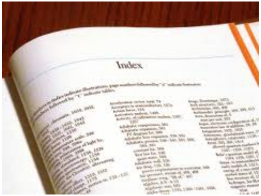
Indexing

- Medicare claims data for 1997-2010 are roughly 10 TB
- These data are stored at NBER in thousands of zipped SAS files

Indexing

- Medicare claims data for 1997-2010 are roughly 10 TB
- These data are stored at NBER in thousands of zipped SAS files
- To extract, say, all claims for heart disease patients aged 55-65, you would need to read every line of every one of those files
    - THIS IS SLOW!!!

Indexing



- The obvious solution, long understood for book, libraries, economics journals, and so forth, is to build an index
- Database software handles this automatically
    - Allows you to specify fields that will be often used for lookups, subsetting, etc. to be indexed
    - For the Medicare data, we could index age, gender, type of treatment, etc. to allow much faster extraction

Indexing

- Benefits
  - Fast lookups
  - Easy to police data constraints
- Costs
  - Storage
  - Time
- Database *optimization* is the art of tuning database structure and indexing for a specific set of needs

Data Warehouses

- Traditional databases are optimized for *operational* environments
  - Bank transactions
  - Airline reservations
  - etc.

Data Warehouses

- Traditional databases are optimized for *operational* environments
  - Bank transactions
  - Airline reservations
  - etc.
- Characteristics
  - Many small reads and writes
  - Many users accessing simultaneously
  - Premium on low latency
  - Only care about current state

Data Warehouses

- In analytic / research environments, however, the requirements are different
  - Frequent large reads, infrequent writes
  - Relatively little simultaneous access
  - Value throughput relative to latency
  - May care about history as well as current state
  - Need to create and re-use many custom extracts

Data Warehouses

- In analytic / research environments, however, the requirements are different
  - Frequent large reads, infrequent writes
  - Relatively little simultaneous access
  - Value throughput relative to latency
  - May care about history as well as current state
  - Need to create and re-use many custom extracts
- Database systems tuned to these requirements are commonly called "data warehouses"

Distributed Computing

Distributed Computing

- Definition: Computation shared among many independent processors

Distributed Computing

- Definition: Computation shared among many independent processors
- Terminology
  - Distributed vs. Parallel (latter usually refers to systems with shared memory)
  - Cluster vs. Grid (latter usually more decentralized & heterogeneous)

On Your Local Machine

- Your OS can run multiple processors each with multiple cores
- Your video card has hundreds of cores
- Stata, R, Matlab, etc. can all exploit these resources to do parallel computing

On Your Local Machine

- Your OS can run multiple processors each with multiple cores
- Your video card has hundreds of cores
- Stata, R, Matlab, etc. can all exploit these resources to do parallel computing
- Stata
    - Buy appropriate “MP” version of Stata
    - Software does the rest

On Your Local Machine

- Your OS can run multiple processors each with multiple cores
- Your video card has hundreds of cores
- Stata, R, Matlab, etc. can all exploit these resources to do parallel computing
- Stata
  - Buy appropriate "MP" version of Stata
  - Software does the rest
- R / Matlab
  - Install appropriate add-ins (*parallel* package in R, "parallel computing toolbox" in Matlab)
  - Include parallel commands in code (e.g., *parfor* in place of *for* in Matlab)

On Cluster / Grid

- Resources abound
  - University / department computing clusters
  - Non-commercial scientific computing grids (e.g., XSEDE)
  - Commercial grids (e.g., Amazon EC2)

On Cluster / Grid

- Resources abound
  - University / department computing clusters
  - Non-commercial scientific computing grids (e.g., XSEDE)
  - Commercial grids (e.g., Amazon EC2)
- Most of these run Linux w/ distribution handled by a “batch scheduler”
- Write code using your favorite application, then send it to scheduler with a bash script

MapReduce

- MapReduce is a programming model that facilitates distributed computing
  - Developed by Google around 2004, though ideas predate that

MapReduce

- MapReduce is a programming model that facilitates distributed computing
  - Developed by Google around 2004, though ideas predate that
- Most algorithms for distributed data processing can be represented in two steps
  - **Map**: Process individual "chunk" of data to generate an intermediate "summary"
  - **Reduce:** Combine "summaries" from different chunks to produce a single output file

MapReduce

- MapReduce is a programming model that facilitates distributed computing
  - Developed by Google around 2004, though ideas predate that
- Most algorithms for distributed data processing can be represented in two steps
  - **Map**: Process individual "chunk" of data to generate an intermediate "summary"
  - **Reduce:** Combine "summaries" from different chunks to produce a single output file
- If you structure your code this way, MapReduce software will handle all the details of distribution:
  - Partitioning data
  - Scheduling execution across nodes
  - Managing communication between machines
  - Handling errors / machine failures

MapReduce: Examples

- Count words in a large collection of documents
  - Map: Document $i \rightarrow$ Set of $(word, count)$ pairs C_i
  - Reduce: Collapse $\{C_i\}$, summing $count$ within $word$

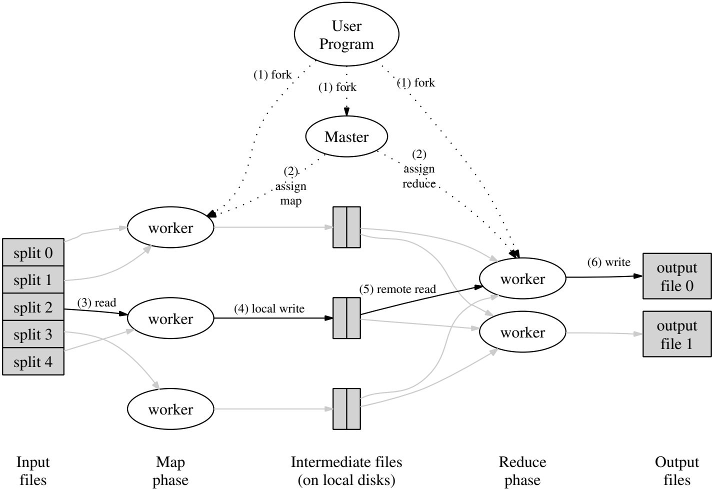
MapReduce: Examples

- Count words in a large collection of documents
  - Map: Document $i \rightarrow$ Set of $(word, count)$ pairs C_i
  - Reduce: Collapse $\{C_i\}$, summing *count* within *word*
- Extract medical claims for 65-year old males
  - Map: Record set $i \rightarrow$ Subset of i that are 65-year old males H_i
  - Reduce: Append elements of $\{H_i\}$

MapReduce: Examples

- Count words in a large collection of documents
  - Map: Document i $\rightarrow$ Set of $(word, count)$ pairs C_i
  - Reduce: Collapse $\{C_i\}$, summing *count* within *word*
- Extract medical claims for 65-year old males
  - Map: Record set i $\rightarrow$ Subset of i that are 65-year old males H_i
  - Reduce: Append elements of $\{H_i\}$
- Compute marginal regression for text analysis (e.g., Gentzkow & Shapiro 2010)
  - Map: Counts x_{ij} of phrase j $\rightarrow$ Parameters $\left(\hat{\alpha}_j, \hat{\beta}_j\right)$ from $E\left(x_{ij}|y_i\right) = \alpha_j + \beta_j x_{ij}$
  - Reduce: Append $\left\{\hat{\alpha}_j, \hat{\beta}_j\right\}$

MapReduce: Model



MapReduce: Implementation

- MapReduce is the original software developed by Google
- Hadoop is the open-source version most people use (developed by Apache)
- Amazon has a hosted implementation (Amazon EMR)

MapReduce: Implementation

- MapReduce is the original software developed by Google
- Hadoop is the open-source version most people use (developed by Apache)
- Amazon has a hosted implementation (Amazon EMR)
- How does it work?
    - Write your code as two functions called *map* and *reduce*
    - Send code & data to scheduler using bash script

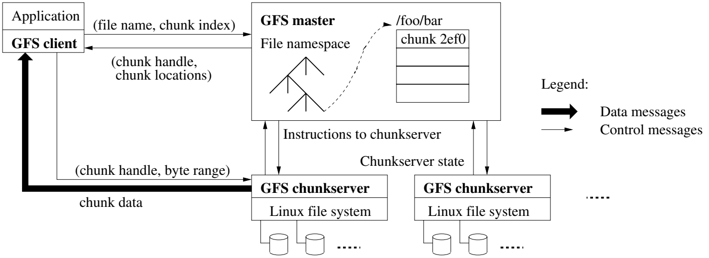
Distributed File Systems

- Data transfer is the main bottleneck in distributed systems
- For big data, it makes sense to distribute data as well as computation
  - Data broken up into chunks, each of which lives on a separate node
  - File system keeps track of where the pieces are and allocates jobs so computation happens "close" to data whenever possible

Distributed File Systems

- Data transfer is the main bottleneck in distributed systems
- For big data, it makes sense to distribute data as well as computation
  - Data broken up into chunks, each of which lives on a separate node
  - File system keeps track of where the pieces are and allocates jobs so computation happens "close" to data whenever possible
- Tight coupling between MapReduce software and associated file systems
  - MapReduce → Google File System (GFS)
  - Hadoop → Hadoop Distributed File System (HDFS)
  - Amazon EMR → Amazon S3

Distributed File Systems



Scenarios

Scenario 1: Not-So-Big Data

- *My data is* 100 *gb or less*

Scenario 1: Not-So-Big Data

- *My data is* 100 *gb or less*
- Advice
  - Store data locally in flat files (csv, Stata, R, etc.)
  - Organize data in normalized tables for robustness and clarity
  - Run code serially or (if computation is slow) in parallel

Scenario 2: Big Data, Small Analysis

- *My raw data is* > 100 *gb, but the extracts I actually use for analysis are* $<< 100$ *gb*

Scenario 2: Big Data, Small Analysis

- *My raw data is* > 100 *gb, but the extracts I actually use for analysis are* $<< 100$ *gb*
- Example
  - Medicare claims data $\rightarrow$ analyze heart attack spending by patient by year
  - Nielsen scanner data $\rightarrow$ analyze average price by store by month

Scenario 2: Big Data, Small Analysis

- *My raw data is* > 100 *gb, but the extracts I actually use for analysis are* $<< 100$ *gb*
- Example
    - Medicare claims data $\rightarrow$ analyze heart attack spending by patient by year
    - Nielsen scanner data $\rightarrow$ analyze average price by store by month
- Advice
    - Store data in relational database optimized to produce analysis extracts efficiently
    - Store extracts locally in flat files (csv, Stata, R, etc.)
    - Organize extracts in normalized tables for robustness and clarity
    - Run code serially or (if computation is slow) in parallel

Scenario 2: Big Data, Small Analysis

- *My raw data is* > 100 *gb, but the extracts I actually use for analysis are* $<< 100$ *gb*
- Example
  - Medicare claims data $\rightarrow$ analyze heart attack spending by patient by year
  - Nielsen scanner data $\rightarrow$ analyze average price by store by month
- Advice
  - Store data in relational database optimized to produce analysis extracts efficiently
  - Store extracts locally in flat files (csv, Stata, R, etc.)
  - Organize extracts in normalized tables for robustness and clarity
  - Run code serially or (if computation is slow) in parallel
- Note: Gains to database increase for more structured data. For completely unstructured data, you may be better off using distributed file system + map reduce to create extracts.

Scenario 3: Big Data, Big Analysis

- *My data is* > 100 *GB and my analysis code needs to touch all of the data*

Scenario 3: Big Data, Big Analysis

- *My data is* > 100 *GB and my analysis code needs to touch all of the data*
- Example
    - 2 TB of SEC filing text $\rightarrow$ run variable selection using all data

Scenario 3: Big Data, Big Analysis

- *My data is* > 100 *GB and my analysis code needs to touch all of the data*
- Example
  - 2 TB of SEC filing text $\rightarrow$ run variable selection using all data
- Advice
  - Store data in distributed file system
  - Use MapReduce or other distributed algorithms for analysis