Introduction to Stata
Lecture 1

Chelsea Garber

Duke University

July 6, 2017
Objectives - Part I

1. Introduction and Motivation

2. Stata Pros and Cons

3. Project Management

4. Programming Guidelines

Note: These lecture slides borrow heavily from Brian Clark’s 2014 slides.
Tell me a little bit about yourself:

- What is your program/year/field?
- What types of datasets are you using/going to use?
- What is your previous experience with Stata?
  - Matlab?
  - \LaTeX{}?
  - Others?
- Is there anything you’re particularly interested in learning about?
Class Expectations

This class will not be:

- A lecture of the Stata manual
- A "how-to" on every possible command
- A one-way demonstration of Stata tools
A Complete Stata Documentation Set contains more than 11,000 pages of information in the following manuals:

[GS] Getting Started with Stata (Mac, Unix, or Windows)
[U] Stata User’s Guide
[R] Stata Base Reference Manual
[XT] Stata Longitudinal-Data/Panel-Data Reference Manual
[MI] Stata Multiple-Imputation Reference Manual
[PSS] Stata Power and Sample-Size Reference Manual
[SVY] Stata Survey Data Reference Manual
[I] Stata Glossary and Index

In addition, installation instructions may be found in the Installation Guide, which comes in the DVD case.
Class Expectations

- This class will provide (I hope):
  - An organized system for managing projects from start to finish
  - A solid foundation for using commands (and learning new ones)
  - A practical guide for coding
  - A hands-on “lab” to explore Stata’s capabilities
You can write code to get Stata to do (almost) anything that you can do by hand with a menu:

- Import all kinds of data
- Merge datasets
- Create complicated variables
- Generate formatted graphics
- Run regressions

Takeaway: Stata is powerful and user-friendly!
There are some tasks where Stata falls short:

- Not great at custom estimation
- Can be very slow when datasets are very large
- Frustrating to manipulate data by hand

Takeaway #1: Use Matlab for any type of estimation you can’t do in Stata

Takeaway #2: Don’t work by hand; write code that is easy to read, manipulate, and share
What else can Stata do?

Stata has many “conventional” programming capabilities in addition to statistical/econometric analysis tools

- Stata can:
  - Erase files anywhere on your computer
  - Unzip or create compressed files
  - Automatically install and update third-party commands
  - Download data from the internet

Takeaway: Anything you can do Stata can do better...and faster
The mistake (almost) everyone makes

- Trying to do things manually that Stata can do by itself
- This is easy to do when you start using Stata:
  - At first it’s faster to do things by hand than invest time learning how to do it with code
  - Most people learn how to code by slowly trying to replace time-consuming things they do by hand with code
How to avoid that mistake

- We’re going to do things the right way: learn how to code first
- Why?
  - Menus only access a subset of what Stata can do
  - We want our work to be replicable
  - Other way is more time consuming in the long run
Examples of the benefits of coding

Let’s take a look at four situations from my experience where using the techniques in this class saved me a lot of time:

1. Downloading data from the internet
2. Merging several data sets together
3. Updating old tables to reflect new or corrected data
4. Reformatting string variables
Example #1: Downloading data

Task: Download Quarterly Census of Employment (QCEW) data from BLS website for each county in the US for years 1990-2015. Merge into one dataset containing all years.

By hand:

- Click through 16 individual files to download from website
- Unzip each file individually
- Copy and paste all years and all counties into one Excel file
  - 3,140 counties * 16 years = 50,240 files
- Import huge Excel file into Stata

Total time: 18-20 (ridiculously boring) hours
Example #1: Downloading data

### QCEW NAICS-Based Data Files (1975-2013)

<table>
<thead>
<tr>
<th>Excel Files</th>
<th>CSVs By Area</th>
<th>Annual Averages</th>
<th>CSVs By Industry</th>
<th>Annual Averages</th>
<th>Quarterly</th>
</tr>
</thead>
<tbody>
<tr>
<td>County High-Level</td>
<td>File Layout</td>
<td>File Layout</td>
<td>File Layout</td>
<td>File Layout</td>
<td>File Layout</td>
</tr>
<tr>
<td>2010</td>
<td>2010</td>
<td>2010</td>
<td>2010</td>
<td>2010</td>
<td>2010</td>
</tr>
</tbody>
</table>
Example #1: Downloading data

- By code:
  - Write .do file loop that imports (and unzips) data directly from website
  - Write .do file loop that automatically appends each county-year
  - Go to the gym, eat lunch, and read a few NYT articles
  - Save final dataset for analysis

- Total time: 30 minutes of coding, 5 hours of leisure while script runs
Example #1: Downloading data

```stata
* annual averages
forvalues i = 1990/2015 {
    copy http://data.bls.gov/cw/download-cw/files/i\'\'csv\'\'i\'_annual_by_area.zip \i\_a.zip
    unzipfile \i\_a.zip
    erase \i\_a.zip
}

* extract the parts we want
local i = 1
foreach year of numlist 1997/2015 {
    di \"year\"
    local filelist : dir \"\year\".annual_by_area* files \*.csv
    qui foreach file of local filelist {
        insheet using \"\year\".annual_by_area\"file\"", clear
        keep if industry_code=="10" | industry_code=="1012" | industry_code=="1013" | industry_code=="102" | industry_code=="1023" | industry_code=="1024"
        gen fips_state = floor(area_tips/1000)
        gen fips_county = area_tips - fips_state*1000
        drop if own_code==1 | own_code==2 | own_code==3
        keep area_tips fips_state fips_county year own_title own_code industry_title industry_code annual_avg_emplvl total_annual_wages annual_avg_estabs_count
        order area_tips fips_state fips_county year own_title own_code industry_title industry_code annual_avg_emplvl total_annual_wages annual_avg_estabs_count

        if \i == 1 save temp.dta, replace
        if \i > 1 {
            append using temp.dta
            save temp.dta, replace
        }
    }
}

drop if fips_county == ,
tostring(industry_code), replace
save qcew_1997_2015.dta, replace

drop if year>2010
save qcew_1997_2010.dta, replace
```
Example #2: Merging data

Task: Merge crime data for each post code unit to house prices in the UK.

- By hand:
  - Compile data from individual police forces around UK.
  - Copy and paste each of these files together for 11 years.
  - Sort by post code and count crimes in each area.
  - Create new file with crime counts for each post code.
  - Merge to house prices by post code.

- Total time: ETERNITY
Example #2: Merging data

Forces:

- Avon and Somerset Constabulary
- British Transport Police
- Cheshire Constabulary
- Cleveland Police
- Derbyshire Constabulary
- Dorset Police
- Dyfed-Powys Police
- Gloucestershire Constabulary
- Gwent Police
- Hertfordshire Constabulary
- Kent Police
- Leicestershire Police
- Merseyside Police
- Norfolk Constabulary
- North Yorkshire Police
- Northumbria Police
- Police Service of Northern Ireland
- South Yorkshire Police
- Suffolk Constabulary
- Sussex
- Warwickshire Police
- West Midlands Police
- Wiltshire Police
- Bedfordshire Police
- Cambridgeshire Constabulary
- City of London Police
- Cumbria Constabulary
- Devon & Cornwall Police
- Durham Constabulary
- Essex Police
- Greater Manchester Police
- Hampshire Constabulary
- Humberside Police
- Lancashire Constabulary
- Lincolnshire Police
- Metropolitan Police Service
- North Wales Police
- Northamptonshire Police
- Nottinghamshire Police
- South Wales Police
- Staffordshire Police
- Surrey
- Thames Valley Police
- West Mercia Police
- West Yorkshire Police
Example #2: Merging data

- By code:
  - Automatically import all forces and all post codes
  - Count crimes by post code with one command
  - Merge on post code to house prices

- Total time: 2 hours
Example #2: Merging data

Define Police Forces to loop over in import
local Forces avon-and-somerset bedfordshire btp cambridgeshire cheshire

Define Years to Loop Over
local Years 2011 2012 2013 2014

Define Months to Loop Over
local Months 01 02 03 04 05 06 07 08 09 10 11 12

Define Loop Over Year, Month, then Police Forces
foreach year of local Years {
    foreach month of local Months {
        foreach force of local Forces {
            // Import Individual Force File, save
            clear
            local currentfile "Raw Data/Crime/'year'-'month'/force.dta"
            display " currentfile"
            insheet using "currentfile"
            drop crimetype fallsoccause outcomecategory context

            // If first entry of year, create pooled set for the year
            if "force" == "avon-and-somerset" & 'month'==01 {
                save "'/Users/Chelsea/Desktop/2nd Year Paper/Main Script Folder/Data/pooled.csv'"
            }

            // Otherwise, save as temp file
            else {
                save "'/Users/Chelsea/Desktop/2nd Year Paper/Main Script Folder/Data/temp.csv'"
                use "'/Users/Chelsea/Desktop/2nd Year Paper/Main Script Folder/Data/pooled.csv'
            }

            // Merge with Crime Counts
            use clean sample
            clear all
            use "'/Users/Chelsea/Desktop/2nd Year Paper/Main Script Folder/Data/zoo.csv'"

            count postcodes in master
            sort postcode
            egen postcode_id=group(postcode)
            sum postcode_id
            replace postcode_id = 848750

            count postcodes in each year of crimes
            foreach year of local Years {
                foreach month of local Months {
                    foreach force of local Forces {
                        import "Raw Data/Crime/'year'-'month'/force.dta"
                        drop crimetype fallsoccause outcomecategory context

                        merge "Unused File Format" using "Master Postcode File"
                        postcode_id
                        postcode_reformat
                        postcode_reformat_id
                        gen postcode_reformat_id = group(postcode_reformat)
                        gen year = 'year'
                        save "'/Users/Chelsea/Desktop/2nd Year Paper/Main Script Folder/Data/Processed/'year'.dta'"
                    }
                }
            }
        }
    }
}
}

Example #3: Updating tables

Task: Update approximately 20 tables after finding a coding error in the dependent variable.

- By hand:
  - Change the incorrect data
  - Re-run each regression one-by-one
  - Manually type in all of the regression results into Word or \LaTeX

- Total time: 6 hours
### Example #3: Updating tables

<table>
<thead>
<tr>
<th>Table 11: Regression Results - Equation (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Inprice</td>
</tr>
<tr>
<td>sector_verime2_L1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sector_perime_L1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sector_drugs_L1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sector_asb_L1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sector_verime2_D1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sector_perime_D1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sector_drugs_D1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sector_asb_D1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>beds</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>_cons</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

*Quarter-Year-Area FE? Yes Yes Yes*

| N | 1093971 | 1001668 | 579737 |

`t` statistics in parentheses

* `p < 0.05`, ** `p < 0.01`, *** `p < 0.001`
Example #3: Updating tables

- By code:
  - Fix the coding mistake
  - Run every regression again in one click
  - Automatically export tables to \LaTeX
  - Compile new tables in one click

- Total time: 10 minutes
Example #3: Updating tables

```stata
ssc install estout
ssc install lactab
ssc install outtex
ssc install sutex2
ssc install tebout

*violent crime lagged once, with and without fixed effects
clear all
use "/Users/Chelsea/Desktop/2nd Year Paper/Main Script Folder/Data/

estimates clear
eststo: quietly reg lnprice sector_vcrime2_l1, r
eststo: quietly reg lnprice sector_vcrime2_l1 _ipc_area_*, cluster()
*putting in quarterly fixed effects
eststo: quietly reg lnpice sector_vcrime2_l1 _ipc_area_*, xq, clust!
eststo: quietly reg lnpice sector_vcrime2_l1 _ipc_area_*, xq if bef

esttab using "/Users/Chelsea/Desktop/2nd Year Paper/Main Script Fo"

*Robustness
estimates clear

*quarterly fixed effects, controlling for all crimes
eststo: quietly reg lnpice sector_vcrime2_l1 sector_vcrime
*more lags of violent crime
eststo: quietly reg lnpice sector_vcrime2_l1 sector_vcrime
*limit bedrooms
eststo: quietly reg lnpice sector_vcrime2_l1 sector_vcrime

esttab using "/Users/Chelsea/Desktop/2nd Year Paper/Main Script Fo"

*unit violent crime
estimates clear

eststo: quietly reg lnpice unit_vcrime2_l1, cluster (pc_area)
eststo: quietly reg lnpice unit_vcrime2_l1 _ipc_area_*, cluster (pc_area)
eststo: quietly reg lnpice unit_vcrime2_l1 _ipc_area_*, xq, cluster (fc ar"
```
Example #4: Reformatting string variables

Task: Reformat names of Community Development Entities so they match and can be perfectly merged across different years’ data sets.

- By hand:
  - Open list of CDE awards in Excel
  - Visually inspect thousands of lines
  - Manually change case, take out commas, take out periods, take out spaces, etc.

- Total time: 10 hours
Example #4: Reformatting string variables

```
Campus Partners for Community Urban Red
CapFund New Markets LLC
CapFund New Markets, LLC
Capital City Properties
Capital Link, Inc.
Capital One Community Renewal Fund, LLC
Capital Trust Agency Community Developm
Capmark Community Development Fund, LLC
Carolina First Community Development Co
Carver Community Development Corporatio
Central Bank of Kansas City
Central Ohio Loan Services, Inc.
Central Valley NMTC Fund, LLC
Central Valley NMTC Fund, LLC
Charter Facilities Funding, LLC
Charter Facilities Funding, LLC
```

Example #4: Reformatting string variables

By code:

- Import list of CDEs
- Take out punctuation marks, spaces, and common prefixes/suffixes
- Make everything lower case
- Clean up the last few CDE names by hand

Total time: 3 hours
Example #4: Reformatting string variables

*Import CSV file

clear

insheet using "Processing and Analysis/Importable Data/Awards.csv", comma

+Clean Up CDE names (take out spaces, periods, and commas)

rename allocatlonewardee CDE

tab CDE

gen CDE1=subinstr(CDE," ","")
gen CDE2=subinstr(CDE1,","," ")
gen CDE3=subinstr(CDE2,",","")

replace CDE=CDE3

drop CDE1 CDE2 CDE3

replace CDE=lower(CDE)
gen CDE1=subinstr(CDE,"the","")
gen CDE2=subinstr(CDE1,"llc"," ")
gen CDE3=subinstr(CDE2,"","")
gen CDE4=subinstr(CDE3,"?"," ")
gen CDE5=subinstr(CDE4,"&"," ")
gen CDE6=subinstr(CDE5,"inc","")

replace CDE=CDE6

drop CDE1 CDE2 CDE3 CDE4 CDE5 CDE6

+Manually clean up the last few names

replace CDE="advantagedcapitalcommunitdevelopmentfund" if CDE=="advantagedcapitalcommunitydevelopmentfundll"

replace CDE="esicnewmarketpartnerslp" if CDE=="esicnewmarketpartnerslp" | CDE=="esicnewmarketpartnerslplimitedpartnership"

replace CDE="lowacommunitydevelopment" if CDE=="lowacommunitydevelopmentlc"

replace CDE="localinitiativessupportcorporation" if CDE=="localinitiativessupportcorporation\lisc"

replace CDE="nationalexmarkets\taxcreditfund" if CDE=="nationalexmarkets\taxcreditfund"

replace CDE="opportunityfund\fundnorrcalifornia" if CDE=="opportunityfund\formerlendersforcommunitydevelopment\fundnorrcalifornia"

replace CDE="lendersforcommunity\developmentfund" if CDE=="nationalcitynewmarketfund"

replace CDE="shoebank\enterprisepacific\b/ashoebank\enterprisepacific\acascadia" if CDE=="shorebank\enterprisegroup\pacific\b/shorebank\enterprisepacific\ascadia"

replace CDE="rocklandtrustcommunitydevelopment" if CDE=="rocklandtrustcommunitydevelopmentcorporation"

replace CDE="universityfinancialcorp" if CDE=="universityfinancialcorporation\b/asunrisecommunitybanks"

replace CDE="reinemarketsinvestment" if CDE=="reinemarketsinvestmentcorp"

replace CDE="capitalimpactpartners" if CDE=="ncbicapi\lmpact""impact"

replace CDE="craft3" if CDE=="shorebank\enterprisepacific"

replace CDE="khcnewmarkets\csd" if CDE=="khcnewmarkets\csd\esiresa"

replace CDE="cn\developmentfoundation" if CDE=="cn\developmentfoundation"

replace CDE="wsb\communitydevelopmentcorporation" if CDE=="wsb\communitydevelopmentcorporation"

replace CDE="oakhillbanks\communitydevelopmentcorporation" if CDE=="oakhillbanks\communitydevelopmentcorporation"

replace CDE="newmarkets\redevelopment\l" if CDE=="newmarkets\redevelopment\l"

replace CDE="northfork\newmarkets\creditcorp" if CDE=="greenpoint\newmarketsslp"
It’s even faster than it looks

- Of course, in all of these examples the coding method was faster than the manual method.

- This doesn’t take into account the most important thing: **Everything is interrelated**

- If you make a change to importing the raw data, it requires re-doing everything after it: generating data, graphs, and regressions.
Project set-up

Think of every project as a set of steps:

1. Open raw data files (.dta, .csv, .txt, .raw)
2. Clean and modify the raw data
3. Combine two or more data sets to create the master data
4. Add new variables to the master data and save
5. Run summary statistics and generate graphs
6. Run canned Stata estimation routines
7. Export results to \LaTeX
Project Structure

- We’re going to use this structure of thinking about a project both in the way we organize our files and in the way we write the code:
  - Different folders for each type of file
  - Split .do files up into tasks
Your “Raw Data” folder should contain copies of all of your original data files.

Any data that were necessary for any part of the processing and/or analysis you reported in your paper should be contained in one of the original data files in your “Raw Data” folder.

Every raw data file should be saved in exactly the format it was in when you first obtained it.

Never modify these files!
Importable Data

For each of the original data files in the “Raw Data” folder, you should create a corresponding version stored in the “Importable Data” folder.

Two cases to consider:

1. The original data file is in a format that your software can open or import
   - Importable data file should be an exact copy of the original

2. The original data must be modified before your software can open or import it
   - May need to convert from original format
   - May want to modify certain variables
Command Files

- This folder should contain one or more .do files that contain every step of data processing and analysis required to reproduce the results you report in your paper.

- These scripts should include detailed comments so someone else (or future you) can understand what the code does.

- You may further organize your command files into sub-folders.

- Give your .do files descriptive names.
Command Files

- Original Data and Metadata
- Processing and Analysis
  - Analysis Data
  - Command Files
    - analyze-novoco
    - APRIL6
    - county-level
    - county-level-analysis
    - county-level-analysis-copy
    - county-level-proportions
    - county-level-proportions2
    - data-rd-graph
    - descriptive-stats-awards
    - descriptive-stats-project
    - download-qcew
    - geocode-novoco
    - Graphamount.gph
    - import-2000-2010-transition
    - import-awards
    - import-cbp
    - import-Census-2000
    - import-NMTC-eligibility
    - import-novoco
    - import-project
    - import-special-eligibility
    - import-TLR
    - lastlastminute
    - lastminute
    - make-analysis-novoco-qcew-data
This folder should contain (ideally) one data set that can be imported and used to generate all results reported in your paper.

When you generate a master .do file to replicate all of your results, this is the data set that you will import.
Getting Started

Stata/SE 12.1

Copyright 1985-2011 StataCorp LP

StataCorp
4905 Lakeway Drive
College Station, Texas 77845 USA
800-STATA-PC  http://www.stata.com
979-696-4600  stata@stata.com
979-696-4601 (fax)

Single-user Stata network perpetual license:
Serial number:  93611859953
Licensed to:  Chelsea Garber

Notes:
1. (-set maxvar-)  5000 maximum variables

Command
*Single line comment, asterisk (*) will comment out anything in the following line but nothing after and they also do not work * in the middle of the line

To start a block comment anywhere: /*

you can write comments for lines and lines until you close the block with */

to insert a comment mid-line // you use the double slash and then you can continue
Summer Stata Course
First Example Do File - "example1.do"

Author: Chelsea Garber
Date: July 5, 2016

Tasks: (1) Illustrate descriptive do file header
(2) Show commenting syntax
(3) Emphasize details and organization

Create section headers to visually break up the code

Indent every time you have a new task in a section
Indent again when you have a command under that task
Indent again when you have a subcommand under that new task
This indenting makes it easy for you to see how lines of code are grouped

Add numbers 1-10
```
di 1+2+3+4+5+6+7+8+9+10 // 'di' or 'display' command
```

Add numbers 1-50
```
di 1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+23+24+25+26+27+28+29+30+31+32+33+34+35+36+37+38+39+40+41+42+43+45+46+47+48+49+50
```

/* Coding Tip: Avoid long do files and avoid long lines within do files. It is annoying when lines look like the one above. This happens frequently when there are regressions with many variables. To get commands to wrap, use the triple slash. */

Add numbers 1-20 with text wrap
```
di 1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+23+24+25+ ///
26+27+28+29+30+31+32+33+34+35+36+37+38+39+40+41+42+43+45+46+47+48+49+50
```
Local and global variables

- Stata can store pretty much anything as a macro:
  - File paths
  - Values for colors on a graph
  - Lists of variables to run in a regression

- It can store these in two ways:
  1. Local: These are created while a do file is running and are immediately deleted once the file finishes
  2. Global: These can be created anytime and are deleted only when Stata closes, or if you enter the command `macro drop _all`
Local and global variables

* Task: Show how to create local and global macro variables *

* Local Macro
  * Set local macro
    local years 2010 2011 2012 2013 2014 2015
  * Call local macro
    di `years'
  * Loop through values of local macro
    foreach y of local years {
      di `y'
    }

* Global Macro
  * Set global macro
    global medals "gold" "silver" "bronze"
  * Call global macro
    di "$medals"
Explicit loops

Two key programming constructs for repetition:

1. **foreach**
   
   ```stata
   foreach v of varlist popgrowth lexp gnppc {
     summarize `v', detail
   }
   ```

2. **forvalues**
   
   ```stata
   forvalues i=1/10 {
     summarize PRweek`i'
   }
   ```

Note also the curly braces, which must appear at the end of their lines.
Some Mathematical Expressions

- `abs(x)` returns the absolute value of x.
- `exp(x)` returns the exponential function of x.
- `int(x)` returns the integer by truncating x towards zero.
- `ln(x), log(x)` returns the natural logarithm of x if x>0.
- `log10(x)` returns the log base 10 of x if x>0.
- `max(x1,...,xn)` returns the maximum of x1, ..., xn.
- `min(x1,...,xn)` returns the minimum of x1, ..., xn.
- `round(x)` returns x rounded to the nearest whole number.
- `round(x,y)` returns x rounded to units of y.
- `sign(x)` returns -1 if x<0, 0 if x==0, 1 if x>0.
- `sqrt(x)` returns the square root of x if x>=0.

Logical and Relational Operators

- `&` and
- `!` not
- `>` greater than
- `>=` greater or equal
- `==` equal
- `|` or
- `~` not
- `<` less than
- `<=` smaller or equal
- `!=` not equal
General Syntax

The general syntax of a Stata command is:

```
[prefix_cmd:] cmdname [varlist] [=exp]
    [if exp] [in range]
    [weight] [using...] [,options]
```

where elements in square brackets are optional for some commands.
For Now…

1. Download Stata
   - https://public.econ.duke.edu/stata/
   - Serial number: 401409002291
   - Code: f6p7 q6Lr w167 8aa6 3wpa r7pv 1886 lvog 1ox5
   - Authorization: 0xto

2. Download \LaTeX
   - https://www.latex-project.org/get/

3. Access Sakai Class Folder

4. Set Up Project Folders
Objectives - Part II

1. Import different types of raw data and save as Stata file
2. Summarize the raw data
3. Drop observations and clean your data
4. Manage variable names and labels
5. Generate new variables (continuous and dummy)
The Help Command

- Stata has extensive help available once you are in the program
  - General help - `help`
  - Help for a class of commands - `help function`
  - Help for a specific command - `help regress`
Working Directory

- First, navigate to your master project folder
  - Change directory - cd "/folder path/Your Name/Processing and Analysis"
  - Present working directory - pwd

- All input and output files will be in this common directory
  - Only provide the (relative) folder/filename rather than the complete directory structure
    - Ex: use "/Importable Data/data.dta" vs. use "/Users/Chelsea/Desktop/Duke/Summer Stata Course/Demo/Chelsea Garber/Processing and Analysis/Importable Data/data.dta"
Log Files

- A log file is Stata’s built-in tape recorder that lets you:
  1. Retrieve the output of your work
  2. Keep a record of your work

- To create a log file - `log using mylog.txt`
- To close a log file - `log close`
- To add to an existing log file - `log using mylog.txt, append`
- To replace a log file - `log using mylog.txt, replace`
Project set-up

1. Open raw data files (.dta, .csv, .txt, .raw)
2. Clean and modify the raw data
3. Combine two or more data sets to create the master data
4. Add new variables to the master data and save
5. Run summary statistics and generate graphs
6. Run canned Stata estimation routines
7. Export results to \LaTeX
Importing Data

- To open Stata (.dta) files
  - use "Importable Data/data.dta"

- To open raw (.raw) files, no extension is needed
  - insheet using "Importable Data/data"

- To open other file extensions, they must be given
  - insheet using "Importable Data/data.csv"
  - insheet using "Importable Data/data.txt"

- To import Excel files
  - import excel using "Importable Data/data.xlsx", firstrow
To save as a Stata (.dta) file
   save "Importable Data/data.dta"

To save as .csv file
   outsheet using "Importable Data/data.csv", comma

To save as Excel (.xlsx) file
   export excel using "Importable Data/data.xls"
Summarizing Data

Several commands are helpful for looking at your data

- browse br
- edit ed
- summarize sum
- describe des
- list li
- codebook
sort var1, (gsort -var1)
  Reorganize data with var1 in ascending (descending) order

tab var1, tab var1 var2
  Provides one- or two-way frequency tables

correlate var1 var2 var3
  Provides the autocorrelation table of the listed variables

scatter var1 var2
  Shows a scatter plot of the data with var2 on the x-axis and var1 on the y-axis
Project set-up

1. Open raw data files (.dta, .csv, .txt, .raw)
2. **Clean and modify the raw data**
3. Combine two or more data sets to create the master data
4. Add new variables to the master data and save
5. Run summary statistics and generate graphs
6. Run canned Stata estimation routines
7. Export results to \text{LaTeX}
Cleaning Your Data

- **preserve**
  - Takes a snapshot of your data and keeps it in memory

- **restore**
  - Reverts back to the preserved version of the data

- **clear**
  - Clears all data in memory

- **drop var1**
  - Removes var1 from the data

- **keep var1**
  - Removes every variable *except* var1 from the data
A Note on Missing Values

- The codebook command gives a sense of how many missing values each variable has.

- For string variables, a missing value is indicated by an empty cell.
  - We can refer to this value as ""

- For numerical variables, a missing value is indicated by a single dot.
  - Caution: This period (.) actually takes on a very large number according to Stata.
    - Be careful with relational operators!
Data Management

- To rename a variable
  - `rename oldvar newvar`

- To recode a (numerical) variable
  - `recode var1 (old value = new value)`

- To replace the value of a variable
  - `replace var1=. if var1==999`
  - `replace var1="new text" if var1=="old text"`

- To label a variable
  - `label variable var1 "This is my label"`
Stata’s Internal Variables

- Stata keeps track of its own variables
  - `return list`

- More variables are stored following a command
  - `sum var1`
  - `return list`

- Post-estimation information is also stored
  - `regress var1 var2`
  - `ereturn list`
By-processing

- You can run a different command for different subsets of the data using the `bysort` prefix
  - `bysort category: sum var1`

- The result is the same as writing a list of `sum` commands with separate `if` statements for each category
  - `sum var1 if category==1`
  - `sum var1 if category==2`
  - `sum var1 if category==3`
  - `sum var1 if category==4`
  - `sum var1 if category==5`
Creating New Variables

- We can create a host of new variables from the existing data with the `gen` command

```
. gen realgdp=(pop*1000)*cgdp /* real GDP in current prices */
. gen lpop=ln(pop) /* log population */
. gen popsq=pop^2 /* squared population */
. gen ten=10 /* constant value of 10 */
. gen id=_n /* id number of observation */
. gen total=_N /* total number of observations */
```

- The `egen` command typically creates new variables based on summary measures

```
. egen totalpop=total(pop), by(year) /* sum of the population per year */
. egen avgpop=mean(pop), by(year) /* average country pop per year */
. egen maxpop=max(pop) /* largest population value */
. egen countpop=count(pop) /* counts number of non-missing obs */
. egen groupid=group(country_code) /* generates numeric id variable for countries */
```

- These can be combined with the `bysort` prefix
There are three ways to create dummy variables

1. Use `generate` and `replace`
   - `gen largevalue=0`
   - `replace largevalue=1 if value>99999 & value!=.`

2. Use `tab` and `gen` option
   - `tab category, gen(cdummy)`

3. Use the `xi` prefix as a command
   - `xi i.category`
   - `xi i.category, noomit`
Exercises

1. Download the Stata example dataset auto.dta using the sysuse command.

2. Obtain summary statistics for mpg and weight according to whether the car type is foreign.

3. Write a do-file to repeat the previous question. This do-file should include a log file. Continue assignment in this do file.

4. Obtain summary statistics for the price variable. Then use the results stored in r() to compute a scalar, cv, equal to the standard deviation divided by the mean of price.

5. Define a global macro named varlist for a variable list with mpg, price, and weight, and then obtain summary statistics for varlist. Repeat this exercise for a local macro named varlist.

6. Figure out what the rep78 variable is. Drop any observations that are missing this value. Save dataset as auto2.dta.
Exercises, Part 2

1. Generate a new variable equal to the sum of headroom and length.
2. Generate a new variable equal to the average mpg across all cars.
3. Generate a new string variable company
   - Split make at the space.
   - Hint: Use the help split command to explore how this works.
4. Create a variable that counts how many makes each company has.
5. Create a new variable equal to the average mpg within each company. Which company has the best average mpg?
6. Create a dummy variable expensive that indicates cars with prices over 10,000.
7. Create a variable equal to the total number of observations.
8. Create an index variable that numbers the observations.