

Stata Workshop: Statistical Analysis Guide

1 Introduction

This guide provides a step-by-step explanation of the Stata commands used in the workshop.

2 Setting Up the Environment

2.1 Install Packages

For today we will need three user-created packages from online. Packages like these are super useful, and can be installed very easily with an internet connection. Other languages like R or Python also have packages like these. When you are coding in a statistical language you will often be using such packages.

```
ssc install ivregress2
ssc install outreg2
ssc install wbopendata
```

2.2 Something You Should Do at the Beginning of Every Stata Session

```
clear all
set more off
```

This clears all the data currently in memory - something you should do in any coding language, but Stata will also specifically not let you load in a new dataset unless you clear the old one from memory. `set more off` just allows you to run code that has a lot of output and not have to keep clicking "show more" every time Stata runs out of room.

2.3 Stata Version

```
version 15
```

You don't need to run this command today (as we are all working in Stata 15), but it ensured that Stata runs in compatibility mode for version 15 while I was preparing this material.

2.4 Setting the Working Directory

```
cd "C:\Users\s226p763\OneDrive - University of Kansas\Stata Workshop"
```

You should change this to your own directory to save exported files later.

3 Loading and Exploring a Dataset

3.1 Loading the Auto Dataset

```
wbopendata, language(en - English) topics(3 - Economy & Growth)
```

This command loads in the World Bank data we have been working on in other Stata workshops.

3.2 Describing the Dataset

```
describe
```

Provides metadata about the dataset.

3.3 Data Cleaning!

Don't worry too too much about the commands in these steps. They are mostly to get the data in a more usable state. I can go into more detail about each step though if you are curious.

Drop all aggregate statistics (we just want countries):

```
drop if incomelevelname == "Aggregates" | countryname == "Africa Eastern and Southern"  
| countryname == "Africa Western and Central"
```

Drop all variables we don't need:

```
drop countrycode region adminregion adminregionname incomelevel lendingtype lendingtypen  
indicatorcode
```

Reshape dataset into long format:

```
reshape long yr, i(indicatorname countryname) j(year)
```

Create new variables for each indicator:

```
gen inflation = yr if indicatorname == "Inflation, consumer prices (annual %)"
```

```
gen services = yr if indicatorname == "Services, value added (% of GDP)"
```

```
gen manufacturing = yr if indicatorname == "Manufacturing, value added (% of GDP)"
```

```

gen capital_growth = yr if indicatorname == "Gross capital formation (annual %
growth)"
gen capital = yr if indicatorname == "Gross capital formation (constant 2015 US$)"
gen consumption_growth = yr if indicatorname == "Final consumption expenditure
(annual % growth)"
gen consumption = yr if indicatorname == "Final consumption expenditure (constant
2015 US$)"
gen fdi = yr if indicatorname == "Foreign direct investment, net inflows (BoP,
current US$)"
gen gdppc = yr if indicatorname == "GDP per capita (constant 2015 US$)"

```

Format variables into correct format:

```

format inflation services manufacturing capital_growth capital consumption_growth
consumption fdi gdppc %20.2f

```

Collapse so that we have the dataset on a year, country level:

```

collapse (max) inflation services manufacturing capital_growth capital consumption_growth
consumption fdi gdppc, by(countryname regionname incomelevelname year)

```

3.4 Save the dataset

```
econ_stats, replace
```

This is so we can reload the data easily and we don't have to go through the long data cleaning steps often.

4 Summary Statistics

4.1 Basic Summary Statistics

```
summarize *
```

Displays summary statistics for all numeric variables.

4.2 Detailed Summary for a Specific Variable

```
sum inflation, det
```

Provides detailed summary statistics for inflation.

5 Tabulations

5.1 One-way Frequency Tables

```
tab incomelevelname
```

Generates a frequency table for income level.

5.2 Two-way Frequency Tables

```
tab incomelevelname regionname
```

Generates a frequency table for income level and region.

5.3 Tests for Independence

```
tab infl_high gdppc_high chi2
```

Tests the association between high inflation and high GDP per capita. Can also use different tests,

5.4 Using tab to create dummy variables

```
tab incomelevelname, gen(cincome)
```

Creates dummy variables for each value of income level. Useful for creating dummies quickly. Can be helpful for bar charts in my experience.

5.5 Apart from tabulate you can use table for more complicated analyses

```
table incomlevelname regionname, c(mean consumption median manufacturing median services)
```

Gives the requested summary statistics for each value of and foreign.

5.6 We can add a third variable to the tabulation

```
table incomelevelname regionname infl_high, c(mean consumption median manufacturing median services) f(%20.2f)
```

Gives the requested summary statistics for each value of income level, region and high inflation, and shows you how to format the resulting statistics.

6 Regression Analysis

6.1 Encoding variables to use as factors

```
label define income_label 1 "Low income" 2 "Lower middle income" 3 "Upper middle  
income" 4 "High income"  
encode incomelevelname, gen(income) label(income_label)  
encode regionname, gen(region)  
encode countryname, gen(country)
```

This just allows us to treat these string variables as factor variables, allowing for easier use in some of the commands to come.

6.2 Basic Regression

```
reg gdppc inflation
```

Runs a simple regression of GDP per capita on inflation.

```
reg gdppc inflation services, vce(robust)
```

Adds services as a covariate with robust standard errors.

6.3 Save estimates

```
reg gdppc inflation i.region, vce(robust)  
est store spec1
```

```
reg gdppc inflation i.income i.region, vce(robust)  
est store spec2
```

```
est restore spec1  
est replay  
est clear
```

We can use `estimates store` to store a regression's estimates in local memory so we can use them later without having to run the regression again. We can also use `estimates replay` to replay the estimates, and `estimates save` to save the estimates to disk if we needed to load them in in other code or something.

6.4 Test hypotheses about coefficients

```
reg gdppc services manufacturing i.income i.region  
reg gdppc services manufacturing i.income i.region, coeflegend
```

One option that is really useful if you are planning on referencing coefficients within a regression is "coeflegend" which will identify each coefficient by how it is stored in Stata.

```
test(_b[services]=150)
```

This tests whether the coefficient for services is equal to 150.

```
test(_b[services]=_b[3.income])
```

This tests whether the coefficient for services is equal to the coefficient of the dummy for income level = 3 (Upper middle income).

We can also use `suest` (seemingly unrelated estimation) to test hypotheses about coefficients from different regressions. We will run two regressions, one for nations in East Asia and Oceania and another for nations in Europe. When we do this, the omitted category for a variable may change because it could be missing on one of the subsets so, first, we need to fix the omitted category of turn to something that shows up in both regressions. In this case, it's lower middle income (2 in the factor variable).

```
fvset base 2 income
```

Now we run the regressions, then use `suest` to combine the estimation results, and then we test whether the coefficient for services is the same in both.

```
reg gdppc services manufacturing i.income if region == 1
est store east_asia
reg gdppc services manufacturing i.income if region == 2
est store europe
```

```
suest east_asia europe
test([east_asia_mean = europe_mean]: services)
```

6.5 Generating Predictions

```
predict yhat, xb
```

Creates predicted values for the dependent variable based on the estimated coefficients.

```
predict rhat, residuals
```

Generates residuals (the difference between observed and predicted values).

6.6 Predictive Margins

```
reg gdpcc c.inflation##i.income, vce(robust)
margins, dydx(inflation)
margins income, dydx(inflation)
```

Calculates predictive marginal effects of inflation on GDP per capita. That is, it is the predicted change in GDP per capita given a one-unit change in inflation, evaluated at the mean of inflation. We can also look at the predicted change of a one unit increase in inflation for different subsets, in this case different levels of income.

We can also ask for the estimated effect at different values of mpg:

```
margins, dydx(mpg) at (mpg=(10(10)40)) plot
```

6.7 Contrasts and Margins Analysis

```
fvset base 1 income
```

Sets low income (1 in the factor variable) as the reference level for income.

```
reg gdpcc c.inflation##i.income, vce(robust)
margins income
```

This simply shows the predicted gdpcc for each value of income.

```
reg price c.mpg##i.turn, vce(robust)
margins r.income
margins a.income
margins ar.income
```

Performs contrasts, analyzing differences from the reference category, then with respect to the next category, and then with respect to the previous category.

7 Panel Data

7.1 Setting Up Panel Data

```
xtset country year
```

Sets up our data as a panel dataset. The first argument to xtset is the individual ID (the variable that uniquely identifies each unit), and the second is the time variable.

7.2 Fixed-Effects Regression

```
xtreg gdpcc inflation services manufacturing fdi consumption capital, fe vce(robust)
```

Runs a fixed-effects regression. The "fe" option will correctly identify year and country as the fixed effects of interest.

7.3 Exporting Results

```
reg gdpcc inflation, vce(robust)
outreg2 using simple_regs.xls, excel ctitle("Baseline - OLS") addtext(Region FE,
No, Year FE, No) label replace
```

Exports regression results to an Excel file. Can also do txt files, Latex files, etc.

```
xtreg gdpcc inflation, fe vce(robust)
outreg2 using simple_regs.xls, excel ctitle("Baseline - FE") addtext(State FE,
YES, Year FE, YES) label append
```

Appends results to the same Excel file instead of replacing it.

outreg2 has many options for deciding how to output your results. Below is a slightly more complicated case: we drop manufacturing from the list of variables we want, set the number of decimals to 3, and we asked for t-statistics below the coefficients, instead of standard errors.

```
xtreg gdpcc inflation services manufacturing fdi consumption capital, fe vce(robust)
outreg2 using simple_regs.xls, label ///
ctitle("A more complicated example") excel drop (manufacturing) dec(3) ///
title ("Outreg2 Example") stats(coef tstat) append
```

7.4 Panel dataset with unbalanced panel

```
webuse nlswork, clear
xtset idcode year Load in dataset from Stata default datasets, and set panel data.
gen L1ln_wage = L.ln_wage
gen L2ln_wage = L2.ln_wage
gen Dln_wage = D.ln_wage
xtdescribe
```

We can see the problems with gaps in the data, and why the xtset was pretty necessary! Otherwise we would be taking the wrong lags.

```
xxtab occ_code
```

```
xttab race
```

Useful commands for showing how often a person shows up in a particular category. Pretty easy shortcuts for some relatively complicated stuff.

8 More Detailed Causal Methods

8.1 Differences-in-Differences

```
webuse hospdd, clear
```

```
xtset hospital
```

Brings in new dataset.

```
xtdidregress (satis)(procedure), group(hospital) time(month) vce(robust)
```

Easy shortcut for a DID regression that estimates the average treatment effect of a new admissions procedure on the satisfaction of patients subject to the new procedure.

```
xtdidregress (satis)(procedure), group(hospital) time(month) vce(cluster hospital)
```

Add clustered standard errors.

8.2 Instrumental Variables

First let's create some simulated data to check how IV is working.

```
clear
```

```
set obs 1000
```

```
gen z = rnormal (5, 3)
```

```
gen conf = rnormal (12, 3)
```

```
gen x1 = 2 + 3z + 5*conf + rnormal ()
```

```
gen x2 = rnormal(2, 4)
```

```
gen y = 10 + 3x1 + 5x2 + 2*conf + rnormal()
```

Generates synthetic data where z is an instrument for x_1 , $conf$ represents unobserved confounders, and x_2 is some other variable. Note that z affects y only through x_1 , and that both x_1 and y are functions of $conf$.

```
ivregress 2sls y x2 (x1 = z)
```

Performs a two-stage least squares regression where x_1 is instrumented by z . Note that in parenthesis you have to specify the endogenous variable before the equals sign, and the exogenous variables to the right of the equals sign.