Econometrics
& Data Science

Duncan Thomas
Economics Department
Duke University

STATA
A BRIEF INTRODUCTION

Table of contents

Overview 1
Running STATA 1
Using data 2
Looking at data 3
Summarizing data 3
Selecting subsets of data 4
Manipulating data 5
Merging datasets 6
Summarizing data graphically 6
Relationships among variables 7
Regression analysis 7
Extensions to the regression model 8
Keeping a copy of results 10
Keeping track of your work 10
Documenting your programs 10
Shortcuts: Macros and loops 11
Documenting your data 12

This document can be retrieved from http://ipl.econ.duke.edu/dthomas/ec204/ho-stata.pdf . It has been prepared for the econometrics sequence in the Duke Economics Department (Econ 104 and 204). Input from Ian Bailey, Rosie Chiang, Damien Kim, Jeremy Lebow, Richard Lombardo, Hannah O' Sullivan, Andrew Patton, Tony Sun, Tony Wang and William Zhao has been very helpful.

August 2023

Overview

STATA is easy to learn, intuitive, powerful and very flexible. It will serve you very well for this class, empirical research in other economics courses, your own empirical research papers and in your career beyond your degree. It is very widely used in the social and health sciences and has an enormous and very active user base that is continually developing new tools. The product is frequently updated to assure that it is on the frontier of econometric practice and empirical methods. In short, it is an outstanding product. STATA is a lot like R and shares many of the same features although the syntax is substantially more intuitive than R.

STATA has exceptional online help and a menu-driven interface that you can use to learn the command syntax. The best text reference is the STATA manual, which is available on-line (www.stata.com/features/documentation/). The STATA web site (www.stata.com) also has extensive help pages, tutorials and links to helpful resources. There is a huge amount of useful material on-line spread across many sites; I link to some websites on the class web page. Finally, *Statistics with STATA*, by L. C. Hamilton, provides a very accessible introduction to STATA.

This note is intended to help you get started with STATA. It is not a substitute for your own experimentation; nor is it a substitute for using the extensive on-line resources and reading the manual. It should, however, provide you with the tools you will need to *begin* working with STATA and completing the problem sets for this class; you should aim to develop some familiarity with the product and understand the logic underlying it. For clarity, the STATA commands in this note are written in **bold** and example code is in *italics*. Menu options are underlined. When using STATA, all commands are in lowercase (everything is case sensitive including variable names).

Getting your own copy of STATA

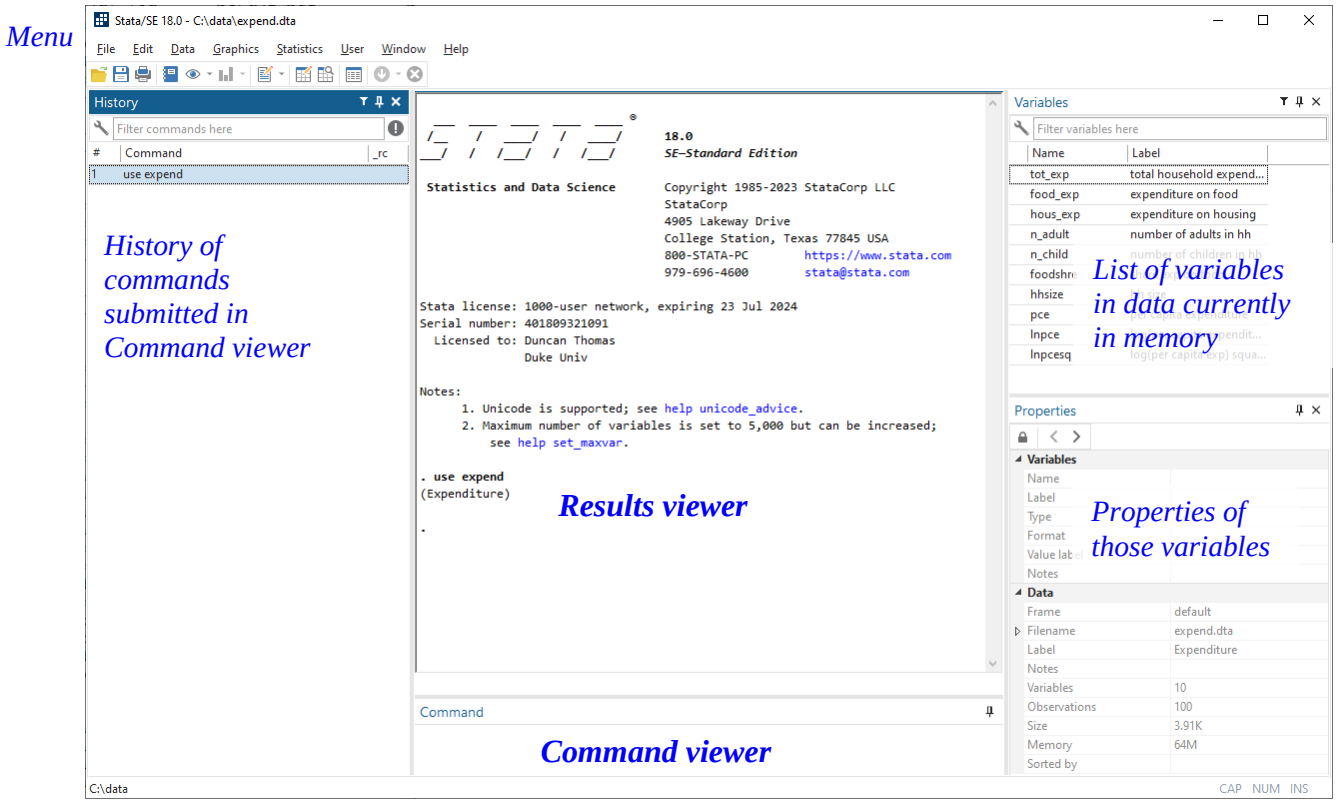
A site license has been purchased so that you can use STATA on your own computer. You may use the license for this class, other classes you are taking this year and for your research. The license is valid for the entire academic year.

To obtain a copy of STATA/SE, go to the class web page and click on "Links for STATA." Please do not share the information on the website with people who are not enrolled in this class. The site license is limited and sharing the license information will jeopardize future use of the license. Having it cancelled would be a total pain as you will have to purchase your own license or use STATA in one of the campus labs.

To buy a copy of STATA, go to www.stata.com. You (or anyone at Duke) can buy STATA under the GradPlan which offers a deep discount to students at Duke.

Running STATA

The easiest way to invoke STATA is to click on the STATA icon. You will see a screen that looks something like the screen below. (The default screen varies across platforms and versions of STATA; the screen is customizable.)



The large window in the center is the *Results viewer* where your commands are echoed and results appear. You type in commands in the window below it, the *Command viewer*. A history of your commands is recorded in the left window. Information about data in memory appears in the right windows; a list of variables in the upper window and information about specific variables in the lower window. A *Menu* is at the top left. [2]

Click on Help in the menu bar that runs along the top of the STATA screen or type **help** in the Command window. A window will open and you can select from the help items. Click on PDF Documentation to load the STATA manuals. They are all hyperlinked and easy to use. You can search for a command by clicking on Search or type **search <topic>** in the Command window where <topic> is some topic you want to learn about. For example, **search *regression*** will provide you with a list of ways to run a regression in STATA. If you know the name of a command, type **help <command>**. For example, **help *regress*** will give you the help information about the STATA command **regress** which is used for regressions. Typically the help information includes really helpful examples that illustrate how you can use the command.

Using data

An example STATA dataset, is available here. (Alternatively, go to the Fall 204 class webpage, click on STATA at Duke to get a copy of STATA, Intro to STATA to get his handout and Example dataset to get the example dataset, **expend.dta**.

[2] The default layout of the screen differs depending on the operating system. The layout can be customized by moving the windows around then saving the layout by clicking on Edit then Preferences.

You will be asked where you want that file to be located on your computer. Assuming that you copy the dataset into your current directory then you can load the data into memory with the **use** command: **use *expend***. Since the default extension for a STATA dataset is ".dta", you do not need to type that. If the data are not in the current directory, you need to navigate to the directory (use **cd *datafolder*** in the command viewer to change your directory to *datafolder*) or include the path after **use,** e.g. **use */path/dataset***.
If you get a message that there is data already in the memory when you try to **use** data, you must include the option **clear**, i.e. **use *expend,* clear** which will clear all data from memory prior to loading expend.dta . This means that anything that is in memory is cleared out and cannot be retrieved. Use **save** to save your data. If the data exists and you want to overwrite those data, use the option **replace** with **save**, i.e. **save *expend,* replace**. Use the **frame** command if you want to hold multiple datasets in memory.

There are many ways to import data into STATA. Click on File in the menu at the top of the screen, then click on Import to see the data types that can be directly imported from another product. You can also read data from a text file. Type **help infile**.[3] You can also export data to other products. Click on File and then click on Export.

Looking at data

To see what variables are in a dataset, use **describe** which displays a list of all variables and their attributes including the variable label which tells you about the variable. This information is displayed in the Variable viewer (in the top right of the STATA window). The command **describe** also tells you how many observations (individual records) are in the dataset.

To look at the *value of each observation*, you might **list** the variable(s). For example, in the Command viewer, type **list *hhsize***. Instead of typing out the name of the variable, you can type **list** in the Command viewer and then click on the variable in the Variable viewer and the name will appear after **list**. Then hit return. You can list all variables by saying **list** and hitting return. (In general, for any command, if you do not specify a set of variables, STATA will execute the command on all variables.) You can also **list** a range of variables. Say you have var1 var2 var3 var4 in the dataset, then var1-var3 will list var1 var2 var3. The order of variables is given by the storage order (as given in the list of variables in the Variables viewer or when you **describe** the dataset).

You can also **browse** and **edit** the data. Click on Data in the menu in the top left and click on Data Editor.

Hit the spacebar when the screen fills and you will get the next screen. Hit CTRL-C or BREAK if you want to stop the list (or any other STATA function from continuing to display results). You can retrieve the last command in the Command window by hitting PageUp or CTRL-R. Click on a command in the Review window and you will get the same effect.

Summarizing data

STATA will **summarize** the data. Type **summarize <variable>** (or just **summarize** for all variables in the dataset). In the Results window, you will see the number of observations, mean, standard deviation, minimum and maximum value of each variable. If you want order statistics, then include the option **detail**

---

[3] If you have data in R format (mydata.Rdata, for example), export it as a STATA dataset with the R commands:
> library (foreign)
> write.dta (mydata,, "mystata.dta")
Executing these commands in R will create a copy of mydata.Rdata called mystata.dta in STATA format.

(after the list of variables separated by a comma) and you will also get the median, bottom and top quartile, a couple of other order statistics (percentiles) as well as the smallest and largest 5 observations in the dataset. For example, **summarize *tot_exp*, detail** will display summary statistics for the variable tot_exp.

In general, options in STATA commands are written at the end of the command usually following a comma. There are, however, some important exceptions. For example, in some cases you will want to calculate statistics with weights. There are many different types of weights and STATA treats them differently. Type **help weights** for information. If you want to calculate weighted statistics, then you need to include the option **[pweight=**variable name] (if you are using 'sampling' weights) or [**fweight**=variable name] (if you are using 'frequency' weights). For example, **summarize *pce* [fweight=*hhsize*], detail** will display detailed summary statistics for the variable pce that is weighted by the variable hhsize.

If the variable is discrete (or categorical) then it will typically take on only a limited number of values. In this case, you might find a frequency tabulation more useful than measures of central tendency: use **tabulate**. The table reports the absolute frequency (Freq.), relative frequency (Percent) and cumulative relative frequency (Cum). The **plot** option will print out a histogram type plot alongside the table. If you have missing observations, then they are suppressed from the frequency count; if you want to include them, add the **missing** option. Use **tabulate *var1 var2*** to create cross-tabulations of two variables. Use **table** for multi-way classifications.

Say you have two variables, one discrete and one continuous (such as hhsize and tot_exp). You want summary statistics of tot_exp for each hhsize: you have at least two options. First, **sort** by the discrete variable (hhsize) and then precede the **summarize** command with **by <varname>:** e.g. first **sort *hhsize***. Next, **by *hhsize*: summarize *tot_exp***. A short cut allows you to do this in one step using **bys** which will sort the variable for you when it calculates the summary statistics. That is, **bys *hhsize*: summarize *tot_exp*.**

Other options are **tabsum** (for calculation of means) or **table** (for other statistics). Type **help summarize**, **help tabulate**, **help tabsum** and **help table** for more information on these commands.

Selecting subsets of data

Sorting is immensely useful: if you plan to do several analyses on subsets of a particular dataset, then sorting may be a good strategy. Virtually all the STATA utilities permit analysis of subsets of the dataset with the **by** option. Note, however, that to use the **by** option, the data *must* be sorted by that variable. If you use the shortcut **bys variable: command** then the data will be sorted by variable prior to running the command.

If you want to restrict attention to a particular subset of observations, then you might use the **in** or **if** option. For example, say you want to restrict attention to the first ten observations: to list them, use **list** with the option **in 1/10.** In general, the option is **in** start_obs/stop_obs. Alternatively, you may be interested only in those households with no more than 4 members. To list them, **list** variable **if** hhsize <=4. You can use any relational operator here: (> < >= <= not equal is ~= and equal is == which distinguishes it from the algebraic = sign). So if you want to list tot_exp if hhsize is equal to 1, then **list tot_exp if hhsize==1**.

Manipulating data

You will want to create new variables, manipulate them, edit values of variables and drop variables.

You may **drop** any variable by typing **drop** variable_names. If you want to drop all variables, use the STATA reserved variable, **_all**. If you want to **keep** a few variables, it would be more efficient to list those variables **keep** variable_names.

If you want to generate a new variable, you need to **generate** new_variable = some manipulation of old_variables. For example, to create the square of total expenditure (tot_exp2 = tot_exp*tot_exp), **generate *tot_exp2 = tot_exp*tot_exp*** or **generate *tot_exp2 = tot_exp^2.*** In general, you may add [+], subtract [-], multiply [*] or divide [/] variables. For any variable (var_name) you can also compute the absolute value [abs(var_name)], logarithm [log(var_name)], exponent [exp(var_name)], square root [sqrt(var_name)] and integer value [int(var_name)]. You can, of course, combine all these operators and also use the **if** and **in** options if you want to work on a subset of the data.

You cannot generate a variable that already exists. If you want to change an existing variable, you must **replace** it. For example, say you want to divide tot_exp2 by 1000, then **replace *tot_exp2 = tot_exp2/1000***. Again, you can use the **if** and **in o**ptions if you only want to replace particular observations: this is very useful for fixing errors in the data. (See also the **edit** command). As another example, say you want to create a variable which takes the value 1 if household size is greater than 4 and it takes the value 0 otherwise. You can **generate** hh_gt4 which is 1 **if** hhsize > 4. Then **replace** hh_gt4 which is 0 **if** hh_gt4==. Notice that hh_gt4 is set to the missing value, '.' when hhsize is 1 through 4 and you use the double equals to replace those dots with zeroes. Thus:

generate *hh_gt4* = 1 if *hhsize* > 4
replace *hh_gt4* = 0 if *hh_gt4* == .

An alternative that has the same effect would be to replace hh_gt4 if hhsize is less than or equal to 4:
replace *hh_gt4* = 0 if *hhsize* <= *4*

There is a very useful shorthand built into STATA

generate *hh_gt4*= (*hhsize* > 4)

which has the same effect as the commands above. In this case, if the statement in the parentheses is true, hh_gt4 is set to 1 and if the statement is not true, hh_gt4 is set to zero.

You can use the same type of conditional statements to **drop** or **keep** observations from the dataset. For example, **drop if *hhsize* > *4*** will drop all households with more than four members.

An extremely useful command in STATA is **assert**. If you assert some expression is true, STATA will tell you if that expression is correct. For example, **assert *hhsize* <= *4*** will be true after you have executed the DROP statement in the last paragraph. This is a good way to check that you have done to the data what you think you did. It is also good way to check any data you use (and manipulate) follow the patterns that you are expecting.

Merging datasets

If you wish to merge two STATA datasets together, you need a variable that is common in both datasets that will serve as the key for the merge. Call that variable ***common*** which is in both mydata1.dta and mydata2.dta. After sorting both datasets by ***common***:

use *mydata1*
merge 1:1 *common* using *mydata2*

will merge mydata1 and mydata2. This **merge** statement tells STATA to do a 1 to 1 merge (the **1:1** part of the statement) which means there are unique values of common in mydata1 and unique values of common in mydata2. If every value of common is in both datasets, every row of the dataset after the merge will have data from both mydata1 and mydata2. If there are values of common that are only in one or the other dataset (say mydata1), the data after the merge will include data for that row from mydata1 that has these values of common and insert missing values for the variables from the other dataset, mydata2 in this case. To let you know from where data are drawn, STATA creates a variable **_merge** that takes the value 1 if data are drawn only from mydata1, 2 if data are only from mydata2 and 3 if data are drawn from both datasets.

If there are multiple values of common in mydata1 and unique values in mydata2, then do an m:1 merge

merge m:1 *common* using *mydata2e*

and if there are multiple values of common in mydata2 and unique values in mydata1, do an 1:m merge

merge 1:m *common* using *mydata2*

Since you cannot have two variables with the same name in a STATA dataset, if **_merge** exists in one of your datasets, you will not be able to merge the data sets together. You must **drop** (or **rename**) **_merge**, the variable that STATA created in the last merge, before you try to execute another merge.

Summarizing data graphically

It is very easy to get simple graphical descriptions of data using STATA. To make pictures very pretty takes more time: you should look at the options available to you (by using the menu, reading the manual or using help). In this note, I will go over only the simplest plots (which we will use) and a couple of potentially useful options.

Histograms

histogram <variable_name> will draw a histogram with 5 bins and the axes identifying the largest and smallest values. You can change the number of bins with the **bin(b)** option, where b is the number of bins you want (and must be less than 50). You can also define the axes with the **xlabel(value1 value2 value3)** and/or **ylabel(value1 value2)** options. The x-axis will identify three values and the y-axis 2 values in this case. If you want the absolute frequency (instead of the relative frequency) on the y-axis, use the **freq** option. For example, **histogram *tot_exp* , bin(*10*) xlabel(*0 2000 5000 10000 15000*) ylabel(*0 20 40*) freq**.

Other univariate graphical displays

stem <variable_name> will create a stem and leaf plot; **graph box <variable_name**> will create a box and whisker plot.

Two-way graphical displays

There are a lot of ways to display the relationship between two (or more) variables in STATA. For example, **twoway (scatter *variable1 variable2*)** will display a scatter plot of variable1 and variable2. **twoway (connect *variable1 variable2*)** will connect the points with a line. You can control the symbol used for observations, the way they are connected, the labels on axes. You can draw bar graphs, pie charts and star graphs and you can have matrices of scatterplots. You can also combine several graphs together. See the graphics chapter in the manual or use the Graph option on the command menu.

Relationships among variables

To compute correlations among a group of variables, **correlate** variables. Clearly you must have at least two variables. If you do not specify any variable names, then the correlations among all variables in your dataset will be calculated. The **covariance** option will result in covariances being calculated instead of correlations. If you use the **means** option, then the output will include the mean, standard deviation, minimum and maximum of each variable.

Regression analysis

It is extremely easy to run regressions with STATA. **regress <y x>** will produce estimates of the regression $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ You may enter as many independent variables, x, as you like.

For example, **regress** ***food_exp tot_exp hhsize***

```
. reg food_exp tot_exp hhsize
```

Source	SS	df	MS	Number of obs	=	100
				F(2, 97)	=	94.84
Model	29939920.9	2	14969960.4	Prob > F	=	0.0000
Residual	15310424.8	97	157839.431	R-squared	=	0.6617
				Adj R-squared	=	0.6547
Total	45250345.7	99	457074.199	Root MSE	=	397.29

food_exp	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
tot_exp	.2129672	.0190884	11.16	0.000	.175082	.2508524
hhsize	65.90173	15.27374	4.31	0.000	35.58758	96.21587
_cons	298.4136	94.07863	3.17	0.002	111.6936	485.1337

The output includes a column of the model (or explained) sum of squares (in the SS column), residual and total sum of squares. The means of these statistics are in the mean square (MS) column. Some summary statistics are printed in the final column of the upper panel; these include R^2, adjusted R^2 and the root mean squared error (MSE). The estimated slope coefficients are in the lower panel in the first column next to each variable name. The intercept is indicated by the name _cons. The other columns in each row provide information about variability of the estimate in the first column.

You can use the **by**, **in** and **if** options to subset the data. The **[pweight=**variable name] option permits you to run a weighted regression.

You will often want to look at predicted values of the dependent variable. After running a regression, **predict** ***variable_name*** will create a variable called variable_name which contains the predicted value of the dependent variable. To obtain residuals, use **predict** ***variable_name,*** **residual**. You can then use **summarize** or the graph options to examine these data.
For example, after

regress ***food_exp tot_exp***

you can

predict ***yhat***
predict ***uhat,*** **residual**

sum *yhat uhat*, detail
sort tot_exp
twoway (scatter food_exp tot_exp) (connect yhat tot_exp)

The first statement estimates the regression plane relating food expenditure to total expenditure and household size. The next command predicts the value of food expenditure for each observation in the dataset (predicted value of the dependent variable) and the third command calculates the predicted residual for each observation. The fourth command summarizes those new variables. The fifth command sorts the dataset by tot_exp from lowest to highest value. The sixth command creates a graph with a scatter plot of the bivariate relationship between food expenditure and total expenditure along with the predicted regression line overlaid on the scatter plot.

Extensions to the regression model

One of the key strengths of STATA is that it provides many options for estimating regression-type models. A small sub-set of those options are introduced in the next section.

Hypothesis testing

It is straightforward to test hypotheses about coefficients in regression models. Consider a model that relates food expenditure to total expenditure:

regress *food_exp tot_exp*

To test whether the coefficient on tot_exp is equal to 1, use the test command

test tot_exp=1

Extend the model to include the square of total expenditure, tot_expsq, as a covariate:

regress *food_exp tot_exp tot_expsq*

To test whether both tot exp and tot_expsq are jointly significant, use this form of the test command

test tot_exp tot_expsq

Estimation of variance of coefficient estimates

The optimality of OLS relies on several assumptions. One is that errors are all drawn from the same distribution (which has a finite variance). This assumption can be tested using the ***estat hettest*** command after estimation of the model:

regress *food_exp tot_exp tot_expsq*
estat hettest, fstat

reports an F test statistic to test the assumption that the errors are homoskedastic.

Heteroskedastic errors If that assumption is rejected, the standard errors on coefficient estimates are likely to be wrong (typically too conservative). A robust (Huber-White or sandwich) estimator can be used to calculate standard errors that take this into account. Add the ***vce(robust)*** option to the regression statement

regress *food_exp tot_exp tot_expsq, vce(robust)*

Autocorrelated errors OLS also assumes that residuals are uncorrelated with one another. That assumption may be violated (say because the residuals share some common factors (are clustered) or because of serial correlation in the residuals). One approach to taking clustered unobserved factors into account in the estimation is to use:

regress *food_exp tot_exp tot_expsq, vce(cluster groupvar)*

where ***groupvar*** is a variable that identifies the groups or clusters of residuals.

Jackknife and bootstrap In some cases, you may prefer to use a non-parametric approach to estimate the variances of the coefficient estimates such as the jackknife or bootstrap. The options for **vce** are:

regress *food_exp tot_exp tot_expsq, vce(jackknife, nodots)*

and

regress ***food_exp tot_exp tot_expsq, vce(bootstrap, reps(#reps) nodots)***
respectively where **nodots** suppresses the dot on the output for every replication.

Quantile regression
The linear regression model provides an estimate of the conditional mean of the dependent variable, that is the mean of the dependent variable conditional on the values of the covariates. In some cases, an investigator in interested in other parts of the conditional distribution of the dependent variable. Quantile regression can be estimated with **qreg** ***y x*** **, q(*quantile*).** The median regression (quantile=50) is the default and provides the least absolute deviation estimator, e.g. **qreg** ***food_exp tot_exp*** **,** ***q(50)*** where the option **,** ***q(50)*** is unnecessary because it is the default. The median is a robust indicator of central tendency relative to the mean; analogously, median regression (also called least absolute deviation) is a robust approach to regression estimation relative to OLS.

Discrete dependent variable models
When the dependent variable is binary (takes on only 2 values), you might use OLS, **probit** or **logit** depending on the assumptions you make about the distribution of the unobserved characteristics in the model. To report odds ratios in the logit model, use the **or** option. If the discrete dependent variable takes on more than two values, you might use the multinomial analogue of probit and logit (**mprobit**, **mlogit**, respectively). If the outcomes are ordered, you might use the ordinal analogues (**oprobit** and **ologit**, respectively). It is often useful to examine marginal effects in these models; use the **margins** statement.

Instrumental variable estimation
Consider the model in which $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ where x_i and ε_i are potentially correlated. The OLS estimates of β_0 and β_1 will be biased. If there is an instrument, z_i, that is correlated with x_i and not correlated with ε_i then the instrumental variables estimate of β_0 and β_1 will be unbiased. Use **ivregress** to estimate this model using instrumental variables. To use the method of two stage least squares, estimate **ivregress** ***2sls*** **y (x=z)** . You can have many covariates, x, as you like. Say you have x_1, which you know is not correlated with ε, and x_2 which is potentially correlated with ε. Then you do not want to predict x_1 in the model and you can estimates **ivregress** ***2sls y x1 (x2=z)*** . If you add the option **, first** after the ivregress command, STATA will print the first stage estimates. That is **ivregress** ***2sls y x1 (x2=z), first*** . The first stage estimates are the same as the results from estimating the model **reg** ***x2 z***

Panel data methods
Another approach to handling correlations between covariates and unobserved heterogeneity is to estimate a model with fixed effects. This is possible when you have repeated observations of the same group, say an individual, a firm, a region or a country. In this case, you can use **xtreg** with the **fe** (for fixed effects) option. There are several other options including **re** for random effects.

For example, to estimate a model with fixed effects, $y_{it} = \beta_0 + \beta_1 x_{it} + \mu_i + \varepsilon_{it}$ where μ_i is a fixed effect for all observations in some group denoted by, say, individual, and represented by the subscript i in the model, estimate **xtreg** ***y x , fe i(individual)*** You can declare the group that you want treated as common in these models using the **xtset** command.

Non parametric regression
Locally weighted smoothed scatterplots (LOWESS) is one approach to examining the shape of the relationship between two variables with minimal assumptions. The command **lowess** ***yvar xvar*** **, gen(*yhat*)** will create a predicted value of yvar (called yhat) that you can plot against xvar to examine the shape of the relationship between yvar and xcar. An important choice parameter is the bandwidth over which the local relationship is estimated, **bwidth(*bb*)** where ***bb*** is your choice and is added after the **,** in the **lowess** statement. It is a number between 0 and 1 indicating the fraction of all data included in each local regression. The default is 0.8 which is often too large.

Keeping a copy of results

If you want to keep a copy (on disk or paper) of your session (including the commands you type and the output) then you need to open a log file. Assuming you can write into your current directory, **log using filename**, will open a file called **filename.log**. You can temporarily suspend the log, **log off**, and restart it, **log on**. If you want to close the log, **log close.** If the file exists, you will need to use the **,replace** option**.** To append results to an existing log file use the **,append** option. Notice that your commands are included in the log file. This helps you keep track of what you have done. Notice, also, that the commands generated by your use of the menus is listed in the command viewer and on the log file. This is one way to learn about STATA's commands.

There are many options for how the output is displayed. STATA writes output in a STATA markup language (to make it pretty). If you want the output in plain text, use the **,text** option. Type **help log.** Alternatively click on File in the command menu and then click on Log.

Use the File Print options in the command menu to print a log file or a graph. You can also cut and paste from the Results window or from the Graph window.

Keeping track of your work

Using the command line (or menus) is useful when you first use STATA. However, it quickly becomes very tedious for several reasons. First, it is useful to have a record of what you did so that you can refer to it in the future. Second, you often want to repeat analyses with different subsamples or variables and it is a pain to write out each command over and over again. More generally, it is really important to keep track of your work so that you can always retrace your steps, reproduce your answers and explain what you have done to someone else.

With this in mind, I very strongly encourage you to keep your commands in a program. STATA calls such a program a Do-File. Create mypgm.do with a text editor and from the command window type **do *mypgm***. (You do not have to include the extension .do since it is implied. If you choose to call your do programs something else, include the extension in the filename. If the program is not in the current directory, include the path.) The program will execute as if you entered each command one by one. You can also create a Do-File within STATA by clicking File in the menu options, followed by New, followed by Do-File.

You will notice that if you use the menu to create a command, the command is displayed in the results menu. By logging your output (e.g**. log using *filename*, text**) you will keep a list of everything you have done in text format in a file called filename. If you edit that file, you can easily turn it into a Do-File.

Documenting your programs

A long stream of STATA commands is going to be difficult for you or anyone else to read. Help yourself and anyone who will read your do file by explaining what it does as clearly and concisely as possible. At the beginning of your do file, you should write an explanation of the purpose of the do file and describe the sources of data, the output you create and the date you wrote the program and when it was updated. For each block of commands in the body of the do file, explain what the block of commands will do so that the reader knows what to expect. If you have a really complicated command or some strange looking logic, explain it to the reader before you use it. Getting into the habit of doing this now will pay dividends in the future when you collaborate with others.

Precede any comments in your program with a * which indicates to STATA that the rest of the line should be ignored. If you want STATA to ignore a block of commands use /* to denote the beginning of the block and */ to denote the end of the block. You can also use this to split commands across lines. STATA thinks a command ends when it sees a carriage return. There are three ways to override this. First, you can put /// at the end of the line which signals the next line is a continuation. Second, you can put /* at the end of the first line of the command and then */ at the beginning of the second line. In both cases, you have told STATA to ignore the carriage return. You can achieve the same goal by changing the indicator for the end of a command from carriage return to something else using the #delimit statement. For example #delimit ; tells STATA that a ; denotes the end of each command. This is most useful when you create your own programs.

It is difficult to overstate the importance of documenting your programs. Getting into the habit of doing it when you first start using STATA will be enormously beneficial to you in the future.

Shortcuts: Macros and loops

Macros
Macros are a convenient way to substitute a word, **macro_name**, for a longer statement such as a list of variables or a programming statement, **macro_content**. To create a macro, use the command **local <macro_name> <macro_content>** and to call the macro use **`macro_name'** which will substitute whatever is in **macro_content** at the time the statement is executed. For example:

local varlist *tot_exp hhsize*
reg *food_exp `varlist'*

will substitute totexp and hhlist every time `varlist' is called. In this case, the substitution will expand the reg statement at the time of execution so that it is

reg *food_exp tot_exp hhsize*

The macro is called local because it is only accessible during the execution of the current program. You can also create global macros which can be passed across programs.

Loops
Loops are ideal for when you need to repeat multiple statements in a program. There are several different types of looks. As an example, a **foreach** loop is set up with

foreach *item in <list of variables>* {…code here…}.

where ***item*** is essentially a local macro that substitutes each element of <list of variables> and executes the code in the brackets for each element. For example, to estimate the same regression with 3 different dependent variables, food_exp, hous_exp and clth_exp you can write each reg statement

reg *food_exp tot_exp hhsize*
reg *hous_exp tot_exp hhsize*
reg *clth_exp tot_exp hhsize*

or put the statement inside a foreach loop:

```
foreach yvar in food_exp hous_exp clth_exp {
        reg `yvar' tot_exp hhsize
  }
```

which will result in STATA executing the same regressions as the three reg statements above.

Forvalues loops are useful if you want to execute statements over a series of values that you specify. For example, to estimate a regression that relate food to total expenditure separately for households of size 1, 2, 3 up to 8 members, you can write 8 regression statements. With forvalues, you can write one regression statement and loop through each value of household size from1 to 8:

```
forvalues J = 1/8 {
        reg food_exp tot_exp if hhsize==`J'
  }
```

where the index variable J is defined to run from 1 to 8 and is identified within the loop as a local macro `J'.

Documenting your data

If you create a dataset, you should document the data. First, you can label any variable in a dataset using **label variable "<variable label>".** You can label a dataset using **label data "<date label>".** In both cases, you can write whatever you like in the <…> between the double apostrophes.

Sometimes it is convenient to label the values of a variable. For example, the variable gender may be 1 if a person in a study is male and 3 if the person is female. It's a bit hard to remember which is male and which is female. First define the labelname using **label define labelname value "text".** For example

label define *lblgender 1 "1.Male" 3 "3.Female"*

Next assign the values of labelname to the variable. For example

label values gender lblgender

Now **tabulate gender** and you will see 1.Male and 3.Female instead of 1 and 3, respectively. For more information on labels, see help label.

This is intended to be a brief introduction to some features of STATA that you will use. To be sure it only scratches the surface of the power of the product.

The best way to learn how to use STATA is to, well, use it. With the menus, on-line help and this handout, you should have enough to get going.

Experiment and try out features. When in doubt, consult the excellent on-line help tool, the manual, a friend, an AI tool or do a web search.