Helpful Hints in Using STATA

Data input

Inputting interactively <u>from keyboard</u> - useful for small datasets

1. Enter data from keyboard
    a. Syntax - **input** [varlist] [, automatic label]
    b. Description - input allows you to type data directly into the dataset in memory. See also [D] edit for a windowed alternative to input.
    c. Options -
        i. automatic causes STATA to create value labels from the nonnumeric data it encounters. It also automatically widens the display format to fit the longest label. Specifying automatic implies label, even if you do not explicitly type the label option
        ii. label allows you to type the labels (strings) instead of the numeric values for variables associated with value labels. New value labels are not automatically created unless automatic is specified.
2. Edit and list data with Data Editor
    a. Syntax - Edit using Data Editor: **edit** [varlist] [if] [in] [, nolabel]
    b. List using Data Editor: **browse** [varlist] [if] [in] [, nolabel]
    c. Description: edit brings up a spreadsheet-style data editor for entering new data and editing existing data. edit is a better alternative to input; see [D] input. browse is similar to edit, except that it will not allow to change the data. browse is a convenient alternative to list; see [D] list.
    d. Option - nolabel causes the underlying numeric values, rather than the label values (equivalent strings), to be displayed for variables with value labels; see [D] label.
    e. Remarks - A tutorial discussion of edit and browse is found in the Getting Started with STATA manual. Technical details can be found in [D] edit. Clicking STATA 's Data Editor button is equivalent to typing edit by itself. Clicking STATA 's Data Browser button is equivalent to typing browse by itself. edit, typed by itself, opens the Data Editor with all observations on all variables displayed. If you specify varlist, only the specified variables are displayed in the editor. If you specify one or both of in range and if exp, only the observations specified are displayed. Depending on the version of STATA you use, if the Data Editor is open, you will not be able to use the Command Window.

Inputting from <u>files and spreadsheets</u> - the common way data are brought into STATA - Summary of the different methods

1. insheet (see [D] insheet)
    a. insheet reads text (ASCII) files created by a spreadsheet or database program.
    b. The data must be tab separated or comma separated, but not both simultaneously, and cannot be space separated.
    c. An observation must be on only one line.
    d. The first line of the file can optionally contain the names of the variables.
    e. When importing a spreadsheet from EXCEL, saving the spreadsheet as a ".csv" first will ease the importing process significantly.
2. infile (free format) -- infile without a dictionary (see infile1)
    a. The data can be space separated, tab separated, or comma separated.
    b. Strings with embedded spaces or commas must be enclosed in quote (even if tab- or comma-separated).

        c. An observation can be on more than one line, or there can even be multiple observations per line.
3. infix (fixed format) (see [D] infix (fixed format))
    a. The data must be in fixed-column format.
    b. An observation can be on more than one line.
    c. infix has simpler syntax than infile (fixed format).
4. infile (fixed format) -- infile with a dictionary (see infile2)
    a. The data may be in fixed-column format.
    b. An observation can be on more than one line.
    c. infile (fixed format) has the most capabilities for reading data.

Other data input commands/types:

1. fdause (see [D] fdause). fdause reads SAS XPORT Transport format files -- the file format
2. haver (Windows only) (see [D] haver)
    a. haver reads Haver Analytics (http://www.haver.com/) database files.
    b. haver is only available for Windows and requires a corresponding DLL (DLXAPI32.DLL) available from Haver Analytics.
3. odbc (see [D] odbc)
    a. ODBC, an acronym for Open DataBase Connectivity, is a standard for exchanging data between programs. STATA supports the ODBC standard for importing data via the odbc command and can read from any ODBC data source on your computer.
4. xmluse (see [D] xmlsave)
    a. xmluse reads extensible markup language (XML) files – highly adaptable text-format files derived from ground station markup language (GSML).
    b. xmluse can read either an Excel-format XML or a STATA -format XML file into STATA.

Managing Your Data

File Storage

1. In your EconS490 folder create a protected reserve data base folder where you save the original databases you download from the internet. This is your primary database. You never make any changes to these.
2. In your EconS490 folder create a working STATA database folder where you save the databases you use to create new variables and estimate the model. This is your secondary database. Save copies of the data sets from your primary folder into your secondary folder and then make the changes you need to your data sets.
3. **Back up your work in 2 or more places.** https://drive.google.com is a good place to store projects and files. https://skydrive.live.com is convenient if you have a hotmail account. Flash drives are also a good option. If all else fails, email it to yourself.

Shaping Data Sets

1. Often you get data for a variable that comes in a block (matrix). For example, you might have one variable, say unemployment rate, by state in the rows and year in the columns. To do your regression, you want unemployment rate to be a single column with an entry for each state from year 1 to year T.

2. Example. Table 1a is a matrix of 12 observations in "Wide Format" – showing a variable for four states by three years. The table below (1b) is the same data but the values are now in one column with 12 rows of data. STATA calls the column version "Long Format." Note that another column has been added to show the year.

Table 1a

State	Var1	Var2	Var3
A	12	45	55
B	13	40	44
C	15	39	39
D	7	27	51

3. STATA can *reshape* the whole data at one time. The reshape command can be used to reshape from wide to long or long to wide. To use the reshape command, the variables have to start with the same prefix: the name of the variable, "*var*". The syntax is

   reshape long var, i(state) j(year)

4. Note that *i(state)* is the existing variable from table 1a, and *j(year)* is the name of the dimension represented by columns in table 1a.

Table 1b

State	Year	Var
A	1	12
A	2	45
A	3	55
B	1	13
B	2	40
…	…	…
D	3	51

Combining Data

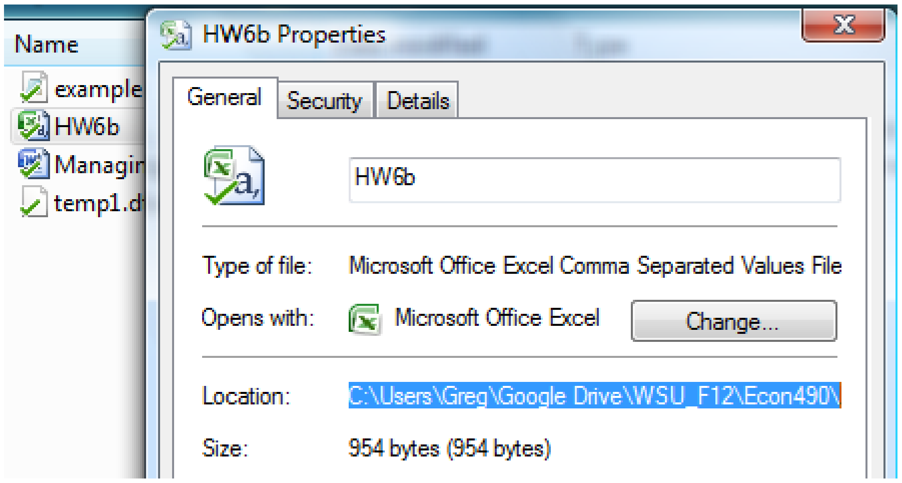
1. STATA command, *append,* combines datasets by stacking. If all the variables have the same name, it will simply stack them on top of each other. You can add an option so that you can identify which original data set the observations come from. Any variables that have different names will show up as a different variable (column) with missing values " . " where they should be in the other data set
2. Here are two syntax examples where data files are named data1 and data2. Although omitted here for simplicity, the file name may need to be included in quotations and/or by the file path.
   Clear
   Append using *data1 data2*

   Use *data1*, **clear**
   Append using *data2*, **generate(***newvar***)**
   where the option, "generate(newvar)" creates a new variable titled "newvar"
3. The *merge* command combines observations in two or more data sets by using some key variables as links between the data sets. For example, if you have a main data set based on state and year with five variables in your current data set, and you want to add data for two more variables for the same states and years you would use the *merge* command. The merge command needs to know a) how many data sets you are using, b) what the data sets are, and c) what the linking variable is.
4. A syntax example:
   Use *data1*, **clear**
   Merge 1:1 *varX*$_1$ **using** *data2*

   In this example data2 was merged with data1, using varX$_1$ as the key (connecting) variable. The 1:1 means that the two data sets were merged into one. As always, refer to the STATA help command *help merge* for more options and examples.

Analyzing Your Data

1. Create a "do file" to keep track of the commands you run on your data in STATA. It allows you to easily change them and re-run them.
2. A do file requires that the file path names are correct. Most of you will be using STATA in the Econ lab in Hulbert so you will only have to specify the file path names once. Every time you work on your project, download your files to the same place. A flash drive is most convenient because you never have to move files even when you use a different computer to run STATA. Just specify the file path name to the correct place on your flash drive.
   a. If you need to find the file path name, open the folder to which you want to download.
   b. Right click on a file in the folder and click on Properties at the bottom of the list.



HW6b

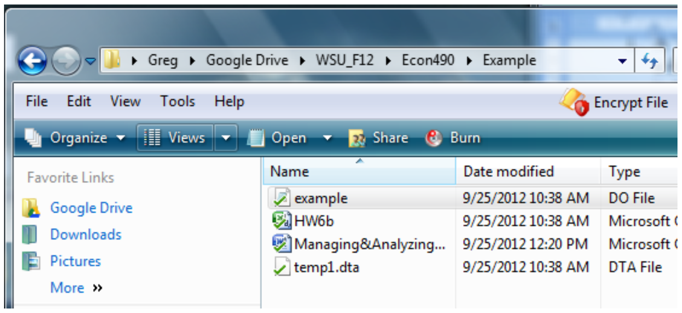
Type of file: Microsoft Office Excel Comma Separated Values File

Opens with: Microsoft Office Excel

Location: C:\Users\Greg\Google Drive\WSU_F12\Econ490\

Size: 954 bytes (954 bytes)

   c. Highlight **the entire line** next to Location. This is the file path name. Press "Ctrl" and "c" at the same time to copy the path name. Paste this into your do file where you see example file path names.
3. For additional guidance, download the example do file, "example1", from "STATA Do-file Example" in the Notes section of the class website to an easily accessible location. Also download the "temp1.dta" file and the "HW6b.csv" file to the same location. Open STATA, and click on the icon that looks like a notepad, then click file and then click open. Alternatively, open STATA, click on File then click Open Do-File.



Name	Date modified	Type
example	9/25/2012 10:38 AM	DO File
HW6b	9/25/2012 10:38 AM	Microsoft (
Managing&Analyzing...	9/25/2012 12:20 PM	Microsoft (
temp1.dta	9/25/2012 10:38 AM	DTA File

4. To run a do file, click the top right icon, Do or Execute (do).
5. Save changes made to do file before closing. Unless you have made changes to the data, you don't need to save the data file.

Mean, Variance, Number of Non-missing Observations, Minimum, Maximum, Etc.

summarize *varlist*	(summary for the variables listed)
summarize *varlist*, detail	(detailed summary for the variables listed)
by *byvars*: **summarize** *varlist*	(For example, "by gender: summarize wage")

Tabulations, Histograms, Density Function Estimates

tabulate *x*	(a table with the number of observations)
tabulate *x1 x2*	(a two-way table with the number of observations)

histogram *x* (plots a histogram of the specified variable)

Scatter Plots

scatter *y x* *(*Plots data, with *y* on the vertical and *x* on the horizontal axis)
scatter *y1 … yk … x* (Plots multiple variables on the vertical and *x* on the horizontal axis)

Regression Results Tables

[Note: you may need to download these commands]

After running each regression, you can save the results.

estimates store *nameofregression1* (make sure each regression has its own unique name)

After running all of your regressions, you can display all of them in one convenient table.

estimates table *nameofregression1 nameofregression2 nameofregression3 etcetera,* star(.01,.05,.1)

There are many available options with this command. Be sure to explore them. In this example, I illustrate the star option. It places stars next to each coefficient depending on the significance level. You can customize this to include number of observations, R-squared, and more.

Another option is to create a .csv (a file that excel can read) of the table directly.

esttab *name of regression1 nameofregression2 nameofregression3 etcetera* using NameFile.csv, replace se plain

The option replace, will write on top of the existing file of that name (useful for running your do-file over again). The option se gives the standard errors. There are other options if you would rather report the t-stats or something else.

Correlations and Covariances

correlate *x1 x2…* (Computes the sample correlations between variables)
correlate *x1 x2* …, covariance (Computes the sample covariances between variables)

Generating Variables

generate *newvar* = … (Generates a new variable using the formula you enter in place of "…")
For example; **gen** f = m * a
tsset time (Creates a time variable without gaps, e.g., 1,…,t, when using time-series data)
tabulate *categorical var.*, **generate(**varname**)** (Creates a series of indicator variables with values of 1 and 0, one for each category in the "*categorical var"*. Do not use this command for continuously measured variables.

OLS Regression (and Weighted LS and Generalized LS)

regress *y xvarlist* (Regresses the dependent variable y on the independent variables xvarlist)

Predicting and plotting residuals and fitted values

1. To generate the residuals from a regression, use:

predict [varname], **resid**

2. To generate fitted values from a regression, use:

   predict [varname], **xb**

Note that these commands must be run directly after the regression has been computed.

General Linear F-Tests

To test for joint significance among two or more variables in your regression, use the command

test var1 var2…

directly after running your regression. This will report the F-value and associated p-value for the linear hypothesis H_0:var1 $=$ var2 $= \cdots = 0$.

Testing for Multicollinearity

Multicollinearity: correlation among regressors

1. Evidence of high multicollinearity:
   a. High R^2 but few significant coefficients
   b. High pair-wise correlations among regressors
   c. High variance inflation factors
2. Testing for high multicollinearity by computing variance inflation factors (VIFs):
   regress *y varlist*
   vif
3. Dealing with high multicollinearity
   a. Obtain additional data or get a new sample.
   b. When possible, combine variables by making an index.
   c. Drop variable(s) with highest VIF (if any greater than 10) unless that violates theory or otherwise impedes accomplishing your research objectives. Be sure to consider the possible consequences of omitted variable bias before deleting any variables.
   d. Do nothing if well tested theory must be violated or important research objectives cannot be accomplished with any of the above actions.

Testing for Heteroskedasticity

Heteroskedasticity: Errors do not have the same variance.

1. Testing for heteroskedasticity:
   hettest
   and/or
   ssc install whitetst
   regress *y xvarlist*
   whitetst
2. Detecting evidence of significant heteroskedasticity:
   If $p < 0.05$, the null hypothesis of no heteroskedasticity is rejected.
3. Dealing with heteroskedasticity:
   regress *y xvarlist*, robust (this provides an estimator for standard errors that, in large samples, is robust to the presence of heteroskedasticity. For small samples, bootstrapped standard errors are preferable.

Testing for Autocorrelation with Time Series Data

Autocorrelation: Errors are correlated with the error of a previous observation

1. Testing for autocorrelation:
   regress *y xvarlist*

tsset [time variable]
The "time variable" will be specific to your data. It is the variable name that includes the time information. This could be days, months, quarters, years, etc.
estat dwatson
Note: if any regressors are endogenous (e.g. a lagged dependent variable), use the command **estat durbinalt**)
<u>and/or</u>
tsset [time variable]
bgodfrey, lags(1 2 3 4)

2. Detecting evidence of significant autocorrelation:
   If $p < 0.05$, the null hypothesis of no autocorrelation is rejected.
3. Dealing with autocorrelation:
   newey *y xvarlist*, lag(1)
   This is the **Newey-West** method of correcting OLS standard errors for autocorrelation; parameter estimates will be identical to the OLS case.
   <u>or</u>
   prais y x, corc
   Note: parameter estimates, t-values, R^2 value, and Durbin-Watson statistic will all differ after the re-estimation.

Regression with Panel Data

1. Panel data can be estimated as a fixed effects or random effects model:
   a. The **fixed effect model** assumes that individual heterogeneity is captured by the intercept term. This means every individual gets its own intercept alpha(i) while the slope coefficients are the same. This also means that the heterogeneity is associated with the regressors on the right hand side.
      xtset *cross-section-identifier time-identifier*
      xtreg *y xvarlist*, fe
      where *cross-section-identifier* is the variable which denotes each cross-sectional unit, and *time-identifier* is the variable that identifies the time dimension such as year.
   b. The **random effects model** assumes in some sense that the individual effects are captured by the intercept and a random component mu_i. This random component is not associated with the regressors on the right hand side but is part of the error term. The intercept becomes $alpha+mu_i$.
      xtset *cross-section-identifier time-identifier*
      xtreg *y xvarlist*, re
2. Testing for random or fixed effects – Hausman test:
   xtset *cross-section-identifier time-identifier*
   xtreg *y xvarlist*, fe
   estimates store fixed (Note: this command stores the *fixed* estimates)
   xtreg *y xvarlist*, re
   estimates store random (Note: this command stores the *random* estimates)
   hausman *fixed random*
   After conducting the Hausman test, use the fixed effect model if $p < 0.05$, random effects model otherwise.
3. Testing for and dealing with multicollinearity with panel data – see instructions under OLS Regression
4. Testing for autocorrelation with panel data:
   findit xtserial (Note: select xtserial from the 3 options and then install)

ssc install xtserial
xtreg *y xvarlist*, re (Note: change re to fe if using a fixed effects model)
xtserial *y xvarlist*

If p < 0.05, the null hypothesis of no autocorrelation is rejected.

5. Testing for heteroskedasticity with panel data in fixed effects model:

   xttest3

   If p < 0.05, the null hypothesis of no heteroskedasticity is rejected.
   [Note: you may need to download this command]

6. Dealing with autocorrelation within panels and cross sectional correlation and/or heteroskedasticity across panels:
   a. With random effects:

      xtreg *y xvarlist*, re vce(robust) (Note: vce(robust) is the Newey-West method of correcting standard errors for autocorrelation and heteroskedasticity in panel data; parameter estimates will be identical to the panel data case with random effects)

   b. With fixed effects:

      xtreg *y xvarlist*, fe vce(robust) (Note: parameter estimates will be identical to the panel data case with random effects)

   <u>or</u>

   xtgls fits cross-sectional time-series linear models using generalized least squares. This command allows estimation in the presence of AR(1) autocorrelation within panels and cross sectional correlation and/or heteroskedasticity across panels.

   xtgls *y xvarlist*, panels(iid) corr(ar1)
   This command accounts for autocorrelated errors.
   <u>or</u>
   xtgls *y xvarlist*, panels(hetero) corr(ar1)
   This command accounts for heteroskedastic and autocorrelated errors.

Regression with a Limited Dependent Variable

1. Binary choice model – dependent variable takes values only of 1 and 0 (e.g., yes or no, approved or denied, success or failure)
   Use **probit** or **logit** regression. Probit regression assumes a standard normal distribution, and logit regression assumes the logistic cumulative distribution function. Empirical results for the binary choice model are not generally very sensitive to the choice of distribution function, so you can typically choose either regression arbitrarily.

   probit *y xvarlist*
   <u>or</u>
   logit *y xvarlist*

2. Ordered limited dependent variable model – dependent variable can take on more than 2 integers, e.g., 1, 2, 3, 4, which are clearly logically ordered in some fashion (e.g., "approved, indifferent, disapproved" or "small, medium, large"). Again, choice between ordered probit and ordered logit is generally arbitrary.

   oprobit *y xvarlist*
   <u>or</u>
   ologit *y xvarlist*

3. Truncated dependent variable model – the effect of truncation occurs when the observed data in the sample are only drawn as a truncated subset of a larger population (e.g., a study of the determinants

of incomes of the poor in which only households with income below a certain poverty line are part of the sample).

> **truncreg** *y xvarlist*, ll(#) (where ll(#) defines the lower truncation point *a*).

If you have both lower and upper truncation points, use:

> **truncreg** *y xvarlist*, ll(varname) ul(varname) (where the lower ll and upper lu thresholds can be observation specific and their values are defined by varname).

4. Censored dependent variable model – censoring occurs when the values of the dependent variable are restricted to a range of values (e.g., income data that are *top-coded* in the survey such as incomes above $200,000).

   > **tobit** *y xvarlist*, ll(0)

   You can also estimate more general models with censoring from below ll(#) and above ul(#)

   > **tobit** *y xvarlist*, ll(#) ul(#)