

Using Stata for Applied Research: Reviewing its Capabilities

Christopher F. Baum, *Boston College, DIW Berlin*

Mark E. Schaffer, *Heriot-Watt University, CEPR, IZA*

Steven Stillman, *Motu Economic and Public Policy Research, University of Waikato, IZA*

1. Introduction

What is Stata, and why should it be the package of choice for applied econometric research?¹

Stata is different things to different users. For many users, Stata is a statistical package, similar to other commercial packages that allow the user to start the program and select menu items to read data, generate new variables, compute statistical analyses and draw graphs. To other users, Stata is a command-line driven package, commonly executed from a do-file of stored commands which will perform all of the steps above without intervention. For some, Stata is considered a programming language in which they develop ado-files that define programs, or Stata commands that extend the Stata language by adding new data transformation facilities, statistical techniques, or graphics commands.

Stata is available in several versions: Stata/IC (the standard version), Stata/SE (an extended version) and Stata/MP (for multiprocessing). The major difference between the versions is the number of variables allowed in memory, which is limited to 2,047 in standard Stata/IC, but can be much larger in Stata/SE or Stata/MP. Stata/MP is a multiprocessor version, capable of utilizing 2, 4, 8...64 processors available on a single computer. Stata/IC will meet most users' needs; if you have access to Stata/SE or Stata/MP, you can use that program to create a subset of a large survey dataset with fewer than 2,047 variables. Stata runs on all 64-bit

¹ Stata's website, <http://www.stata.com>, provides detailed information about the software while <http://www.stata.com/whystata/> covers some of the same ground as this article. Appendix 1 provides an extensive, albeit incomplete, list of the estimation methods available in Stata. See <http://www.stata.com/capabilities/> for the complete list of what everything that Stata is capable of doing.

operating systems which can access a larger memory space. This can be important because Stata works differently than some other packages in requiring that the entire dataset to be analyzed must reside in memory. This brings a considerable speed advantage, but means that the size of a dataset that can be examined is limited by the RAM (memory) on your computer and the extent to which this can be accessed by Stata. Otherwise, datasets can contain an unlimited number of observations in all three versions.

Stata's pricing is unusual on two grounds: all versions of Stata have the same feature set, with no add-on toolboxes, applications or packages, and there are no annual fees. Standard licences are perpetual, allowing the user to continue to use the current version of Stata on their computer (and on a second system, e.g. at home, barring concurrent usage). Updates within the life of a version are free and often contain major enhancements. For instance, the current Version 11.1 includes many features added since the Version 11.0 release (the current version, released June 2009). A new version usually appears about every two years, at which time further development of the prior version ceases. However, support for the earlier version continues to be available by phone and email.

For a single-user perpetual academic licence, Stata/IC costs USD 595, while Stata/SE costs USD 895 and the two-processor Stata/MP version costs USD 1,345. The corresponding corporate/government prices are USD 1,245, USD 1,745 and USD 2,495, respectively. There are also options for Stata/MP with 4, 8, 16, ... 64 cores, as well as network licences, volume purchase agreements, educational lab licences and leasing arrangements. Beginning with Stata 11, each copy includes a complete set of manuals (over 6,000 pages) in PDF format, hyperlinked to the on-line help. If a paper copy of the 18 manuals is desired in addition, this can be purchased for USD 250.

Academic institutions making use of Stata in instruction and research should set up a Stata GradPlan, in which a local coordinator holds inventory and fulfils orders. This provides for significant reductions in cost and next-day fulfilment of campus orders. A perpetual Stata/IC licence, for students or faculty, is in this case USD 179, or USD 425 for Stata/SE and USD 875 for Stata/MP2. Students may purchase a one-year licence for Stata/IC for USD 98, or a six-month licence for USD 65. A fourth version of Stata, 'Small Stata', is aimed at student users and is available for USD 49/USD 29 for a one year and six-month license, respectively. This version of Stata is adequate for a beginning statistics course, but not for more advanced coursework.

Stata has traditionally been a command-line-driven package that operates in a graphical (windowed) environment. There are four windows in the default interface: the Review, Results, Command and Variables window. You may alter the appearance of any window using the Preferences->General drop-down menu, and make those changes on a temporary or permanent basis. For instance, if you do not care for the current default colour scheme of black text on a white background, you can alter your window preferences to change back to the old beloved green on black scheme.

As you might expect, you may type commands in the Command window. When a command is executed---with or without error---it appears in the Review window, and the results of the command (or an error message) appears in the Results window. You may click on any command in the Review window and it will reappear in the Command window, where it may be edited and resubmitted. Figure 1 presents a screenshot of Stata after opening the sample 'Auto' dataset that is included with Stata and running the `describe` command.

Since Stata 8, commands can also be entered using drop-down menus and dialog boxes. A major advantage of Stata's menu system is that every command issued in this manner is

echoed to the Review window allowing the user to then reissue these commands or variants of them using the traditional Stata command-line-driven approach. Thus, you may first enter a command using the drop-down menus, then examine its syntax, revise it in the Command window and resubmit it. This can be a very efficient way of learning the syntax for commands new to the user.

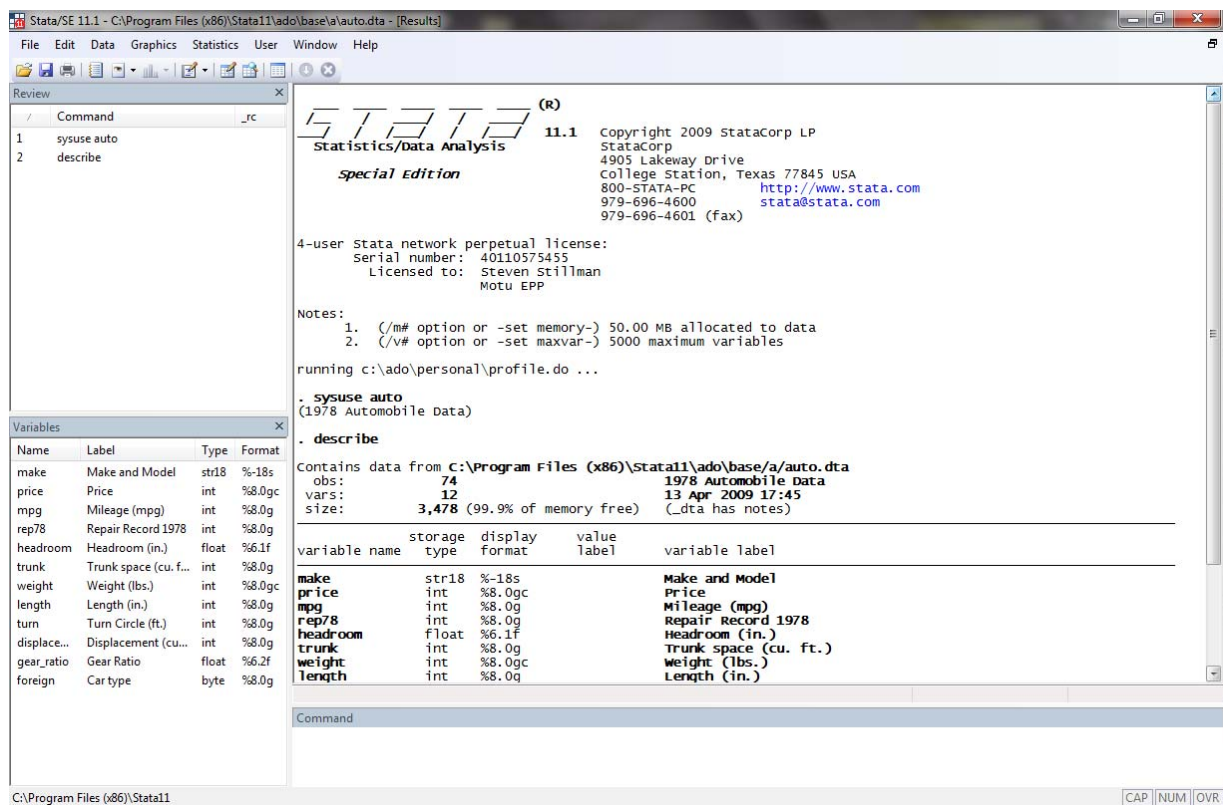


Figure 1: Stata Windowing using Default Preferences

2. The Stata Command Environment (or User Interface)

Before we discuss the specifics of Stata usage, let's consider a meta-issue: why would you want to learn how to use a command-line-driven package? Stata may be used in an interactive mode, and those learning the package may wish to make use of the menu and dialog system. But when you execute a command from a pull-down menu, it records the command that you could have

typed in the Review window, and thus you may learn that with experience you could type that command (or modify it and resubmit it) more quickly than by use of the menus.

This approach allows for all previous work to be readily reproduced through the use of batch files, known as 'do-files' and 'log-files'. It also allows for the continual expansion of Stata's capabilities. The vast majority of Stata commands are written in Stata's own programming language--the 'ado-file' language. Hence, Stata users can also write commands that will work just like official commands.

Let us consider the form of Stata commands. One of Stata's great strengths, compared with many statistical packages, is that its command syntax typically follows strict rules: in grammatical terms, there are few irregular verbs. This implies that when you have learned the way a few key commands work, you will be able to use many more without extensive study of the manual or even the on-line help. The fundamental syntax of all Stata commands follows a *template*. Not all elements of the template are used by all commands, and some elements are only valid for certain commands. But where an element appears, it will appear in the same place, following the same grammar. Like Unix or Linux, Stata is case sensitive. Commands must be given in lower case. For best results, keep all variable names in lower case to avoid confusion.

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. In some cases, only the `cmdname` itself is required. For example, `describe` without arguments gives a description of the current contents of memory (including the identifier and timestamp of the current dataset), while `summarize` without arguments provides summary statistics for all numeric variables. Both may be given with a `varlist` specifying the variables to be considered.

What are the other elements?

The `varlist` is a list of one or more variables on which the command is to operate: the subject(s) of the verb. Stata works on the concept of a single set of variables currently defined and contained in memory, each of which has a name. As `describe` will show you, each variable has a data type (various sorts of integers and real numbers, and string variables of a specified maximum length). The `varlist` specifies which of the defined variables are to be used in the command.

The order of variables in the dataset matters, since you can use hyphenated lists to include all variables between first and last (the `order` and `move` commands can alter the order of variables). You can also use 'wildcards' to refer to all variables with a certain prefix. If you have variables `pop60`, `pop70`, `pop80`, `pop90`, you can refer to them in a `varlist` as `pop*` or `pop??`.

The `exp` clause is used in commands such as `generate`, `replace` and `egen` (extended generate) where an algebraic expression is used to produce a new (or updated) variable. In algebraic expressions, the operators `==`, `&`, `|` and `!` are used as equal, AND, OR and NOT, respectively. The `^` operator is used to denote exponentiation. The `+` operator is overloaded to denote concatenation of character strings.

Stata differs from several common programs in that commands automatically apply to all observations currently defined. You need not write explicit loops over the observations. You can, but it is usually bad programming practice to do so. Of course you may not want to refer to all observations, but to pick out those that satisfy some criterion. This is the purpose of the `if exp` and `in range` clauses. The `in range` clause restricts the analysis to a particular subset of observations based on their position in the dataset while the `if exp` clause restricts the set of observations to those for which the `exp`, a Boolean expression, evaluates to true.

Some commands access files: reading data from external files or writing to files. These commands contain a `using` clause, in which the filename appears. If a file is being written, you must specify the `replace` option to overwrite an existing file of that name. Stata's own binary file format, the `.dta` file, is cross-platform compatible, even between machines with different byte orderings (low-endian and high-endian). A `.dta` file may be moved from one computer to another using `ftp` or `scp` in binary transfer mode.

Many commands make use of options, such as `clear on use`, or `replace on save`. All options are given following a single comma, and may be given in any order. Options, like commands, may generally be abbreviated (with the notable exception of `replace`).

A number of Stata commands can be used as *prefix commands*, preceding a Stata command and modifying its behaviour. The most commonly employed is the `by:` prefix, which repeats a command over a set of categories. The `statsby:` prefix repeats the command, but collects statistics from each category. The `rolling:` prefix runs the command on moving subsets of the data (usually time series). There are several other command prefixes: `simulate:`, which simulates a statistical model; `bootstrap:`, allowing the computation of bootstrap statistics from resampled data; and `jackknife:`, which runs a command over jackknife subsets of the data. The `svy:` prefix can be used with many statistical commands to allow for survey sample design.

3. Advantages of Stata

No matter what your level of sophistication as a Stata user might be, there are some distinctive features of Stata that should be noted and well understood if you are to use Stata effectively and efficiently. In this context, efficiency refers primarily to the use of your time: an efficient Stata user does not type (nor copy and paste) repetitive commands, nor do they continuously reinvent the wheel. Pure computational efficiency is a component of that measure. If a Stata do-file can be

rewritten to run in ten seconds rather than two minutes, it should be. But the primary concern should be the ease by which you can revise the analysis being performed, easily rerun it, and do so months later. If you are an efficient user of Stata, you will be able to save many hours of your time by investing in a few techniques which will lead to the generation of comprehensible and extensible do-files that may be rerun with a single command.

3.1. Stata is Easy to Learn

You can easily learn Stata commands, even if you do not know the syntax. Stata contains dialogs giving access to almost every official command and when you execute a command using a dialog, the command syntax defined by your choices appears in the Review window, just as if you had typed it. Although the dialog system allows you to submit a command without closing the dialog, you may often find that you want to execute several commands repetitively (for instance, generating a new variable, then summarizing its values). Even if you have only used Stata with the dialog approach, you can readily recall prior commands, modify them and resubmit them using the Review and Command windows. The contents of the Review window may be saved to a file or copied into the do-file editor window for modification and re-execution. These options become available with a control-click or right-click on the Review window.

3.2. Ease of Replicability

Stata makes maintaining a commented/documented research trail particularly easy to do. Instead of submitting command one at a time in the Command window, most experienced Stata users write do-files, which are text files that list a series of Stata commands separated by either carriage returns or other user selected delimiters. To execute the full set of commands listed in a particular do-file the user only needs to type `do dofile.do`. For example, the following do-file reads some daily Dow-Jones Averages data, graphs daily returns, then performs Dickey-Fuller

tests for unit roots on the DJIA, its log, and its returns (log price relatives), and on their first differences. AR(3) models are then estimated on the series, and the Box–Pierce portmanteau test is then performed on the residuals.²

```
log using intro2,replace
use http://fmwww.bc.edu/ec-p/data/micro/ddjia.dta
describe
summarize
tsset
tsline ret
foreach v of varlist djia ldjia ret {
    dfgls `v', maxlag(12)
    dfgls D.`v', maxlag(12)
    regress `v' L(1/3).`v', robust
    predict eps_`v', resid
    wntestq eps_`v'
}
log close
```

While a do-file can be edited in any text editor, Stata has a built-in do-file editor which enables the user to submit only code fragments as opposed to the entire do-file (this is also possible with other editors, but requires some programming experience). Using your mouse, you can select any subset of the commands appearing in the do-file editor and execute only those commands. That makes it very easy to test whether those commands will perform the desired analysis.

It is also standard practice among experienced Stata users to open a log-file at the beginning of every session. This is done by typing `log using logfile` in the command window. A log-file is either a text-file or a file written in Stata's own mark-up language (and readable inside Stata) that echoes everything that is displayed in the Results window while it is opened. In other words, it records both the full set of commands that are issued either

² This example makes use of “local macros” (with values ``v'`), which enable Stata to perform the same operations on several named variables without having to write out the commands for each variable. This facility may be used with variable lists of any length and makes it very easy to generate parallel analyses, produce graphs, etc. for an arbitrary set of variables or time periods. This is discussed further in section 3.7 below.

interactively or from a do-file and the resultant output from these commands including error messages. Hence, the combined use of do-files and log-files ensures reproducibility and creates a documented record of a user's research strategy (especially if you make use of Stata's comments to describe what is being done, by whom, on what date, etc).

3.3. Web Enabled and Fully Supported

Stata is fully "web-aware", hence commands like `use filename` are by no means limited to the files on your hard disk. For example, `webuse klein` or `use http://fmwww.bc.edu/ecp/data/Wooldridge/crime1.dta` will read these datasets into Stata's memory over the web.

One of Stata's great strengths is that it can be updated over the Internet. Stata regularly (or when desired by the user) contacts its web server and enquires whether there are more recent versions of either Stata's executable (the kernel) or the ado-files. The kernel is updated relatively infrequently---once a month at most---but the ado-files may be modified every ten days or so. This enables Stata's developers to distribute bug fixes, enhancements to existing commands, and even entirely new commands during the lifetime of a given release. Updates during the life of the version you own are free. You need only have a licensed copy of Stata and access to the Internet (which may be by proxy server) to check for and, if desired, download the updates.

3.4. Cross-platform Portability

Stata is eminently portable and its developers are committed to cross-platform compatibility. Stata runs the same way on Windows, Mac OS X, Unix and Linux systems. The only platform-specific aspects of using Stata are those related to native operating system commands: e.g., the fully qualified name of the file to be accessed. A Stata license may be used on any machine which supports Stata, as there are no machine-specific licenses for Stata 11. You may install Stata on a home and office machine, as long as they are not used concurrently. Stata's binary data

files may be freely copied from one platform to any other, or even accessed over the Internet from any machine that runs Stata. You may store Stata's binary datafiles on a webserver (HTTP server) and open them on any machine with access to that server.

3.5. *User Extensibility / Extensive User Community*

The importance of Stata being web aware goes far beyond the limits of official Stata. Since the `adopath` includes both Stata directories and other directories on your hard disk (or on a server's filesystem), you may acquire new Stata commands from a number of web sites. The *Stata Journal* (SJ), a quarterly refereed journal, is the primary method for distributing user contributions. Between 1991 and 2001, the *Stata Technical Bulletin* (STB) played this role and a complete set of its issues are available on line at the Stata website.

The SJ is a subscription publication (articles more than three years old are freely downloadable), but the `ado-` and `sthlp-`files from all issues may be freely downloaded from Stata's web site. The Stata `help` command accesses help on all installed commands; the Stata command `findit` will locate commands that have been documented in the *STB* and the *SJ*, and with one click you may install them in your version of Stata. Help for these commands will then be available on your desktop.

But this is only the beginning. Stata users worldwide participate in the *Statalist* listserv, and when a user has written and documented a new general-purpose command to extend Stata functionality, they announce it on the listserv. Since September 1997, Statalist users have placed the programs they have written and wish to share in the Boston College Statistical Software Components (SSC) Archive in RePEc (Research Papers in Economics), available from IDEAS (<http://ideas.repec.org>) and EconPapers (<http://econpapers.repec.org>). At the time of writing this article, there were well over 1,000 Stata packages available from SSC, with more being added all

the time. Numerous books have been written about using Stata and are available at <http://www.stata.com/bookstore/statapressbooks.html>. For example, Baum (2009) provides a comprehensive discussion of programming in Stata for both those just becoming familiar with the package and expert users.

Any component in the SSC Archive may be readily inspected with a web browser, using IDEAS' or EconPapers' search functions and if desired you may install it with one command from the archive from within Stata. For instance, if you know there is a module in the archive named `mvsumm`, you could use `ssc describe mvsumm` to learn more about it and `ssc install mvsumm` to install it if you wish. Anything in the archive can be accessed via Stata's `ssc` command: thus `ssc describe mvsumm` will locate this module and make it possible to install it with one click.

The importance of all this is that Stata is infinitely extensible. Any ado-file on your `adopath` is a full-fledged Stata command. Stata's capabilities thus extend far beyond the official supported features described in the manuals to a vast array of additional tools. Traditionally, most official Stata commands were written as ado-files, making it extremely easy to modify them to add new features. However, one current issue is that official Stata is increasingly being written in compiled Mata code (see below for more information on Mata). While this has increased the speed and capability of many programs, StataCorp is not currently distributing the uncompiled source code. Hence, it is not currently possible for the user community to use many official Stata commands' code as templates for their programming efforts.

3.6. Working with Groups of Observations

You could loop over observations, but... Stata differs from several other statistical packages in its approach to variables. When you have read in a Stata dataset, Stata's memory contains a table of

rows and columns: the rows corresponding to the observations and the columns being the variables. This view of the contents of memory can be readily accessed from the Data Viewer or Data Editor icons on Stata's Toolbar. It is important to understand that Stata's commands operate on variables without explicit mention of the observations. If we give the command `generate priceK = price/1000`, we need not specify for which of the observations that new variable is to be generated. Stata will create every available observation of `priceK`, limited only by missing data or an `if` or `in` clause. Unlike other statistical package languages, there are very few circumstances in Stata where you must refer to the specific observation, and Stata will run much faster if you avoid doing so.

The `by:` prefix makes working with groups of observations a breeze. When a command is prefixed with a `bylist`, it is performed repeatedly for each element of the variable or variables in that list, each of which must be categorical. The ability to define by-groups from one or more categorical (integer-valued) variables implies that the Stata commands needed to carry out some quite sophisticated data transformations are short and quite simple. This is a very handy tool, which often replaces explicit loops that must be used in other programs to achieve the same end.

Another useful aspect of `by:` is the way in which it modifies the meanings of the observation number symbol. Usually `_n` refers to the current observation number, which varies from 1 to `_N`, the maximum defined observation. Under a `bylist`, `_n` refers to the observation within the `bylist`, and `_N` to the total number of observations for that category. This concept is very powerful. Consider a set of household survey data in which each observation is an individual's set of responses on variables like age and income, with households identified by a household ID variable. Then, we may generate an indicator (or dummy) variable identifying the set of single-member households with a single command: `bysort hhid: generate byte`

`single = (_N==1)`. If we need to analyze only these households, we may use the qualifier `if single` to select them.

Stata also fully supports time-series data. The `D.`, `L.`, and `F.` operators may be used under a time-series calendar (including in the context of panel data) to specify first differences, lags and leads, respectively. These operators understand missing data and numlists: e.g. `L(1/4).x` is the first through fourth lags of x , while `L2D.x` is the second lag of the first difference of x . It is important to use the time series operators to refer to lagged or led values, rather than referring to the observation number (e.g., `_n-1`). The time series operators respect the time series calendar and will not mistakenly compute a lag or difference from a prior period if it is missing. This is particularly important when working with panel data to ensure that references to one individual do not reach back into the prior individual's data.

3.7. Easy Ways to Avoid Repeating Code

If you can loop over variables, by all means do so! One of Stata's most valuable features, in terms of saving your time and effort, is the ability to repeat steps (data transformations, estimation, or the production of graphics) over a number of variables. The commands `forvalues`, `foreach` and `local` can be used to produce a do-file that will loop over variables rather than containing a separate command for each one and may be easily modified for a different list of variables when a variation on the analysis is needed. The sample do-file in section 3.2 illustrates one example of doing this.

3.8. Post-Estimation Commands and Results Left Behind

All estimation commands share the same syntax. Estimation commands display confidence intervals for the coefficients and tests of the most common hypotheses. More complex hypotheses may be analyzed with the `test` and `lincom` commands; for nonlinear hypotheses,

`testnl` and `nlcom` may be applied, making use of the delta method. Predicted values and residuals may be obtained after any estimation command with the `predict` command. For nonlinear estimators, `predict` will produce other statistics as well: e.g., the log of the odds ratio from logistic regression. The `margins` command may be used to generate predictions of conditional means and marginal effects, including elasticities and semi-elasticities, for any estimation command.

All estimation commands 'leave behind' results of estimation in the `e()` array, where they may be inspected with `ereturn list`. Any item here, including scalars such as R^2 and RMSE as well as the coefficient vector and its estimated variance-covariance matrix may be saved for use in later calculations. Similarly, data manipulation and statistical commands leave behind results in the `r()` array, where they may be inspected with `return list`. For example, typing `return list` after running the `summarize` command on one variable reveals that the mean of this variable is stored in the scalar `r(mean)`, which can then be referred to say to generate an indicator variable for whether an observation is above or below the mean by typing `generate abovemean = x > r(mean)`.

3.9. Easy to Produce Publication Quality Graphs and Tables

The `graph` command can be used to produce a wide variety of publication quality graphs, for example, scatterplots, line plots, scatterplot matrices, bar charts, dot charts, box-and-whisker plots, pie charts, as well as a wide range of distributional diagnostic plots, smoothing and densities, regression diagnostics plots, time-series and panel data graphs, survival analysis graphs, ROC analysis and quality control charts (see <http://www.stata.com/capabilities/graphics.html> for more information). Figure 2 presents an example of a composite graph

produced in Stata exploring some properties of `auto.dta`. Mitchell (2008) is an excellent resource for learning how to make many different types of graphs in Stata.

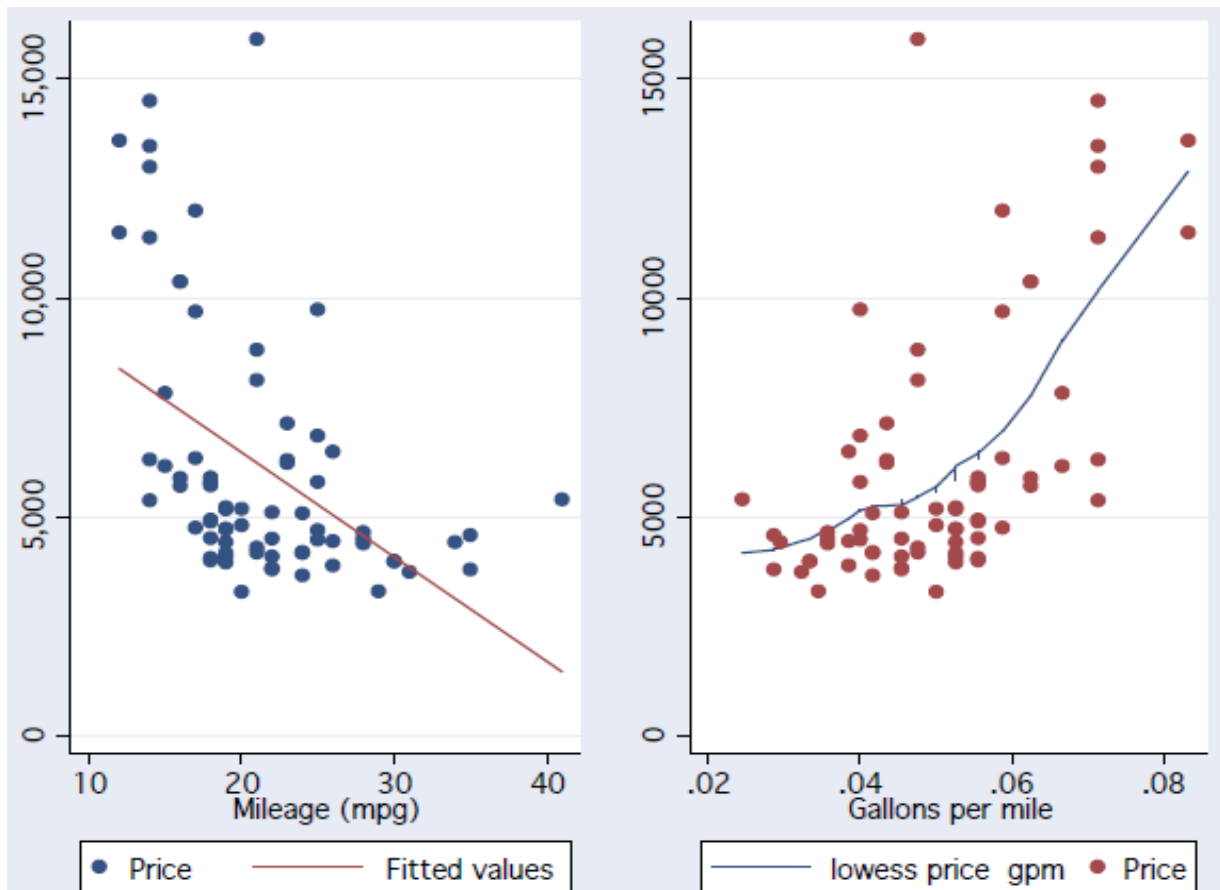


Figure 2: An Example of Graphical Output

The `estimates` suite of commands allow you to store the results of a particular estimation for later use in a Stata session and will produce a nicely-formatted table of results. Options on `estimates table` allow you to control precision, whether standard errors or t-values are given, significance stars, summary statistics, etc. Although `estimates table` can produce a summary table quite useful for evaluating a number of specifications, we often want to produce a publication-quality table for inclusion in a word processing document. Ben Jann's `estout`

command suite processes stored estimates and provides a great deal of flexibility in generating such a table.

Programs in the `estout` suite can produce tab-delimited tables for MS Word, HTML tables for the web, and LaTeX tables for professional papers. In the LaTeX output format, `estout` can generate Greek letters, sub- and superscripts, and the like. This program is available from SSC, with extensive on-line help, and was described in Jann (2005; 2007) and at <http://repec.org/bocode/e/estout>. Figure 3 provides an example of a table created using this program.

Table 1: Models of auto price

	(1)	(2)	(3)
	price	price	price
mpg	-186.7* (-2.13)		
length	52.58 (1.67)	-97.63* (-2.47)	-88.03* (-2.65)
turn	-199.0 (-1.44)		
weight		4.613** (3.30)	5.479*** (5.24)
displacement		0.727 (0.10)	
gear_ratio			-669.1 (-0.72)
foreign			3837.9*** (5.19)
_cons	8148.0 (1.35)	10440.6* (2.39)	7041.5 (1.46)
R^2	0.251	0.348	0.552
bic	1387.2	1377.0	1353.5
N	74	74	74

t statistics in parentheses
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Figure 3: An Example of Regression Outcome Formatted using the Estout Command

4. Recent Additions to Stata That People Might Not Know About

4.1. Mata

As of version 9, Stata contains a full-fledged matrix programming language, *Mata*, with all of the capabilities of MATLAB, Ox or GAUSS. Mata can be used interactively or Mata functions can be developed to be called from Stata. A large library of mathematical and matrix functions is provided in Mata including equation solvers, decompositions, eigensystem routines and probability density functions. Mata functions can access Stata's variables and can work with virtual matrices ('views') of a subset of the data in memory. Mata also supports file input/output.

Mata code is automatically compiled into bytecode, like Java, and can be stored in object form or included in-line in a Stata do-file or ado-file. Mata code runs many times faster than the interpreted ado-file language providing significant speed enhancements to many computationally burdensome tasks.

4.2. Graph Editor

One very useful addition to Stata 10 was the new Graph Editor. This lets the user change the look of any graph previously created in Stata. Additions can be made to titles, legends, axes, lines, arrows, markers, annotations, etc.. Previously created components can be modified or removed. Changes are made either by using the graphical interface (for example, dragging a title to a new location) or via drop-down menus. Even better, any changes made with the Graph Editor can be 'recorded' and then applied to another similar graph. So, if you have a series of similar graphs and you want to change the look of the x-axis in all of them, you just need to edit one in the Graph Editor, record the changes you make, and then apply this to the remaining graphs.

4.3 Factor Variables and the Margins Command

A new feature in Stata 11 is the *factor variable*. Stata has only one kind of numeric variable. However, if a variable is categorical, taking on non-negative integer values, it may be used as a factor variable with the `i.` prefix. The use of factor variables not only avoids explicit generation of indicator (dummy) variables for each level of the categorical variable, but it means that the needed indicator variables are generated ‘on the fly’, as needed. There is a great advantage in using factor variables rather than creating new interaction variables. If you define interactions with the factor variable syntax, Stata can then interpret the expression in post-estimation commands such as `argins`. For instance, you can say `i.race#i.sex`, `i.sex#c.bmi`, or `c.bmi#c.bmi`, where `c.` denotes a continuous variable and `#` specifies an interaction.

With interactions between indicator and continuous variables specified in this syntax, we can evaluate the total effect of a change without further programming. For instance,

```
regress healthscore i.sex#c.bmi c.bmi#c.bmi
argins, dydx(bmi) at(sex = (0 1))
```

which will perform the calculation of $\partial \text{healthscore} / \partial \text{bmi}$ for each level of categorical variable `sex`, taking into account the squared term in `bmi`.

Another new feature in Stata 11, the `argins` command, may be used to generate predictions of conditional means and marginal effects, including elasticities and semi-elasticities, for any estimation command. One of its major advantages is that it computes, by default, average marginal effects (AMEs) rather than marginal effects at the point of means (or other specific points). This command supersedes the old `mfx` command which could only produce marginal effects via numerical evaluation and consequently was quite slow when used on larger datasets. As discussed above, `argins`' ability to evaluate total effects when factor variables are used makes an extremely useful contribution to the Stata toolkit of post-estimation commands.

4.5 Flexible GMM Estimation

Generalized method of moments (GMM) is a flexible estimation framework that unlike maximum likelihood (ML) does not require the user to make strong distributional assumptions about the model. Stata 11 introduced the `gmm` command, which allows the user to fit GMM models using a simple syntax to specify moment conditions, instruments, and the type of weighting matrix to be used. For more complicated models, a program can be written that computes each moment condition. Weighting matrices can be chosen which are suitable for use with cross-sectional, time-series, panel and clustered data. This command is a major addition to the toolkit of econometric techniques which can be run in Stata.

5. In Summary

Stata is an extremely powerful, yet flexible and easy to use statistical package that supports both command-line users and those who rely on drop-down menus and dialog boxes to enter commands. Stata also contained a standard programming language and a full-fledged matrix programming language, both which are extensively used by StataCorp when writing official Stata commands. More than anything, Stata has a number of features that are designed to maximise the experiences of its users. It makes maintaining a commented/documented research trail particularly easy to do. It is web enabled and fully supported with frequent updates and bug fixing. It is fully portable across computing platforms. Best of all, Stata is infinitely extensible and has a large user community discussing how to solve problem and writing new commands on a daily and even hourly basis.

References

Baum, C. F. 2009. *An introduction to Stata programming*. College Station, TX: Stata Press.

Jann, B. 2005. "Making regression tables from stored estimates." *Stata Journal* 5 (3): p 288-308.

Jann, B. 2007. "Making regression tables simplified." *Stata Journal* 7 (2): p 227-244.

Mitchell, M. 2008. *A visual guide to Stata graphics*. 2nd ed. College Station, TX: Stata Press.

Appendix 1: Estimation Methods Available in Stata

(1) General estimation methods: OLS, generalized least squares, instrumental variables, seemingly unrelated regression, nonlinear least squares, nonlinear systems of equations, maximum likelihood estimation (both analytical and numerical) and generalised method of moments (GMM).

(2) Limited dependent variable estimation commands: logit model, logistic regression, binomial probit model, one- and two-limit tobit model, censored normal regression (generalized tobit), ordered logit and probit models, multinomial logit model, poisson regression, Heckman selection models, bivariate probit regression, stereotype logistic regression.

(3) Time series estimation commands: Box-Jenkins models, regressions with ARMA errors, models of autoregressive conditional heteroskedasticity, unit root tests (including those for panel data), correlogram estimation, vector autoregressions (basic and structural), impulse response functions, variance decompositions, vector error–correction models (cointegration), dynamic factor models, and state-space models.

(4) Panel data estimation commands: linear fixed effects, linear random effects, panel-data models using generalized least squares, instrumental variables panel data estimator, panel-data logit models (including conditional logit), panel-data probit models, panel-data poisson regression, panel-data models using generalized estimating equations, linear mixed (multi-level) models and Arellano-Bover/Blundell-Bond dynamic panel-data models.