

# **Important Things to Know about Stata**

# **Installing Keyserved Stata 7.0**

## **What is the KeyServer?**

The KeyServer allows you to copy, use and share commercial, copyrighted software for Windows machines. The publishers have granted licenses for their products because the use of these products is controlled and monitored by the Dartmouth KeyServer.

The way in which the KeyServer works is as follows: Users install Stata on their local computer, as well as an additional program called *KeyAccess*. Then, whenever Stata is started, *KeyAccess* sends a message across the Dartmouth network to the KeyServer to check to make sure the number of concurrent users doesn't exceed the number of licenses purchased. If there is not a license available, *KeyAccess* will run in the background, checking to see when a license is available. You will be notified automatically when a license is available. KeyServed applications cannot be run unless you are connected to the Dartmouth network.

## **Installing Stata 7.0 for Windows 95/98/2000 Professional**

Stata 7.0 is a keyserved application that is available from the Wilson fileserver.

In order to use Stata 7.0 on your Windows machine, you must install both the *KeyAccess* program, as well as the Stata 7.0 application on your computer. Your computer needs to be on the Dartmouth network in order to install and run keyserved applications.

### **Check to see if the KeyServer program is installed on your computer:**

Click the Start menu and drag to Find/Files or Folders (in Windows 2000, drag to the Search menu). Type the word Key and press <Return>. If you have the file entitled keyacc32.exe in your list of files that have "key" in them, *KeyAccess* is running on your computer.

### **To install the KeyServer program:**

Wilson is the name of the file server that contains the KeyServer software and also the KeyServed applications (i.e. Stata 7.0) that are currently available to the Dartmouth community on Windows 95/98/2000 computers. To install *KeyAccess* or Stata from the Wilson file server, you first need to map a drive to the Wilson\Public Files folder.

To do this, click **Start**, and then **Run**. In the **Open** box, type <\\Wilson>, then click **OK**. A window will open listing all of the printers and files available from Wilson. right-click on the folder labeled **Public Files**. Then click the **Map Network Drive** option. The next available drive letter on your computer will be the default. Make sure the **Reconnect at Logon** is *not* checked. Click **OK**.

**Note:** Although you need to have a drive mapped to Wilson to install *KeyAccess* and Stata, you do not need it to run keyserved applications.

## Installing KeyAccess

After you map a drive to the **Public Files** folder on Wilson, start *Windows Explorer* and open that drive if it isn't already open on your screen. Open the **Licensed Software** folder, the **KeyServed Software** folder, and then the **KeyAccess Client Software** folder. double-click the **KSClient** icon to install KeyAccess on your computer. Click **Next** to accept the defaults and the installation will begin. In the **Choose a Network Protocol** section, type **keyserver.dartmouth.edu**. Click **Continue**, then **Close**, then restart your computer.

KeyAccess should start automatically. (if you don't want KeyAccess to start automatically, you will need to remove it from your Startup folder.) If KeyAccess does not start automatically, you can run it from **Programs/KeyServerClient/KeyAccess**.

## Customizing KeyAccess

Open the KeyAccess window to configure the program. It is recommended that you select **Remove messages after 5 minutes**. This allows messages you receive from the KeyServer to be automatically dismissed without having to click **OK**. This option prevents KeyServer messages from blocking unattended backups or file transfers.

If you want your clock to be set to the same time as the KeyServer's clock, check the box next to **Synchronize clock with KeyServer**.

Check the box next to **Give hints for efficient use** if you want to be reminded not to run KeyServed applications from the server across the network. Running applications over the network is much slower than running them on your local computer.

In the **Connection** box, **keyserver.dartmouth.edu** should appear – if it doesn't, type it in then click **Logon**.

## Installing Stata 7.0 on your Windows Computer

Once KeyAccess is installed on your computer, the next step is to install Stata. To use a program controlled by the KeyServer efficiently, Stata should be installed on your computer and the program launched from your computer. Do not start a program from a file server (i.e. Wilson).

Begin by mapping a drive to Wilson\Public Files as described above. After you've mapped the drive, locate the Stata folder, which is in Public Files/ Licensed Software/ Key Served Software/ No Support folder. Open the Stata folder, then double click the Stata 7 folder. Double click the Setup.exe icon and follow the instructions as they appear on the screen. If you need more information, double click the **Readme** file in the same folder.

## Running Stata 7.0 on your Windows Computer

You run keyserved Stata 7.0 just like you would run an application that's installed on your computer. Either double click the icon on the Desktop or select it from the **Programs** menu. If you

want to increase the amount of memory allocated to Stata, you can use the **set memory** command. Just type **help memory** to learn more about this command.

Stata is looking in one directory for files. To see what folder this is, choose filename from the file menu. The folder that it shows you is the default folder. You can change the default folder before starting Stata by right clicking on the shortcut icon and choosing “properties.” On the shortcut tab, fill in the “start-in” box with the preferred folder. You can also just change your current directory using the **cd** command. Type **help cd** to learn more about this command.

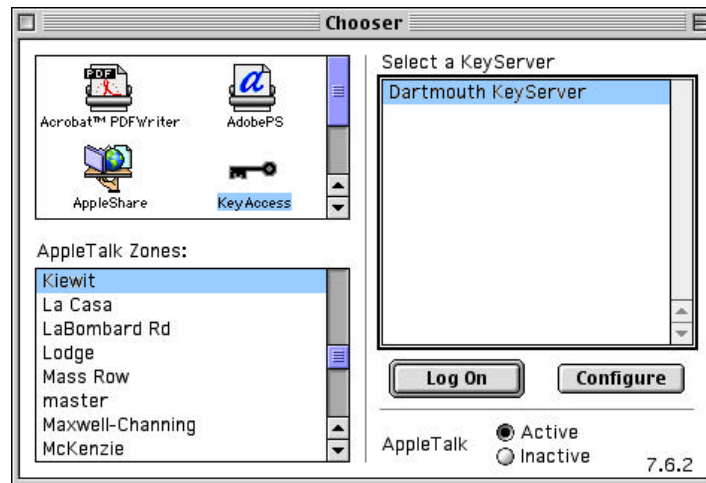
Since Stata 7.0 is keyservered, once you are finished using Stata, please quit the application so that others may have access.

## **Installing Stata 7.0 for Macintosh**

Stata 7.0 is a keyservered program that is available from the PUBLIC fileserver. In order to use Stata 7.0 on your Macintosh, you must install both the KeyAccess program, as well as the Stata 7.0 application on your computer. Your computer needs to be on the Dartmouth network in order to install and run keyservered applications.

### **Check to see if the KeyServer program is installed on your computer**

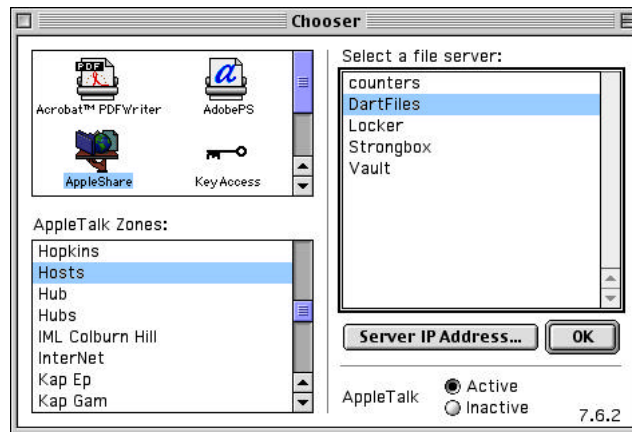
Click the Apple () menu at the top left corner of your screen and drag to Chooser. Select **KeyAccess**, Zone **Kiewit** and then single click the **Dartmouth KeyServer** on the right side of the Chooser window. Then click <Log On>. Your computer will be logged into the Dartmouth KeyServer.



If the KeyAccess icon is not present in your chooser window, you need to install KeyAccess on your computer as follows:

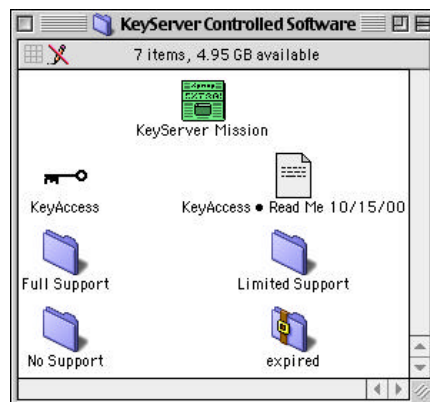
1. Open PUBLIC from your Macintosh as follows:

Click the Apple () menu at the top left corner of your screen and drag to Chooser. From the Chooser window, select AppleShare, Zone: Hosts and file server: Dartfiles and click <OK>.



Connect to PUBLIC as a Guest and click <Connect>. Click <OK> at the next window (do not load PUBLIC when you start your computer). The PUBLIC icon will appear on your computer desktop.

2. Once PUBLIC is on your desktop, double click to open, then double click the **Licensed Software** folder. Open the **KeyServer Controlled Software** folder. Locate the file called **KeyAccess** (see picture below). Drag that file to your computer – it should be placed in the System folder/ Extensions folder of your computer. After copying this file to your computer, restart and then login to the keyserver using the instructions above.



## Installing Stata 7.0 on your Macintosh

After verifying your computer has access to the KeyServer, you will need to install the Stata 7.0 software on your computer from PUBLIC.

Access PUBLIC as instructed above. Once PUBLIC is on your desktop, double click to open, then double click the **Licensed Software** folder. Open the **KeyServer Controlled Software** folder. Double click the **Limited Support** folder, then double click the **Stata** folder.

Drag the file “Stata 7.sit” to your desktop. Once that file is on your desktop, double click it and Stata 7.0 will install on your computer.

## Running KeyServed Stata from your Macintosh

You run keyserved Stata 7.0 just like you would run an application that's installed on your computer. Double click the file called Stata § which is a file that's in the Stata folder on your computer. Note that if you want to assign more memory to Stata, the "set memory" command does not work on a Macintosh. You have to set the amount of memory you'd like to use before starting the program. This is done the same way as any other program - just highlight the Stata § file and choose "get info" and then "set memory" from the menu. Just fill in the boxes for minimum and preferred memory with amounts of your choosing.

Stata is looking in one directory for files. To see what folder this is, choose filename from the file menu. The folder that it shows you is the default folder. You can also change your current directory using the **cd** command. Type **help cd** to learn more about this command.

Since Stata 7.0 is keyserved, once you are finished using Stata, please quit the application so that others may have access.

## Overview of Important Stata Commands

To get more information on any of the commands discussed below, you can use Stata's online help. For example, typing: **help summarize** will give you information on the summarize command, its syntax and options. If you are not sure of the command names, you can try Stata's online search command for looking up terms in its help. For example, typing: **search summary statistics** will provide a list of commands (including summarize) that refer to summary statistics. You can also work through Stata tutorials – just start the first one up by typing: **tutorial intro**.

### *Do Files:*

Stata can be used interactively – just type in a command at the command line, and Stata executes that command. A record of your commands is kept in the review window, and you can click on any past command to bring it back to the command line. Nonetheless, it can be very helpful to have a file of commands that are executed, rather than simply typing them in one at a time. Such a file of commands is called a do file, because it will be named myfile.do and to execute it you type the command **do myfile**. A useful command to put in your do file is **set more 1**. This tells Stata not to stop when a screen is filled and wait for you to hit a key to continue.

You can start a project interactively, but still be able to rerun your commands later if you use a command log, described below. Stata has a do-file editor so that you can write/edit your file of commands within Stata. Just click on the button of a pencil writing on paper (it actually looks a little like an envelope to me) and a blank screen that you can use just like a word processor will pop up. Note that just making changes in the window is not enough – you have to save the file before using the do command. You will find it useful to put comments in your do file. Any line starting with \* is interpreted as a comment and is not executed.

You may run into trouble if you have very long commands. For legibility, it is easiest to split them across two lines, but then Stata thinks the command is incomplete. To get around this, you can tell Stata that all commands will continue until a delimiter character is seen. I like to use ; as my delimiter. To do the same, put **# delimit ;** at the top of your do file. Then end every command in the file with ; so that your next line might be **set more 1;** and so on.

### *Log Files:*

It is important to use a log file to store the work you do in order to print it. For example, before starting the first problem set type **log using PS1.log** at the command line. This will create a file called PS1.log that will save the Stata commands you execute, as well as the output. *If you do not open a log file, your work will not be saved, so it is important to create this file before doing any work in Stata.* You can name your log file anything you want, but if it is not something.log, it will be in a strange Stata format rather than in a text format that can easily be edited in Word. Note that you will need to delete PS1.log before rerunning the program. If you are planning on having to rerun it, you may want to use **log using PS1.log, replace** from the start. Be careful with the replace option, though, as it will overwrite the existing file without warning you. When you are done, you just type **log close** to stop writing to the file.

A related command is **cmdlog**, which only saves the command you type, not the output produced. This is useful if you want to be able to recreate what you've done, or make a lot of mistakes interactively. When you are done, type **cmdlog close**. Then you can turn the cmdlog file into a do file and run it again. So for example, you could type **cmdlog using PS1cmd.do** as you started work on the problem set. After closing the file, you could open PS1cmd in the do-file editor and remove any mistakes, etc. (and making sure the first line is **log using PS1.log** ) before saving it. Then you would just type **do PS1cmd** at the command line and you'd have both a do-file and log-file as a record of your work.

### *Using and Saving Data Sets*

Usually, you can just click on a Stata data set, and Stata will start up using that data. This doesn't always work, for example if the data set was made on Windows and you're using a Mac, or the data set is too big for your default memory allocation. You can always open a data set from within Stata by typing **use mydata** where mydata is the name of the data set. If you can't remember the name, you can always choose "open" from the menu and browse your disk to find it. Stata will then type the appropriate command in the command window for you. If you have made changes to the data set, you can save the new data. Just type **save mynewdata**. If mynewdata already exists, Stata will not let you overwrite it – you will have to throw out the old one first. Alternatively, you can type **save mynewdata, replace** to overwrite it. Be very careful using the replace option. Once Stata has overwritten a file, it is gone!

### *Descriptive Commands:*

It is always a good idea to know what the data you are using are like. Typing **describe** will tell you what the names of the variables are, whether they are strings or numbers, and if the data set has been labeled, what the variable is. Creating labels for your variables is a good idea. While some variables can be given a fairly mnemonic name, for others it is useful to see a more in-depth description. You can type **label var myvar "description of what myvar is"** to assign a label to the variable myvar. You can also label values of categorical variables to make self-explanatory tables. For example, suppose you have a variable named sex that is 1 if the individual is male and 2 if they are female. You can type **label define sexlabel 1 "male" 2 "female"** and then **label values sex sexlabel** to associate the correct category labels with the numbers, but the variable is still a number.

In addition to knowing what kind of variables you have, you will want to see if the data make sense. Looking at some simple summary statistics is a good way to do this. Typing **summarize** (or just **sum**) will give you the number of nonmissing observations, the mean, the standard deviation, the minimum and the maximum for every variable. If you just want to look at one or two variables, type **sum var1** or **sum var1 var2**. Sometimes you want more detail about the distribution of a variable. Typing **sum var1, detail** will give you additional statistics, such as the median. For categorical variables, you may just want to look at a frequency distribution. The **tabulate** (or just **tab**) command is what you need. Typing **tab x1** will give you the frequency distribution of the variable named x1. Two-way tables are also possible. Typing **tab x1 x2** will just give you the frequencies. To get per row, column and cell percentages, type **tab x1 x2, row col cell** instead. Note that cross-tabs with too many cells are not possible. You only want to do this for variables

with a fairly limited number of values. If a variable has a value label, the labels, rather than the numbers will appear in the table. More complicated tables can be made using the **table** command. Assuming you have the variables wage, sex and year in your data, you could type **table year sex, c(mean wage)** to look at average wages for males and females by year. Other descriptive statistics (up to 5 total) can be included in parentheses. For example, **table year sex, c(mean wage median wage n wage std wage max wage)** would add in the median wage, the number of observations, the standard deviation and the maximum. The descriptives don't have to be all about one variable. Suppose you also have an education variable. You could type **table year sex, c(mean wage std wage mean education std education)** to get the mean and standard deviation of both wages and education by sex and year.

Data can also be summarized graphically. To look at a histogram for the variable x1, you just type **graph x1**. Sometimes you will want to break the histogram into smaller intervals, so you need to change the **lbin@number**. The bin size is the number of bars in the histogram. For example: **graph x1, bin(30)** [the bin can range from 2 to 50, and has a default of 5.] If you give 2 variable names, say you type **graph x y**, then you will get a plot of x against y. There are a lot of options to the graph command that make a better-looking graph. You can type **help graph** to investigate these options further. Stata will not save graphs in the log file! You can print them when they appear on the screen by clicking on "file" and then "print graph". You can also save the graph to a separate file that can be printed later by typing, for example: **graph x1, bin(30) saving(graph1)** or simply by clicking on "file" and then "save graph". Then, if you want to see the graph later you can type **graph using graph1** which brings the graph back up, and you can print as before. Finally, you can copy graphs directly into a Word document by clicking on "edit" then "copy graph", and then going to the document and pasting the graph.

Occasionally, you might want to actually look at your data. Typically you only do this with very small data sets, for a select set of variables, or using the "if qualifier" (described below). Typing **list** will print out all of the data. Alternatively, **list x1 x2 x3** will only print the variables named x1, x2 and x3.

### *Creating New Variables*

To create new variables, use the **generate** command (or **gen**) followed by the name of the new variable and an expression defining it. This command is best described by examples:

```
gen xplusy=x+y  
gen halfx=x/2 (or gen halfx=x*.5)  
gen xsquared=x^2 (or gen xsquared=x**2 either form of exponentiation works)  
gen logx=log(x) (or gen logx=ln(x) either form implies the natural log)
```

There are a few special types of variables for which need additional discussion. Dummy variables can be created by using an expression that evaluates to either true (=1) or false (=0). Suppose that you want a dummy variable equal to 1 for males, 0 for females and currently you have a variable named sex that is 1 for females and 2 for males: **gen male=sex==2** (or **gen male=sex~=1** will also work). It is important to note that for expressions that are either true or false, Stata requires double equal signs (==). The single equal sign is only for assigning a value. Not equal is ~=. Other possibilities are: > (greater than), < (less than), >= (greater or equal than), <= (less or equal than).

These same expressions can be used with the “if qualifier” (described below). To combine expressions, use **&** for and, and use **|** for or. So, typing **gen whitemale=(male==1 & white==1)** will create a dummy for white males and typing **gen blkorhisp=(black==1 | hispan==1)** will create a dummy for being black or hispanic.

Dummy variables can also be created for categorical variables by using an option to the **tabulate** command. Suppose that **marstat** was a variable equal to 0 for never married, 1 for married, living with spouse, and 2 for married, not living with spouse. Typing **tabulate marstat, gen(mardum)** would result in both a frequency tabulation of **marstat**, and the creation of three dummy variables: **mardum1** is 1 if **marstat**=0, and is 0 otherwise; **mardum2** is 1 if **marstat**=1, and is 0 otherwise; **mardum3** is 1 if **marstat**=2, and is 0 otherwise.

Lagged variables are also easy to create, as long as you know the data is in the correct order. Stata has an internal variable **\_n** that is the observation number. Typing **gen lagx=x[\_n-1]** will create the lag. To make sure data is in order, you need the **sort** command. Suppose this is time series data with one observation per year. Assuming you have a variable named **year**, just type **sort year** before generating the lag. Panel data is trickier. Suppose you have 5 years of data for each of 50 states. Then type **sort state year** and then **by state: gen lagx=x[\_n-1]** to properly make lags for each state. If you do not start with **by state:** then the first observation of the next state will assign the last observation of the previous state as the lag.

A fancier, more powerful version of **gen** is **egen**. It is most useful for when you want to attach a summary statistic to each observation in the data set. For example, suppose you have stock data and want a market return. Typing **egen mktreturn=mean(return)** will create the variable **mktreturn** that is equal to the mean of all of the firm returns in the data set. Alternatively, typing **egen indreturn=mean(return), by(industry)** would create a new variable that is the mean return for each firm within a firm's industry.

In all cases if the variable you want to define already exists, you just want to redefine it, you need to use the **replace** command instead of generate. Alternatively, you could type **drop newx** and then **gen newx=expression**. You cannot use **gen** for a variable that exists, and you cannot use **replace** for a variable that does not exist.

### *Using the “if qualifier”*

Any Stata command can be carried out for just a subset of the data by using the “if qualifier”. For example, you would type **sum x y z if age>=18** to get descriptive statistics only for adults. An alternative way to define a dummy variable for males would be the two-step process of **gen male=1 if sex==2** and **replace male=0 if sex==1**. Essentially, you just add **if expression** to the end of any command. The one exception is that the “if qualifier” comes before options, that is, before a comma. Thus, to get detailed summary statistics for just adults, type **sum x y z if age>=18, detail** or to do a regression (see below) on just adults, but with standard errors corrected for possible heteroskedasticity, type **reg x y z if age>=18, robust**. It is important to remember that Stata considers missing values, represented by a period, to be the highest possible number. Suppose you want to create an income variable that is only defined for people with positive earnings. If you type **gen incwear=income if earnings>0** you will include those with missing earnings. Instead, you should type **gen incwear=income if earnings>0 & earnings ~.**, to properly define your variable.

## Running Regressions

The syntax for running regressions is similar for most specifications. The first word is the type of regression you want to run. The second word is the dependent variable, followed by a list of independent variables. Last, you can type a comma followed by options. Some examples:

<b>reg y x1 x2</b>	“reg” means do OLS – here you’re regressing y on x1 and x2
<b>reg y x1 x2, level(90)</b>	the “level(x)” option changes the reported confidence bands to x%
<b>reg y x1 x2, nocons</b>	the “nocons” option runs the regression with no constant term
<b>reg y x1 x2, robust</b>	the “robust” option corrects the standard errors for heteroskedasticity
<b>reg y x1 x2, cluster(state)</b>	the “cluster” option corrects for heteroskedasticity & within state correlation
<b>reg y x1 x2 [w=z]</b>	WLS where z is the weight
<b>predict yhat</b>	predicts fitted values after the last estimation command
<b>predict uhat, resid (or predict uhat, r)</b>	predicts residuals after an estimation command. Note that it is the option resid that makes it the residual, not naming it uhat instead of yhat!
<b>test x1=x2</b>	run after “reg y x1 x2”, tests whether coefficients on x and z are the same
<b>test x1 x2</b>	run after “reg y x1 x2”, tests whether coefficient on x and z each equal 0 (jointly)
<b>probit y x1 x2</b>	run a probit regression, reporting coefficient estimates
<b>dprobit y x1 x2</b>	same as probit, but reports changes in probabilities instead
<b>logit y x1 x2</b>	run a logit regression, reporting coefficient estimates
<b>tobit y x1 x2, ul</b>	run a tobit regression where y is topcoded, reporting coefficient estimates
<b>heckman y x1, select(z x1)</b>	run a heckman selection corrected regression of y on x1, identified by z
<b>areg y x1, absorb(year)</b>	regression including year fixed-effects (allows robust option)
<b>xtreg y x1, fe i(year)</b>	alternative regression including year fixed-effects (no options allowed)
<b>xtreg y x1, re i(year)</b>	regression including random effects for year
<b>reg y x1 (z)</b>	regresses y on x1, using z as an instrument for x1
<b>ivreg y (x1=z)</b>	alternative form of above IV regression
<b>reg y x1 x2 (z x2)</b>	regresses y on x1 x2, using z as an instrument for x1
<b>ivreg y (x1=z) x2</b>	alternative form of above IV regression
<b>tsset timevar</b>	prepares to use the following time series commands
<b>prais y x1 x2</b>	performs Prais-Winsten estimation (vs OLS from reg y x1 x2)
<b>prais y x1 x2, corc</b>	use the Cochrane-Orcutt method instead
<b>prais y x1 x2, corc two</b>	use the two-step Cochrane-Orcutt method instead
<b>newey y x1 x2, lag(4) t(year)</b>	performs Newey-West estimation assuming 4 years of correlation

## Bringing Data into Stata

For most of this class, you will be given data that is already in the form of a Stata data set. For your 40-level class, though, you will probably be starting with a text file or a spreadsheet (like Excel).

If you start with a spreadsheet (which is probably easiest), you should save it as a text file, tab-delimited or comma-separated. You can then read it into Stata using the **insheet** command, e.g. **insheet using mydata.csv**. If you have data that are too big for one Excel sheet, you could do it piece by piece and then **append** the data sets.

In general, if you have a large data set, you will need to read in the data directly using either **infile** or **infix**. The best choice depends on your data, so you will want to type **help infile** and **help infix**

to see how each command works. You may then need to go look at the manual for more examples. Consider the simplest case where you have a text file organized with rows as observations and columns as variables. Let's say your data file is named *data.txt* and the top of your data file looks like this:

```
21 15000 m
45 65000 f
```

You will need a “dictionary” file to tell Stata how to read the text file. You need to give this file the extension “.dct”, for example *mydict.dct*. The file should look like this:

```
dictionary using data.txt {

float   age      “age of respondent”
float   income   “in thousands of dollars”
str1    sex      “m = male, f = female”

}
```

The first column (float or str1, here) tells Stata what kind of variable to look for – float is a number, and str# is a “string” (meaning letters or a combination of letters and numbers) of length #. Here, your sex variable takes on the values “m” and “f”. Note that “float” is the default – you don't actually have to type it. The second column is the name of the variable that Stata will use – note that Stata is case-sensitive. The third column is optional, and contains labels for the variables that are reminders to you or information for anyone else using your data set. The last curly bracket is important – don't leave it off!

Once you've written and saved the dictionary file, you open Stata, and type the command **infile using mydict.dct**. Stata will then read the in *data.txt* using the dictionary from *mydict.dct*. It will tell you how many observations it read in, what the variables are, etc. Be sure that this agrees with your expectations!

You can also type data directly into Stata, but I wouldn't recommend it. If you're typing data in, use a spreadsheet like Excel, save the data as text, and proceed as above.

### *Using Stata as a Statistical Table (and Calculator)*

You can use the **display** command along with any expression that you would use in a **gen** command to use Stata as a calculator. Typing **display 2+2** will result in 4 being displayed and **display 3\*4** will result in 12, etc. This command is most useful in concert with the built-in statistical functions that let you skip looking up critical values in a table and can directly calculate p-values for you. For example, if you have a t-statistic that is 1.4 from a model with 25 degrees of freedom, typing **display ttail(25, 1.4)** will display .087, which is the p-value for a 1-sided t-test (just double it for a 2-sided test). Typing **display invttail(25,.087)** would display 1.4. Similar functions are **Ftail(ndf, ddf, F)** and its counterpart **invFtail(ndf, ddf, p)** for an F-statistic, F, with ndf and ddf numerator and denominator degrees of freedom respectively, and **chi2tail(df, x)** and **invchi2tail(df, p)** for a chi-squared statistic, x.

### *More Sources of Support*

The [Social Science Computing Lab](#) is the best on-campus source for support. A full set of Stata manuals should be available for browsing there. You probably don't want (or need) your own full set. However, if you are interested in ordering any of the available Stata resources, you can check out the bookstore at [www.stata.com](http://www.stata.com) and see what is available. Items of possible interest include an [Introduction to Stata](#) multimedia CD, a textbook-style reference called [Statistics with Stata 7.0](#), a Getting Started manual for [Windows](#) or [Mac](#), or for hardcore Stata fans, the Reference Manual [Abstract](#). None of these are necessary, though, as we will cover commands essential for the class.