ECONOMICS 400: ECONOMIC STATISTICS

Boone A. Turchi
Gardner Hall 200A
919-966-5348
email: turchi@email.unc.edu

Office Hours:
Tuesday, 4:45-5:45 p.m.
or by appointment
Web Page: https://sakai.unc.edu

This is the required introductory course in economic statistics for economics majors. It introduces students to the basic concepts of statistical description, probability theory, and statistical inference as they apply to economic analysis. In particular, the course will emphasize regression analysis, since economics students will be exposed to many regression-like analyses in their upper division economics courses. My goals for students in the course are twofold: (1) to achieve a rigorous understanding of the foundations of statistical theory, and (2) to gain real facility in performing statistical analysis on the computer. By the end of the course students will be both confident and capable using a sophisticated statistical software package to the point that they will be able to use it routinely in other courses and activities.

The format of the course is lecture/discussion and laboratory. Most weeks the laboratory will take place when/where you wish. All you will need will be your computer, the Stata program on your computer and the lab manual. There will be a final exam (35% of the final grade), two midterm exams (25% each), laboratory/problem sets (15%), and a "qualifying exam" that tests the student's ability to use the software package. *There are no make-up exams for missed midterms.* Students with an approved excuse *for a maximum of one missed exam* may have extra weight placed on the final exam, which must be taken at the regularly scheduled time and place. The qualifying exam will be "pass-fail"; however, students must pass this examination before they can receive credit for the course. You *must complete the qualifying exam before you will be allowed to take the final exam.*

Materials for Purchase:

Required Texts and Software:

Required material for this course consists of two books and statistical software program (Stata). The Groebner text comes bundled with online access to MyStatLab, an online homework program that we will use. One of the following two packages (Groebner text + Hamilton book + Stata program):

Package 1:

David F. Groebner, Patrick W. Shannon, Phillip C. Fry, and Kent D. Smith, "Business Statistics: A Decision-Making Approach. (9th Edition) Prentice-Hall 2014. ISBN:9780133098785 – (Printed textbook + MyStatlab with ebook) + Hamilton book + Stata program.

Package 2:

David F. Groebner, Patrick W. Shannon, Phillip C. Fry, and Kent D. Smith, "Business Statistics: A Decision-Making Approach. (9th Edition) Prentice-Hall 2014. ISBN:9780133098785 – (MyStatlab with ebook only). + Hamilton book + Stata program.

You could buy a used copy of the Groebner book (9th edition), but you must also buy *mystatlab* and the combination could be more expensive than the packages above.

Lab/Reference Manual: Lawrence C. Hamilton, *Statistics with Stata: Updated for Version 12.* Brooks/Cole Cengage Learning 2013. ISBN13: 978-0-8400-6463-9. (Also available as an e-book see http://www.cengagebrain.com/shop/search/9780840064639)

Required Statistical Software: This course will provide intensive instruction in the use of the Stata statistical package. Stata is an extraordinarily powerful statistical tool that comes in various versions. Ordering instructions and descriptions of the options available are contained in a separate handout. **Purchase of Stata is required for all enrolled students. I will assume that you have Stata available on your computer.**

Your E-mail Address: Every student must have a functioning UNC e-mail address, and you *must* be reachable through that address. Your UNC e-mail address must be the address that accompanies the official UNC on-line class roll.

Mac vs. PC: The University provides and supports Windows PCs to faculty. All course material is guaranteed to work on Windows PCs. Mac users having trouble with course material should consult User Services in the basement of the undergraduate library. In particular, the standard web browser on the Mac (Safari) apparently does not refresh web pages automatically. If you're having trouble accessing course web material, try ***refreshing*** the course web page.

A course outline and schedule follow. Both are *tentative* at this point. If we deviate from the schedule, I will keep you informed as to where you ought to be.

Tentative Course Outline

Tuesday			Thursday		
Activity/Date	**GSFS text**	**Hamilton****	**Activity/Date**	**GSFS text**	**Hamilton****
			1/8: Describing Data	Ch 1 Ch 2	Ch. 1 Ch. 3: Graphs
1/13: Describing Data	Ch. 2 Ch. 3	Ch. 5: Summary Statistics	*1/15: Describing Data*	Ch 3	Ch. 2 Data Mgmt*
1/20: Probability	Ch. 4	Ch. 2 Data Mgmt*	*1/22: Probability*	Ch. 4	Ch. 2 Data Mgmt*
1/27: Probability	Ch. 4	Ch. 2 Data Mgmt*	*1/29: Discrete Prob. Distributions*	Ch. 5	Ch. 2 Data Mgmt*
2/3: Discrete Prob. Distributions	Ch. 5	Ch. 2 Data Mgmt*	*2/5: Continuous Distributions*	Ch. 6	Ch. 2 Data Mgmt*
2/10: Sampling Distributions	Ch. 7	Ch. 2 Data Mgmt*	*2/12: Sampling Distributions*	Ch 7	Ch. 2 Data Mgmt*
2/17: Sampling Distributions	Ch. 7		*2/19: Estimating Means & Proportions*	Ch 8	
2/24: Estimating Means & Proportions	Ch. 8		***2/26: Midterm 1***		
3/3: Testing Hypotheses	Ch. 9		*3/5: Testing Hypotheses*	Ch. 9	
3/10: Spring Break			***3/12: Spring Break***		
3/17: Testing Hypotheses	Ch. 9		*3/19: Testing Hypotheses*	Ch. 9, 10	
3/24: Testing Hypotheses	Ch. 10, 11		*3/26: Linear Regression*	Ch. 14	Ch. 7
3/31: Linear Regression	Ch. 14	Ch. 7	*4/2: Linear Regression*	Ch. 14	Ch. 7
4/7: Multiple Regression	Ch. 15	Ch. 7,8	***4/9: Midterm 2***		
4/14: Multiple Regression	Ch. 15	Ch. 7,8	*4/16: Multiple Regression*	Ch. 15	Ch. 7,8
4/21: Multiple Regression	Ch. 15	Ch. 7,8	*4/23: Multiple Regression*	Ch. 15	Ch. 7,8

* When working on Hamilton's Chapter 2, Data Management, you may find the following online tutorial helpful: ***http://www.cpc.unc.edu/research/tools/data_analysis/statatutorial***

Another very useful web site for Stata can be found at:

http://www.ats.ucla.edu/stat/stata/default.htm

** Other readings from Hamilton will be assigned in conjunction with computer exercises.

☠ **Final Exam: Tuesday April 28th @ 12 noon**

ATTENTION: Place order immediately; the first Stata exercise will be assigned on January 22, 2015.

Instructions for ordering Stata

Introduction

All enrolled students are required to buy one of the versions of **Intercooled Stata** (or better) described below. **Intercooled Stata** is available as a **six-month** license, a **one-year** license or as a ***perpetual*** *license* (that is, once installed on your computer [or your next computer] it will run forever; the time-limited licenses stop working after six months or a year). **Intercooled Stata** can have as many as 798 right-hand-side variables in a model. **Intercooled Stata** can analyze data sets with as many as 2,047 variables and the only limit on observations is the amount of RAM on your computer. This means that **Intercooled Stata** can be used to analyze essentially any size problem that you can think of.

(Stata Corp. also offers a version of Stata called **Small Stata**, which is *not sufficient for use in this course.* **Small Stata** is limited to analyzing data sets with a maximum of 99 variables on approximately 1,000 observations; however, I have found the practical data limitations to be even more restrictive. It is simply not a serious tool for data analysis and I want you to use a serious tool.). In addition, there are more powerful versions of Stata available: Stata/SE, and Stata /MP. They are more powerful and more expensive. You can buy one of these, but **Intercooled Stata** will do virtually anything you could conceivably want to do.

You have a choice to make: Which version of **Intercooled Stata** do you want: six-month, one-year, or perpetual? If you plan in the future to write papers or theses using statistical analysis, or take advanced econometrics courses such as Economics 570, you might want to invest in **Intercooled Stata** with a perpetual license now.

Available Packages through Stata's GRADPLAN (Available for Mac or Windows computers):

1. Stata/IC 12: Six-month license - Product code ICGP6. Price: $69
2. Stata/IC 12: One-year license - Product code ICGPA. Price: $98
3. Stata/IC 12: Perpetual license - Product code ICGP. Price: $189

What is my Recommendation?

If you can afford it, I recommend, package #3. A primary purpose of this course is for you to achieve competency in statistics and in the *practice of statistics.* The Department wants you to use statistical analysis in your other economics courses, other courses in general, other campus activities (e.g., Daily Tar Heel, student government, etc.) and eventually, perhaps, in your post-college work. I can cite a number of alumni of this course who have gotten their first jobs out of college based in part on their knowledge of Stata. In particular, if you decide to take Econ 570 (Econometrics) as your advanced course, you'll already have the software and won't have to buy it again. If you're a senior who managed to avoid Econ 400 until your last semester, you can get by with package #1; however, you'd better be sure to pass the course this time around ;-). Other students should seriously consider buying the perpetual version; I realize it's expensive (maybe not so bad when you consider that current Econ 101 textbooks cost over $200) but you may regret purchasing the time-limited version when the license ends and you realize that you really *do* want to work further with the program. **A complete set of Stata manuals in PDF form is contained on the program disk that you receive.** Manuals are also on reserve at the Undergraduate Library. A separate handout contains the reserve list.

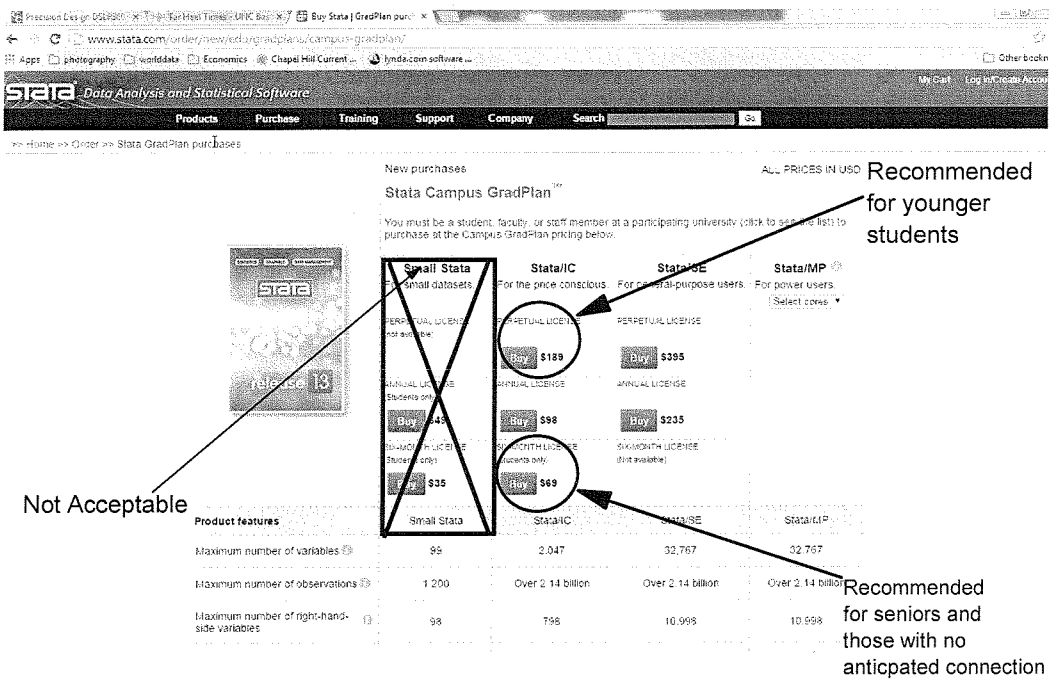
How to Order

1. Place your order with Stata by going to **http://www.stata.com/order/new/edu/gradplans/campus-gradplan/**
2. Fill out the online order form. ***Be sure to include your .EDU email address when ordering***. Typically, orders are shipped within 1 or 2 business days after the order is placed. Stata accepts Visa, MasterCard, Discover, and American Express. If paying by credit card, please include the expiration date with the card number. If you prefer, you may fax this information to Stata at 979-696-4601 or call Stata at 800-782-8272.
3. Stata will mail your software directly to you.
4. See picture of Stata GradPlan website on reverse of this page.

ATTENTION: Place order immediately; the first Stata exercise will be assigned on January 22, 2015.

Stata Grad Plan Website:

http://www.stata.com/order/new/edu/gradplans/campus-gradplan/

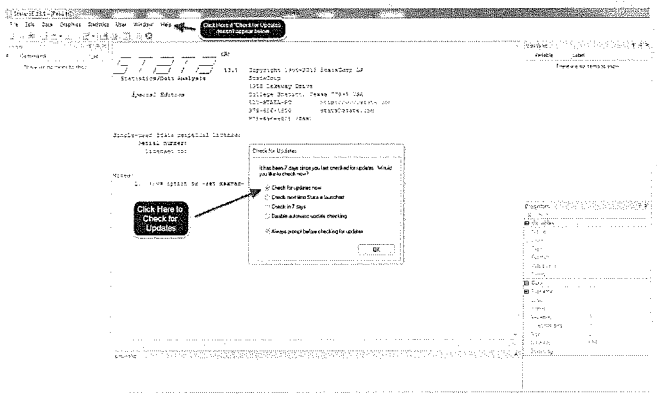


Updating Stata

This handout tells you how to update your copy of Stata

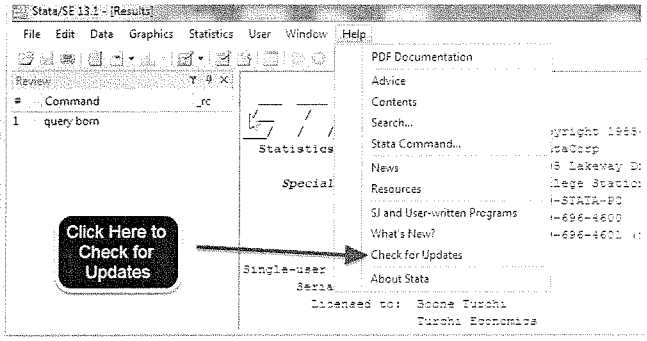
Start up Stata : You need "Administrator Privileges" to update your program.

You will either see the 'Check for Updates' box, as below, or you can click the "Help" menu item to see the choices there (see the second picture below this text. In either case, you want to check for updates.



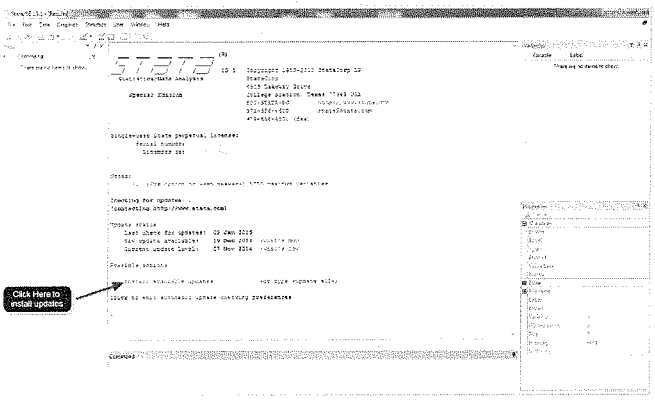
Updating Stata

Alternative way to check for updates



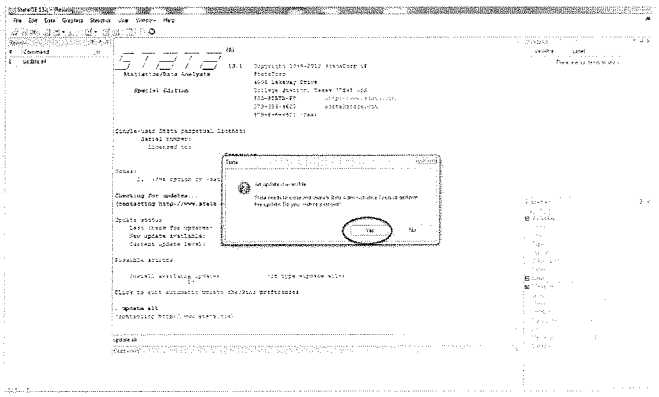
Updating Stata

Tell Stata to install available updates



Updating Stata

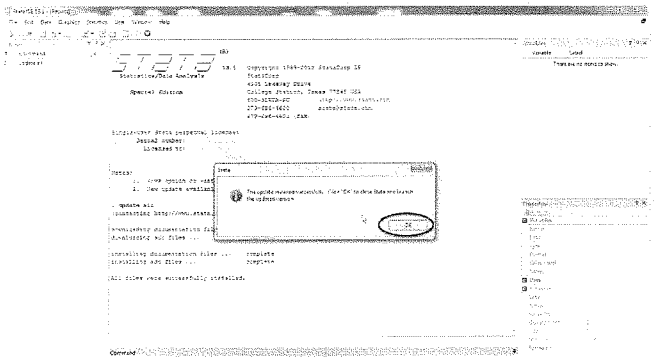
If updates are available, you'll see the notification below. Click 'Yes' to proceed.



Updating Stata

When Stata is done updating, you'll see the following message.

Click "OK" to close Stata and reopen the new version



Homework submission requirements.

You will be submitting two types of homework for grades in Economics 400:

(1) Book problems that will be assigned and which you will complete online using MyStatLab. These problems will be assigned to you on (roughly) a weekly basis. You will have approximately one week to complete the exercises for full credit. You will supply the answers through MyStatLab and they will be graded and recorded in your file. You are to do the book problems alone and not collaborate with other students. You can request assistance from Professor Turchi or the Teaching Assistants, but *the Honor Code will apply to all homework.*

(2) Computer Exercises will be assigned and turned in *on paper, at the beginning of class on the date due.*

- ✓ The exercise should be submitted at the beginning of lecture on the day that it is due. You have until **noon the following day** to turn in the exercise ***with penalty*** (see below). If you miss the grace deadline, do not bother to turn in the exercise; it will not be accepted. If the exercise is not turned in during lecture, you **must** turn it into the *receptionist in Room 107 Gardner Hall and ask her to put your exercise in* ***the Econ 400 Homework Box.*** *Exercises delivered any other place (e.g., slipped under a TA's or Professor Turchi's door) will not be accepted.*
- ✓ The exercise should be **stapled** before submission (there are public heavy duty staplers on the first floor of the Davis library and there is a stapler in the reception room of the Economics Department, GA 107) as well as at the main desk of the Global Center.
- ✓ Finally, be sure to sign the Honor Pledge on the front page of your exercise before turning it in.

Homework Grading Policy

If the student turns a computer assignment in late (but before noon the next day), he/she is going to get 50% *at most* even though he/she might do the exercise completely correctly.

For homework completed in MyStatLab, I may allow late submissions online, but only for a few days after the due date and with a grade penalty. More on this later.

When your final homework average is computed, your lowest homework grade (including a zero for assignments not turned in or turned in past the grace time) will be dropped.

HOW TO SOLVE IT

UNDERSTANDING THE PROBLEM

First. You have to *understand* the problem.

What is the unknown? What are the data? What is the condition? Is it possible to satisfy the condition? Is the condition sufficient to determine the unknown? Or is it insufficient? Or redundant? Or contradictory?
Draw a figure. Introduce suitable notation.
Separate the various parts of the condition. Can you write them down?

DEVISING A PLAN

Second. Find the connection between the data and the unknown. You may be obliged to consider auxiliary problems if an immediate connection cannot be found. You should obtain eventually *a plan* of the solution.

Have you seen it before? Or have you seen the same problem in a slightly different form?
Do you *know a related problem? Do you* know a theorem that could be useful?

Look at the unknown! And try to think of a familiar problem having the same or a similar unknown.
Here is a problem related to yours and solved before. Could you use it? Could you use its result? Could you use its method? Should you introduce some auxiliary element in order to make its use possible?
Could you restatethe problem? Could you restate it still differently?
Go back. to definitions.

If you cannot solve the proposed problem try to solve first some related problem. Could you imagine a more accessible related problem? A more, general problem? A more special problem? An analogous problem? Could you solve a part of the problem? Keep only a part of the condition, drop the other part; how far is the unknown then determined, how can it vary? Could you derive something useful from the data? Could you think of other data appropriate to determine the unknown? Could you change the unknown or the data, or both if necessary, so that the new unknown and the new data are nearer to each other?
Did you use all the data? Did you use the whole condition? Have you taken into account all essential notions involved in the problem?

CARRYING OUT THE PLAN

Third. *Carry out* your plan.

Carrying out your plan of the solution, *check each step.* Can you see clearly that the step is correct? Can you prove that it is correct?

LOOKING BACK

Fourth. *Examine* the solution obtained.

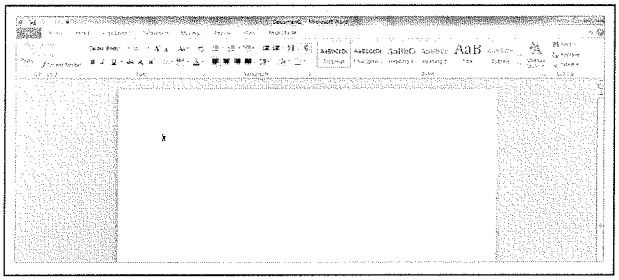
Can you *check the result?* Can you check the argument?
Can you derive the result differently? Can you see it at a glance? Can you use the result, or the method, for some other problem?

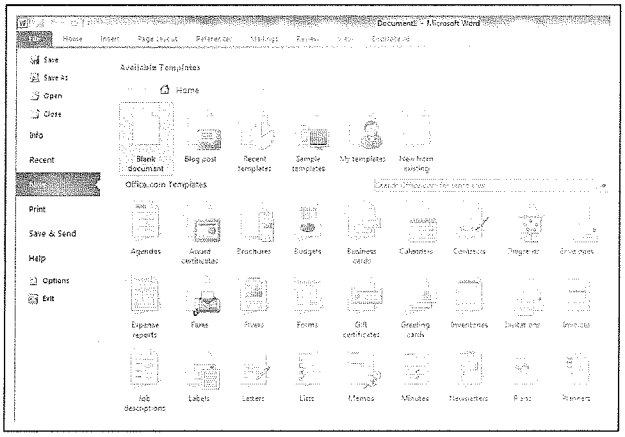
This excerpt comes from George Polya's wonderful book *How to Solve It (*Paperback - 224 pages Reissue edition, November 1, 1971 Princeton Univ Press. ISBN: 0691023565.) If you want to learn more about how to solve problems systematically and efficiently, then I strongly recommend that you either buy a copy of this book or check it out from the library.

Instructions for Completing Computer Exercises

Professor B. Turchi Economics 400

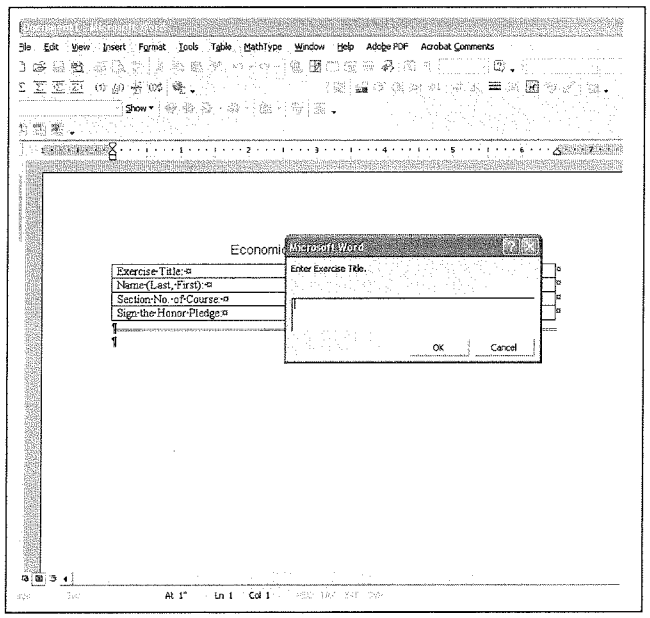
Throughout the semester you will be submitting a number of computer exercises that will require you to include graphics, tables and other material from your work in Stata and, possibly, other programs. In order to facilitate the collection, grading and return of these exercises, I require that you use a specific format in Microsoft Word. The format is contained in a Word "template", a file that Word uses to format your document. Many of you have probably used only the "normal" template that Word uses whenever it starts a plain document. So, say you open Word 2013 as you normally do. You'll see a screen with a blank document as in the picture on the left below. Clicking on the "file" tab you can create a new document by selecting "New" on the left side of the page. Word will give you a choice of templates, automatically selecting "Blank Document" as shown in the picture on the right. This is the "Normal" template. As you can see you have a large choice of templates that



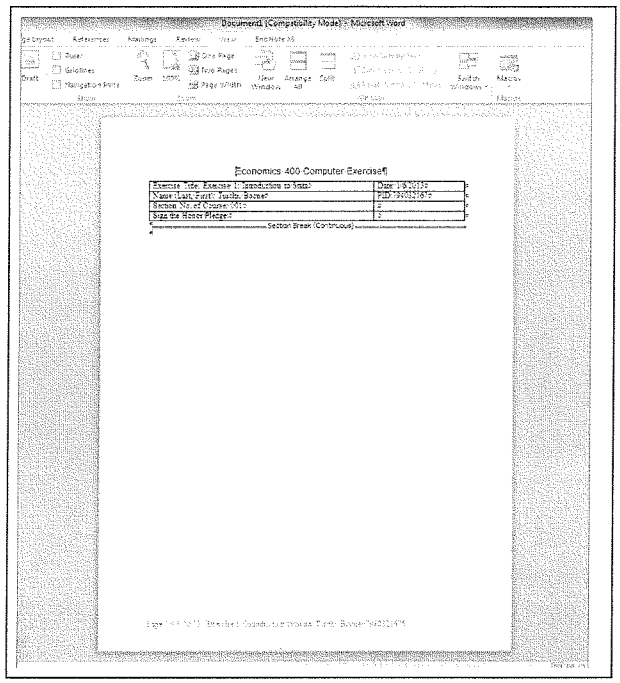


Microsoft has prepared for you.

The template that I've prepared for you to use can be filed in one of the Office folders where Word can find it (e.g., in the following directory: "C:\Users***yourname***\AppData\Roaming\Microsoft\Templates"). Or, you can file the template in a place that's convenient for you, say, on your desktop or in your "Econ 400" folder. For the moment assume that you found the template, "*Ec400TP.dot* " (or "*ec400TP.dotx"* if you're using Word 2007 or later). The easiest way to start a new computer exercise is to double-click on the template and Word will open with the following screen:



Fill in the requested information, clicking "OK" each time a new entry is requested. When you're done, immediately go to "File/Save As/" and save the file as a Word ***document*** with a name you choose (e.g., 'Exercise1.docx"). Then, you're in a position to edit your exercise prior to turning it in. Before you add any text the document will appear as below:



A note about typeface:

When you include Stata output in your exercise, be sure that the *output* is shown in **courier new** font. Stata often relies on a "fixed-pitch" font to make tables and other printed graphics line up and otherwise look pretty. When you paste the output into a Word document, Word will likely change the font to "times new roman", or other variable pitch font. The resulting output will be a mess unless you change its font to courier new or other fixed pitch font.

To get the template: Go to the course website and download it as per instructions. The file's name is **Ec400TP.dot** (For Word versions before 2007) or **Ec400TP.dotx** (Office 2007 or later). Right-click on the template file and select "Save Target As:"

1. The easiest way to use the template is to save the template file any place you wish (on your desktop, for example). Then, any time you want to start a new exercise, simply ***double-click on the template file***. Microsoft Word will open a new document with the **Ec400TP.dotx** template and will give you the dialog boxes to fill in required information. *After filling in the required information, you must immediately save your document (as a document, not a template) under a new name as described above.* Then, you can edit the document at your convenience.

2. *If you don't have your own computer*, you need to store the template file where you can have access as you need to use it.

    - If you always work on an ATN lab computer (say in the Undergraduate Library), you can save the template in a directory on your personal "H" network drive. It will be available on every ATN lab computer.

    - Otherwise, you might save the template file on a USB flash drive that you have available when you need it.

Professor Boone A Turchi

Subscripts, Summation, Variables and Functions[1]

Throughout this course we will use certain symbols to distinguish between the numbers in a set of data, and to indicate the sum of such numbers. For example, we may wish to distinguish between the monthly sales of a certain business, and then sum these monthly sales to get the yearly sales. To do this, suppose we let the symbol x denote the monthly sales of this firm. Furthermore, we will add a subscript to this symbol to denote which month is being represented. Thus, x_1 = sales in first month, x_2 = sales in second month, and so forth, with x_{12} = sales in the twelfth month. To illustrate, if sales in the sixth month were 120 units, then we would write $x_6 = 120$. The notation x_i thus stands for "sales in the i th month," where i can be any number from 1 to 12; that is, $i = 1, 2, \ldots, 12$. The dots in this last expression are used to indicate "and so on."

Now, assume that we want to sum the sales for all 12 months in a year, which is

$$x_1 + x_2 + \cdots + x_{12}.$$

Another way of writing this sum is to use the Greek letter Σ(capital sigma). This symbol is read as "take the sum of." At the bottom of this Σ sign we usually place the first value of i which is to be included in the sum. The last value of i to be summed is usually placed at the top of the sum sign. Thus,

$$\sum_{i=1}^{12} x_i$$

is read as "sum the values of x_i, starting from $i = 1$ and ending with $i = 12$." That is,

$$\sum_{i=1}^{12} x_i = x_1 + x_2 + \cdots + x_{12}.$$

Similarly, suppose we wanted the sum of only the last seven months in the year. This sum would be written as follows:

$$\sum_{i=6}^{12} x_i = x_6 + x_7 + \cdots + x_{12}.$$

In statistics we will often not know in advance what the final value in a summation will be. For example, we know that we want to sum a set of sales values, but we don't know how many values there are to be summed. To designate this situation, we will let the symbol n represent the last number in the sum (where n can be any integer value, such as 1, 2, 3....). The notation

$$\sum_{i=1}^{n} x_i = x_1 + x_2 + \cdots + x_n.$$

is thus read as "the sum of n numbers, where the first number is x_1, the second is x_2, and the last is x_n." In summing monthly sales over a year, we would thus let $n = 12$, so that $\sum_{i=1}^{n} x_i = \sum_{i=1}^{12} x_i$.

I should perhaps mention that I will sometimes omit the limits of summation, and simply write Σx_i. This notation should be interpreted to mean "sum all relevant values of x_i." In these instances I will make sure that you always know what the relevant values of x_i are. Also, I might point out that the choice of symbols in designating a sum of numbers is often quite arbitrary. For example, we might have used the letter y to denote monthly sales (instead of x), and used the letter j as a subscript (instead of i). In this case $\sum_{j=1}^{12} y_j$ would denote the sum of the twelve monthly values.

Double Summations

[1] Adapted from James L. Murphy.

Occasionally we will find it convenient to use two subscripts instead of just one. In these instances the first subscript indicates one characteristic under study, and the second subscript some other characteristic. For example, suppose we let x_{ij} = sales in the ith month by the jth salesman. The notation $x_{6,2} = 15$ would indicate that in the sixth month ($i = 6$), salesman number 2 ($j = 2$) sold 15 units. Using the same procedure as described above, we can denote the total sales over 12 months by the jth salesman as the sum of x_{1j} (sales in the 1st month by the jth salesman) plus $x_{2j}, \ldots$, plus $x_{12,j}$ (sales in the 12th month by the jth salesman). That is,

Total sales by salesman j:

$$\sum_{i=1}^{12} x_{ij} = x_{1j} + x_{2j} + \cdots + x_{12,j}.$$

As another example of a similar type of sum, suppose we want to denote the sum of sales in the i th month (where i is some number between 1 and 12) over all the salesmen in the company. If we let m = total number of salesmen, then this sum is x_{i1} (sales in month i by salesman #1) plus $x_{i2}, \ldots$, plus x_{im} (sales in month i by salesman m). That is,

Total sales in month i:

$$\sum_{j=1}^{m} x_{ij} = x_{i1} + x_{i2} + \cdots + x_{im}.$$

Finally, we might wish to sum over all months ($i = 1, 2, \ldots , 12$) and all salesmen ($j = 1, \ldots , m$). This sum could be written as:

Total sales over all months and all salesmen:

$$\sum_{\text{All j}} \sum_{\text{All i}} x_{ij} = \left\{ \begin{array}{l} x_{11} + x_{12} + \cdots + x_{1m} \\ +x_{21} + x_{22} + \cdots + x_{2m} \\ \vdots \\ +x_{12,1} + x_{12,2} + \cdots + x_{12,m} \end{array} \right\}.$$

Professor Boone A Turchi

Variables and Functions

Variables

Variables and the relationship between variables represent an important part of statistics. Hence, it is important that we define these concepts carefully.

A variable is a quantity which may assume any one of a set of values. For example, we might describe the worth of a common stock by the variable "current worth on the stock market." The values of this variable are the different prices the stock can assume. Or, we might be interested in describing how well a specific brand of alkaline battery works by defining the variable "the length of time before failure when in constant use." The values of this variable are the various times it might take before the battery fails.

Variables are often classified according to whether their values are *discrete* or *continuous*. With a discrete variable, the values of the variable are individually distinct; i.e., they are separable from one another. The price of a common stock, for instance, represents a discrete variable because the prices a stock can assume are all distinguishable from one another. The following examples also represent discrete variables:

1. the number of defectives in a production lot,
2. the amount of advertising expenditure a certain company plans for next year,
3. the amount of federal income tax owed by an individual or a corporation.

Most discrete variables represent some quantity which can be "counted."

When a variable is *continuous*, the values of this variable are indistinguishable from one another. Quantities which are *measured* are usually continuous variables; for example, measures of time, weight, length, and area typically represent continuous variables. Thus, in our earlier example, the time it takes a battery to fail represents a continuous random variable. With a continuous variable, there are always an infinite number of values of the variable.[2] The following variables are also continuous.

1. the percentage increase in the consumer price index last month,
2. the amount of natural gas available in the U.S.A. next year,
3. the quality of the air in Los Angeles yesterday, measured on a scale which can be *any* number from 0 to 100.

One of the practical difficulties with continuous variables is that the devices used to measure such variables are usually read only in a discrete manner. For example, the variable "amount of gasoline needed to fill a car" is clearly a continuous random variable, since this amount may be *any* value between zero and the capacity of the gas tank. From a *practical* point of view, however, this variable is discrete because most gas pumps cannot be read (at least accurately) beyond a few decimal points (usually 1/100 of a gallon). *For most statistical analysis it makes little difference if we treat such variables as discrete or continuous, although a continuous variable is often easier, to manipulate than is a discrete variable which has many different values.*

Functions

If a unique value of some variable y is associated with every possible value of another variable x, then the variable y is said to be "a function of" the variable x. To illustrate a functional relationship, suppose we let x represent the number of gallons of gasoline you purchase at a service station, and let y be the amount of money you must pay for this gasoline. In this case y is a function of x [written $y = f(x)$] because it is known exactly what (unique) amount *(y)* you

[2] With a discrete variable, the number of values may be either finite or infinite.

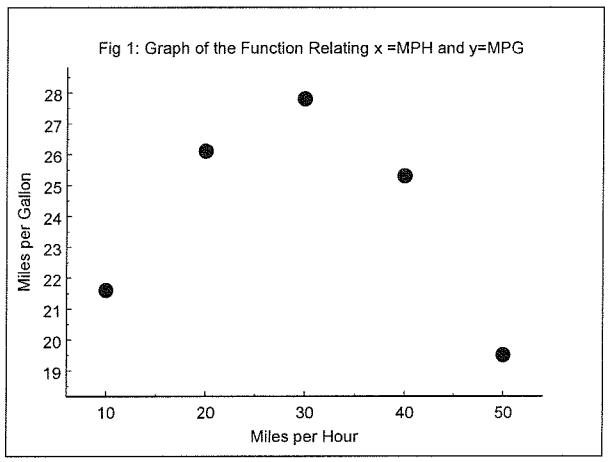
will be charged for every possible gasoline purchase *(x)*.

There are three commonly-used methods for describing a functional relationship: 1) a table, 2) a graph, and 3) an equation. As we will illustrate below, the first two of these methods work well for discrete functions, while the latter two work well for continuous functions.

Discrete functions. By a discrete function, we will mean any function where x is a discrete variable. If x is discrete, then y must be discrete as well. To illustrate a discrete function, suppose the Environmental Protection Agency (EPA) is testing a new car to determine its gas mileage (y) at various speeds (x). This car was tested at x = 10, 20, 30, 40, and 50 miles per hour (MPH). The miles per gallon (MPG) at these speeds were y = 21.6, 26.1, 27.8, 25.3, and 19.5 respectively. This information is shown below.

miles per hour (x)	miles per gallon [$y = f(x)$]
x = 10	y =f(10) = 21.6
x = 20	y =f(20) = 26.1
x = 30	y =f(30) = 27.8
x = 40	y =f(40) = 25.3
x = 50	y =f(50) = 19.5

Note that the variable x is discrete since all possible values of this variables are distinguishable from one another -- i.e., they are individually distinct. It is important to understand the symbolic notation in writing functions. For example, the notation *f(10) = 21.6* means that when x = *10,* the value of *f(x)* is y = *21.6.* Similarly, *f(50) = 19.5* means that y = 19.5 when x = 50.



We must resist the temptation to connect the points in Figure 1 with a line, since such a line might incorrectly lead a viewer to assume the function is defined for speeds other than x = 10, 20, 30, 40, and 50 MPH. It may be possible to define a function which relates additional values of x to y, but in this example the function is defined only for five x-values. When the number of x-values is large, a formula is often useful in describing a functional relationship.

Continuous Functions. When the random variable x is continuous, then the functional relationship between x and y usually must be expressed as a formula. Consider a simple example, where the variable x is temperature measured on the Fahrenheit scale, and the variable y is temperature measured on the Celsius scale. For converting values from the Fahrenheit scale (x) to the Celsius scale (y), the following functional relationship is used:[3]

$$y = \frac{5}{9}x - \frac{160}{9}.$$

[3] We could have written this formula as $f(x) = \frac{5}{9}(x - 32)$.

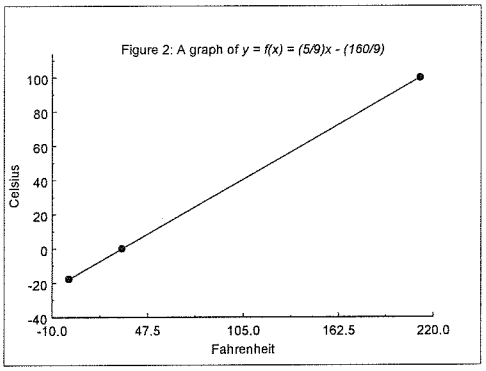
Note that this relationship represents a function since a unique value of y is specified for each value of x. The function is continuous since the values of x are indistinguishable. Now, suppose we want to graph this function. First, we recognize that it is a straight line, since the exponent of the variable x is 1. To graph a straight line we only need two points. The easiest two points to take are usually the one where $x = 0$ and the one where $y = 0$. When $x = 0$,

$$y = \tfrac{5}{9}(0) - \tfrac{160}{9} = -17.78$$

Similarly, when $y = 0$ we can solve for x as follows:

$$0 = \tfrac{5}{9}x - \tfrac{160}{9}$$
$$\tfrac{5}{9}x = \tfrac{160}{9}$$
$$x = 32.$$

We now have two points on our function, (0, -17.78) and (32, 0). This function is graphed, in Figure 2, by connecting these two points.



From either the straight line in Figure 2, or the function itself, we can solve for one additional point most of us are familiar with, namely the boiling point of water. Since water boils at 212 degrees Fahrenheit, $x = 212$. The comparable value of y is F(212):

$$y = F(212) = \tfrac{5}{9}(212) - \tfrac{160}{9},$$

or

$$y = F(212) = 100,$$

which is the Celsius boiling point of water. We could have solved for any one of the infinite number of different y-values in a similar manner.

This page intentionally left blank

18 Language syntax

Contents

18.1 Overview

With few exceptions, the basic Stata language syntax is

[by *varlist*:] *command* [*varlist*] [=*exp*] [if *exp*] [in *range*] [*weight*] [, *options*]

where square brackets denote optional qualifiers. In this diagram, *varlist* denotes a list of variable names, *command* denotes a Stata command, *exp* denotes an algebraic expression, *range* denotes an observation range, *weight* denotes a weighting expression, and *options* denotes a list of options.

18.1.1 varlist

Most commands that take a subsequent *varlist* do not require that one be explicitly typed. If no *varlist* appears, these commands assume a *varlist* of _all, the Stata shorthand for indicating all the variables in the dataset. In commands that alter or destroy data, Stata requires that the *varlist* be specified explicitly. See [U] **18.4 varlists** for a complete description.

▷ Example

The summarize command lists the mean, standard deviation, and range of the variables specified. In [R] **summarize**, we see that the syntax diagram for summarize is

[by *varlist*:] summarize [*varlist*] [*weight*] [if *exp*] [in *range*] [, detail format]

Since everything but the word summarize is enclosed in square brackets, the simplest form of the summarize command is summarize. Typing summarize without arguments is equivalent to typing summarize _all; all the variables in the dataset are summarized. Underlining denotes the shortest allowed abbreviation: we could have typed just su; see [U] **18.2 Abbreviation rules**.

The drop command eliminates variables or observations from a dataset. The syntax diagram for the version that drops variables is

drop *varlist*

Typing drop by itself results in the error message "varlist or in range required". To drop all the variables in the dataset, you must type drop _all.

Even before looking at the syntax diagram, we could have predicted that the *varlist* would be required—drop is destructive, and hence we are required to spell out our intent. The syntax diagram informs us that the *varlist* is required since *varlist* is not enclosed in square brackets. Since drop is not underlined, it cannot be abbreviated.

◁

18.1.2 by varlist:

The by *varlist*: prefix causes Stata to repeat a command for each subset of the data for which the values of the variables in the *varlist* are equal. When prefixed with by *varlist*:, the result of the command will be the same as if you had formed separate datasets for each group of observations, saved them, and then given the command on each dataset separately.

▷ Example

Typing summarize mrg dvc produces a table of the mean, standard deviation, and range of mrg and dvc using all the observations in the data:

```
. summarize mrg dvc
Variable |  Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |   50      0.0187    0.0257     0.0104     0.1955
     dvc |   50      0.0080    0.0032     0.0040     0.0237
```

Typing by region: summarize mrg dvc produces a set of tables, one table for each region of the country:

```
. by region: summarize mrg dvc
-> region=N. Central
Variable |  Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |   12      0.0139    0.0016     0.0123     0.0181
     dvc |   12      0.0066    0.0016     0.0046     0.0103
-> region=N. East
Variable |  Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |    9      0.0121    0.0018     0.0104     0.0150
     dvc |    9      0.0058    0.0015     0.0040     0.0079
```

```
-> region=South
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |    16      0.0162      0.0037     0.0104     0.0247
     dvc |    16      0.0079      0.0018     0.0054     0.0112
-> region=West
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |    13      0.0307      0.0496     0.0120     0.1955
     dvc |    13      0.0108      0.0043     0.0064     0.0237
```

Note that by *varlist*: can be used with almost any Stata command. For instance, we could now obtain the correlations, by region, between `mrg` and `dvc` by typing `by region: correlate mrg dvc`.

◁

❑ Technical Note

The *varlist* in `by` *varlist*: may contain up to ten variables. For instance, if you had data on automobiles and wished to obtain means according to market category (`market`) broken down by manufacturer (`origin`), you could type `by market origin: summarize`. That *varlist* contains two variables: `market` and `origin`.

❑

❑ Technical Note

The *varlist* in `by` *varlist*: may contain either string or numeric variables, or both. In the example above, `region` is a string variable, in particular, a `str10`. The example would have worked, however, if `region` were a numeric variable with values 1, 2, 3, and 4 or even 12.2, 16.78, 32.417, and 15.213.

❑

18.1.3 if exp

The `if` *exp* qualifier restricts the scope of a command to those observations for which the value of the expression is *true* (which is equivalent to the expression being nonzero; see [U] **20 Functions and expressions**).

› Example

Typing `summarize mrg dvc if region=="West"` produces a table for the western region of the country:

```
. summarize mrg dvc if region=="West"
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |    13      0.0307      0.0496     0.0120     0.1955
     dvc |    13      0.0108      0.0043     0.0064     0.0237
```

The double equal sign in `region=="West"` is not an error. Stata uses *double* equal signs to denote equality testing and a *single* equal sign to denote assignment; see [U] **20 Functions and expressions**.

A command may have at most one `if` qualifier. If you want the summary for the West restricted to observations with values of `mrg` in excess of 0.02, do *not* type `summarize mrg dvc if region=="West" if mrg>.02`. Type

```
. summarize mrg dvc if region=="West" & mrg>.02
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |     3      0.0793      0.1007     0.0211     0.1955
     dvc |     3      0.0155      0.0072     0.0104     0.0237
```

You may not use the word 'and' in place of the symbol '`&`' to join conditions. To select observations that meet one condition *or* another, you use the '`|`' symbol. For instance, `summarize mrg dvc if region=="West" | mrg>.02` summarizes all observations for which `region` is West *or* `mrg` is greater than 0.02.

◁

▷ Example

`if` may be combined with `by`. Typing `by region: summarize mrg dvc if mrg>.02` produces a set of tables, one for each region, reflecting summary statistics on `mrg` and `dvc` among observations for which `mrg` exceeds 0.02:

```
. by region: summarize mrg dvc if mrg>.02
-> region=N. Central
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |     0
     dvc |     0
-> region=N. East
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |     0
     dvc |     0
-> region=South
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |     2      0.0231      0.0023     0.0214     0.0247
     dvc |     2      0.0087      0.0035     0.0062     0.0112
-> region=West
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |     3      0.0793      0.1007     0.0211     0.1955
     dvc |     3      0.0155      0.0072     0.0104     0.0237
```

The results indicate that there are no states in the Northeast and North Central regions for which `mrg` exceeds 0.02, while there are two such states in the South and three states in the West.

◁

18.1.4 in range

The `in` *range* qualifier restricts the scope of the command to a specific observation range. A range specification takes the form $\#_1[/\#_2]$, where $\#_1$ and $\#_2$ are positive or negative. Negative integers are understood to mean "from the end of the data", with -1 referring to the last observation. The implied first observation must be less than or equal to the implied last observation.

The first and last observations in the dataset may be denoted by `f` and `l` (letter *el*) respectively. A range specifies absolute observation numbers within a dataset. As a result, the `in` qualifier may not be used when the command is preceded by the by *varlist*`:` prefix.

▷ Example

Typing `summarize mrg dvc in 5/25` produces a table based on the values of `mrg` and `dvc` in observations 5 through 25:

```
. summarize mrg dvc in 5/25
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |    21      0.0136      0.0033      0.0104     0.0247
     dvc |    21      0.0064      0.0016      0.0040     0.0098
```

This is, admittedly, a rather odd thing to want to do. It would not be odd, however, if we substituted `list` for `summarize`. If we wanted to see the states with the 10 lowest values of `mrg`, we could type `sort mrg` followed by `list mrg in 1/10`.

Typing `summarize mrg dvc in f/l` is equivalent to typing `summarize mrg dvc`—all observations are summarized.

◁

▷ Example

Typing `summarize mrg dvc in 5/25 if region=="South"` produces a table based on the values of the two variables in observations 5 through 25 for which the value of `region` is South:

```
. summarize mrg dvc in 5/25 if region=="South"
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
     mrg |     4      0.0175      0.0056      0.0111     0.0247
     dvc |     4      0.0080      0.0018      0.0062     0.0098
```

The ordering of the `in` and `if` qualifiers is not significant. The command could also have been specified as `summarize mrg dvc if region=="South" in 5/25`.

◁

▷ Example

Negative ranges can be most usefully employed with `sort`. For instance, we have data on automobiles and wish to list the five with the highest mileage rating:

```
. sort mpg
. list make mpg in -5/l
                make      mpg
70.     Toyota Corolla       31
71.       Plym. Champ       34
72.             Subaru      35
73.        Datsun 210       35
74.         VW Diesel       41
```

◁

18.1.5 =exp

The *=exp* specifies the value to be assigned to a variable and is most often used with `generate` and `replace`. See [U] **20 Functions and expressions** for details on expressions and [R] **generate** for details on the `generate` and `replace` commands.

▷ Example

Expression	Meaning
`generate newvar=oldvar+2`	creates a new variable named `newvar` equal to `oldvar+2`
`replace oldvar=oldvar+2`	changes the contents of the existing variable `oldvar`
`egen newvar=rank(oldvar)`	creates `newvar` containing the ranks of `oldvar` (see [R] **egen**)

◁

18.1.6 weight

weight indicates the weight to be attached to each observation. The syntax of *weight* is

[*weightword=exp*]

where you actually type the square brackets and where *weightword* is one of

weightword	Meaning
`weight`	default treatment of weights
`aweight` or `cellsize`	analytic weights
`fweight` or `frequency`	frequency weights
`pweight`	sampling weights
`iweight`	importance weights

The underlining indicates the minimum acceptable abbreviation. Thus, `weight` may be abbreviated `w`, or `we`, etc.

▷ Example

Before explaining what the different types of weights mean, let's obtain the population-weighted mean of a variable called **medage** from data containing observations on 50 states of the U.S. The data also contains a variable named pop which is the total population of each state.

```
. summarize medage [weight=pop]
(analytic weights assumed)
Variable |     Obs      Weight      Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------------------
  medage |      50   225907472     30.11        1.67     24.20      34.70
```

In addition to telling us that our data contains 50 observations, we are informed that the sum of the weight is 225,907,472, which was the number of people living in the U.S. as of the 1980 census. The weighted mean is 30.11. We were also informed that Stata assumed we wanted "analytic" weights. ◁

Stata understands four kinds of weights:

1. `fweights`, or frequency weights, indicate duplicated observations. `fweights` are always integers. If the `fweight` associated with an observation is 5, that means there are really 5 such observations, each identical.
2. `aweights`, or analytic weights, are automatically scaled to sum to N, the number of observations. These kinds of weights are also called regression weights. Because of the automatic rescaling, the original scale of the weights is irrelevant.
3. `pweights`, or sampling weights, denote the inverse of the probability that this observation is included in the sample due to the sampling strategy. A `pweight` of 100, for instance, indicates that this observation is representative of 100 subjects in the underlying population. The scale of these weights does not matter in terms of estimated parameters and standard errors, except for estimating totals.
4. `iweights`, or importance weights, indicate the relative "importance" of the observation.

See [U] **26.12 Weighted estimation** for a thorough discussion of weights and their meaning.

`weight` is each command's idea of what are the "natural" weights and is one of `aweight`, `fweight`, `pweight`, or `iweight`. When you specify the vague `weight`, the command informs you which kind it assumes.

❑ Technical Note

When you do not specify a weight, the result is equivalent to specifying `[fweight=1]`. The emphasis is on equivalent, since Stata may go to more work when you specify a weight. ❑

18.1.7 options

Many commands take command-specific options. These are described along with each command in the *Reference Manual*. Options are indicated by typing a comma at the end of the command, followed by a list of options.

▷ Example

Typing `summarize mrg` produces a table of the mean, standard deviation, minimum, and maximum of the variable `mrg`:

```
. summarize mrg
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
     mrg |      50      .0187    .0257      .0104      .1955
```

The syntax diagram for `summarize` is

[by *varlist*:] summarize [*varlist*] [*weight*] [if *exp*] [in *range*] [, detail format]

Thus, the options allowed by `summarize` are `detail` and `format`. The shortest allowed abbreviations for `detail` and `format` are `d` and `f`, respectively; see [U] **18.2 Abbreviation rules**.

Typing `summarize mrg, detail`, produces a table that also includes selected percentiles, the four largest and four smallest values, the skewness, and the kurtosis.

```
. summarize mrg, detail
                            mrg
-------------------------------------------------------------
      Percentiles      Smallest
 1%       .0104           .0104
 5%       .0106           .0104
10%       .0110           .0106       Obs                  50
25%       .0127           .0107       Sum of Wgt.          50

50%       .0150                       Mean                .0187
                          Largest      Std. Dev.           .0257
75%       .0179           .0212
90%       .0204           .0214       Variance            .0007
95%       .0214           .0247       Skewness           6.6853
99%       .1955           .1955       Kurtosis          46.4750
```

◁

❑ Technical Note

Once you have typed the *varlist* for the command, options may be placed anywhere in the command. You can type `summarize mrg dvc if region=="West", detail` or you can type `summarize mrg dvc, detail, if region=="West"`. Note the use of a second comma to indicate return to the command line as opposed to the option list. Leaving out the comma after the word `detail` would cause an error, since Stata would attempt to interpret the phrase `if region=="West"` as an option rather than as part of the command.

You may not type an option in the middle of a *varlist*. Typing `summarize mrg, detail, dvc` will result in an error.

Options need not be contiguously specified. You may type `summarize mrg dvc, detail, if region=="South", noformat`. Both `detail` and `noformat` are options.

❑

❑ Technical Note

Most of the command options are toggles—they indicate that something either is or is not to be done. Sometimes it is difficult to remember which is the default. The following rule applies to all options: If *option* is an option, then `no`*option* is an option as well, and vice versa. Thus, if we could not remember whether `detail` or `nodetail` were the default for `summarize` but we knew that we did not want the detail, we could type `summarize, nodetail`. Typing the `nodetail` option is unnecessary, but Stata will not complain.

Some options take *arguments*. The Stata `kdensity` command has a `n(#)` option that indicates the number of points at which the density estimate is to be evaluated. When an option takes an argument, it is always enclosed in parentheses.

Some options take more than one argument. The `graph` command has a `ylabel` option in which you specify the values to be labeled on the *y*-axis. When an option takes more than one argument, the arguments are separated from each other by *commas*. To label the *y*-axis with the values 1, 2, 3, and 4, you would type `ylabel(1,2,3,4)`. You may place spaces as well as commas between the arguments, but you must not omit the commas.

The `graph` command's `ylabel` option is also an example of an option in which the arguments are optional. `ylabel` does not require arguments and, if you do not specify any, Stata will automatically choose the values to be labeled. The option `noylabel` exists as well; it indicates that the axis is not to be labeled.

Some options take string arguments. The `graph` command's `title` option works this way—for instance, `title("Figure 1.")`. To play it safe, you should type the quotes surrounding the string, although it is not required. If you do not type the quotes, any sequence of two or more consecutive blanks will be interpreted as a single blank. Thus, `title(Figure  1.)` would be interpreted the same as `title(Figure 1.)`.

❑

18.2 Abbreviation rules

Stata allows abbreviations. In this manual we typically avoid abbreviating commands, variable names, and options to ensure readability:

```
. summarize myvar, detail
```

Real Stata users, on the other hand, tend to abbreviate:

```
. sum myv, d
```

As a general rule, command, option, and variable names may be abbreviated to the shortest string of characters that uniquely identifies them.

This rule is violated if the command or option does something that cannot easily be undone: in that case, the command must be spelled out in its entirety.

In addition, a few common commands and options are allowed to have even shorter abbreviations than the general rule would allow.

The general rule is applied, without exception, to variable names.

18.2.1 Command abbreviation

The shortest allowed abbreviation for a command or option can be determined by looking at the command's syntax diagram. This minimal abbreviation is shown by underlining:

```
regress
rename
replace
rotate
run
```

Lack of underlining means no abbreviation is allowed. Thus, `rename` and `replace` may not be abbreviated, the underlying reason being that these commands change the data.

`regress` can be abbreviated `reg`, `regr`, `regre`, `regres`, or can be spelled out in its entirety.

In the on-line help, highlighting or color is used in place of underlining in the syntax diagram.

As mentioned above, sometimes very short abbreviations are also allowed. Commands that begin with the letter *d* include `decode`, `describe`, `dir`, `discard`, `display`, `do`, and `drop`. This suggests that the shortest allowable abbreviation for `describe` is `des`. Since `describe` is such a commonly used command, you may abbreviate it with the single letter `d`. You may also abbreviate the `list` command with the single letter `l`.

The other exception to the general abbreviation rule concerns commands that alter or destroy data; such commands must be spelled out completely. Two of the commands that begin with the letter *d* above, `discard` and `drop`, are destructive in the sense that once you give one of these commands there is no way you can undo the result; therefore, both must be spelled out.

Another exception to the abbreviation rule occurs when you type `help` *command*. The *command* may not be abbreviated. For instance, although you may type `su` as an abbreviation for `summarize`, you must type `help summarize` to see the on-line help. Typing `help su` will result in the message "help for su not found".

The final exceptions to the general rule are commands implemented as ado-files. Such commands may not be abbreviated. Ado-file commands, while appearing to be part of Stata, are external and their names correspond to the names of disk files.

18.2.2 Option abbreviation

Option abbreviation follows the same logic as command abbreviation: you determine the minimum acceptable abbreviation by examining the command's syntax diagram. The syntax diagram for `summarize` reads

[`by` *varlist*`:`] `summarize` ... , `detail` `format`

Option `detail` may be abbreviated `d`, `de`, `det`, ..., `detail`. Similarly, option `format` may be abbreviated `f`, `fo`, ..., `format`.

Options `clear` and `replace` occur with many commands. The `clear` option indicates that even though completion of this command will result in the loss of all data in memory, and even though the data in memory has changed since it was last saved on disk, you are aware of the situation and it is okay to continue. `clear` must be spelled out, as in `use newdata, clear`.

The `replace` option indicates that it's okay to save over an existing data set. If you type `save mydata` and the file `mydata.dta` already exists, you will receive the message "file mydata.dta already exists" and Stata will refuse to overwrite it. To allow Stata to overwrite the dataset, you type `save mydata, replace`. `replace` may not be abbreviated.

❑ Technical Note

`replace` is a stronger modifier than `clear` and one you should think about longer before specifying. With a mistaken `clear`, you can lose perhaps hours of work. With a mistaken `replace`, however, you can lose days of work.

❑

18.2.3 Variable-name abbreviation

Variable names may be abbreviated to the shortest string of characters that uniquely identifies them given the data currently loaded in memory.

If your dataset contained four variables, `state`, `mrgrate`, `dvcrate`, and `dthrate`, you could refer to the variable `dvcrate` as `dvcrat`, `dvcra`, `dvcr`, `dvc`, or `dv`. You might type `list dv` to list the data on `dvcrate`. You could not refer to the variable `dvcrate` as `d`, however, since that abbreviation does not distinguish `dvcrate` from `dthrate`. If you were to type `list d`, Stata would respond with the message "ambiguous abbreviation". (If you wanted to refer to *all* variables that started with the letter *d*, you could type `list d*`; see [U] **18.4 varlists**.)

18.3 Naming conventions

A name is a sequence of one to eight letters (`A`–`Z` and `a`–`z`), digits (`0`–`9`), and underscores (`_`).

Programmers: local macro names can have no more than 7 characters in the name; see [U] **24.3.1 Local macros**.

Stata reserves the following names:

`_all`	`double`	`long`	`_rc`
`_b`	`float`	`_n`	`_se`
`byte`	`if`	`_N`	`_skip`
`_coef`	`in`	`_pi`	`using`
`_cons`	`int`	`_pred`	`with`

You may not use these reserved names for your variables.

The first character of a name must be either a letter or an underscore. We recommend, however, that you do not begin your variable names with an underscore. All Stata built-in variables begin with an underscore, and we reserve the right to incorporate new *_variables* freely.

Stata respects case; that is, `myvar`, `Myvar`, and `MYVAR` are three distinct names.

All objects in Stata—not just variables—follow this naming convention.

18.4 varlists

A *varlist* is a list of variable names. The variable names in a *varlist* refer either exclusively to new (not yet created) variables or exclusively to existing variables.

18.4.1 Lists of existing variables

In lists of existing variable names, variable names may be repeated.

▷ Example

If you type `list state mrgrate dvcrate state` the variable `state` will be listed twice, once in the leftmost column and again in the rightmost column of the list.

◁

Existing variable names may be abbreviated as described in [U] **18.2 Abbreviation rules**. You may also append a '`*`' to a partial variable name to indicate all variables that start with that letter combination.

▷ Example

If the variables `poplt5`, `pop5to17`, and `pop18p` are in your dataset, you may type `pop*` as a shorthand way to refer to all three variables. For instance, `list state pop*` lists the variables `state`, `poplt5`, `pop5to17`, and `pop18p`.

◁

You may place a dash (`-`) between two variable names to specify all the variables stored between the two listed variables, inclusive. You can determine storage order using `describe`; it lists variables in the order in which they are stored.

▷ Example

If your data contains the variables `state`, `mrgrate`, `dvcrate`, and `dthrate`, in that order, typing `list state-dvcrate` is equivalent to typing `list state mrgrate dvcrate`. In both cases, three variables are listed.

◁

18.4.2 Lists of new variables

In lists of *new variables*, no variable names may be repeated or abbreviated.

You may specify a dash (`-`) between two variable names that have the same letter prefix and that end in numbers. This form of the dash notation indicates a range of variable names in ascending numerical order.

▷ Example

Typing `input v1-v4` is equivalent to typing `input v1 v2 v3 v4`. Typing `infile state v1-v3 ssn using rawdata` is equivalent to typing `infile state v1 v2 v3 ssn using rawdata`.

◁

You may specify the storage type before the variable name to force a storage type other than the default. The numeric storage types are `byte`, `int`, `long`, `float` (the default), and `double`. The string storage types are `str#`, where # is replaced with an integer between 1 and 80, inclusive, representing the maximum length of the string. See [U] **19 Data**.

For instance, the list `var1 str8 var2 var3` specifies that `var1` and `var3` are to be given the default storage type, whereas `var2` is to be stored as a `str8`—a string whose maximum length is eight characters.

The list `var1 int var2 var3` specifies that `var2` is to be stored as an `int`. You may use parentheses to bind a list of variable names. The list `var1 int(var2 var3)` specifies that both `var2` and `var3` are to be stored as `int`s. Similarly, the list `var1 str20(var2 var3)` specifies that both `var2` and `var3` are to be stored as `str20`s. The different storage types are listed in [U] **19.2.2 Numeric storage types** and [U] **19.4.4 String storage types**.

▷ Example

Typing `infile str2 state str10 region v1-v5 using mydata` reads the `state` and `region` strings from the file `mydata.raw` and stores them as `str2` and `str10`, respectively, along with the variables `v1` through `v5`, which are stored with the default storage type `float` (unless you have specified a different default with the `set type` command).

Typing `infile str10(state region) v1-v5 using mydata` would achieve almost the same result, except that the `state` and `region` values recorded in the data would both be assigned to `str10` variables. (You could then use the `compress` command to shorten the strings. See [R] **compress**; it is well worth reading.)

◁

❑ Technical Note

You may append a colon and a *value label name* to numeric variables. (See [U] **19.6 Dataset, variable, and value labels** for a description of value labels.) For instance, `var1 var2:myfmt` specifies that the variable `var2` is to be associated with the value label stored under the name `myfmt`. This has the same effect as typing the list `var1 var2` and then subsequently giving the command `label values var2 myfmt`.

The advantage of specifying the value label association with the colon notation is that value labels can then be assigned by the current command; see [R] **input** and [R] **infile (free format)**.

❑

▷ Example

Typing `infile int(state:stfmt region:regfmt) v1-v5 using mydata, automatic` reads the state and region data from the file `mydata.raw` and stores them as `int`s, along with the variables `v1` through `v5`, which are stored with the default storage type.

In our previous example, both state and region were strings, so how can strings be stored in a numeric variable? See [U] **19.6 Dataset, variable, and value labels** for the complete answer. The colon notation specifies the name of the value label and the `automatic` option tells Stata to assign unique numeric codes to all character strings. The numeric code for state, which Stata will make up on the fly, will be stored in the `state` variable. The mapping from numeric codes to words will be stored in the value label named `stfmt`. Similarly, region will be assigned numeric codes which are stored in `region` and the mapping will be stored in `regfmt`.

If you were to `list` the data, the `state` and `region` variables would look like strings. `state`, for instance, would appear to contain things like `AL`, `CA`, and `WA`, but actually it contains only numbers like 1, 2, 3, and 4.

◁

18.5 by varlist: construct

by *varlist*: *command*

The `by` prefix causes *command* to be repeated for each unique set of values of the variables in the *varlist*. The data must already be `sorted` by the *varlist*. *varlist* may contain numeric, string, or a mixture of numeric and string variables.

`by` is an optional prefix to perform a Stata command separately for each group of observations where the values of the variables in the *varlist* are the same.

During each iteration, the values of the system variables `_n` and `_N` are set in relation to the first observation in the by-group; see [U] **20.7 Explicit subscripting**. The `in` *range* qualifier cannot be used in conjunction with `by` *varlist*: because ranges specify absolute rather than relative observation numbers.

❑ Technical Note

The inability to combine `in` and `by` is not really a constraint, since `if` provides all the functionality of `in` and quite a bit more besides. If you wanted to perform *command* for the first three observations in each of the by-groups, you could type

```
. by varlist: command if _n<=3
```

❑

The results of *command* will be the same as if you had formed separate datasets for each group of observations, `saved` them, `used` each separately, and issued *command*.

▷ Example

We provide some examples using `by` in [U] **18.1.2 by varlist:** above. We demonstrate the effect of `by` on `_n`, `_N`, and explicit subscripting in [U] **20.7 Explicit subscripting**.

`by` requires that the data first be sorted. For instance, if we had data on the average January and July temperatures in degrees Fahrenheit for 420 cities located in the Northeast and West and wanted to obtain the averages, `by region`, across those cities, we might type

```
. by region: summarize tempjan tempjuly
not sorted
r(5);
```

Stata refused to honor our request since the data is not sorted by `region`. We must `sort` the data first; see [R] **sort**.

```
. sort region
. by region: summarize tempjan tempjuly
-> region=Northeast
Variable |  Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------
 tempjan |  164       27.89        3.54       16.60      31.80
tempjuly |  164       73.35        2.36       66.50      76.80
-> region=West
Variable |  Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------
 tempjan |  256       46.23       11.25       13.00      72.60
tempjuly |  256       72.11        6.48       58.10      93.60
```

◁

❑ Technical Note

Using the same data as in the example above, we estimate regressions, by `region`, of average January temperature on average July temperature. Both temperatures are specified in degrees Fahrenheit.

```
. by region:  regress tempjan tempjuly
-> region=      NE
  Source |       SS       df       MS              Number of obs =     164
---------+------------------------------           F(  1,   162) =  479.82
   Model |  1529.74026     1  1529.74026           Prob > F      =  0.0000
Residual |  516.484453   162  3.18817564           R-squared     =  0.7476
---------+------------------------------           Adj R-squared =  0.7460
   Total |  2046.22471   163  12.5535258           Root MSE      =  1.7855

------------------------------------------------------------------------------
 tempjan |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
---------+--------------------------------------------------------------------
tempjuly |   1.297424   .0592303    21.905   0.000     1.180461    1.414387
   _cons |  -67.28066   4.346781   -15.478   0.000    -75.86431     -58.697
------------------------------------------------------------------------------

-> region=    West
  Source |       SS       df       MS              Number of obs =     256
---------+------------------------------           F(  1,   254) =    2.84
   Model |  357.161728     1  357.161728           Prob > F      =  0.0932
Residual |  31939.9031   254  125.74765           R-squared     =  0.0110
---------+------------------------------           Adj R-squared =  0.0072
   Total |  32297.0648   255  126.655156           Root MSE      =  11.214

------------------------------------------------------------------------------
 tempjan |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
---------+--------------------------------------------------------------------
tempjuly |   .1825482   .1083166     1.685   0.093    -.0307648    .3958613
   _cons |   33.0621    7.84194      4.216   0.000     17.61859     48.5056
------------------------------------------------------------------------------
```

The regressions show that a one-degree increase in the average July temperature in the Northeast corresponds to a 1.3 degree increase in the average January temperature. In the West, however, it corresponds to a 0.18 degree increase, which is only marginally significant.

❑

18.6 File-naming conventions

Some commands require that you specify a *filename*. Filenames are specified in the way natural for your operating system:

DOS and Windows	Unix	Macintosh
`mydata`	`mydata`	`mydata`
`mydata.dta`	`mydata.dta`	`mydata.dta`
`b:mydata.dta`	`~friend/mydata.dta`	`"My Data.dta"`
`myproj\mydata`	`myproj/mydata`	`myproject:mydata`
`\analysis\data\mydata`	`~/analysis/data/mydata`	`"My Project:My Data"`
`c:\analysis\data\mydata`	`~/analysis/data/mydata`	`"~:My Project:My Data"`
`..\data\mydata`	`../data/mydata`	`":..:My Project:My Data"`

In most cases (the exceptions being `dir`, `ls`, `erase`, `rm`, and `type`), Stata automatically provides a file extension if you do not supply one. For instance, if you type `use mydata`, Stata assumes you mean `use mydata.dta`, since `.dta` is the file extension Stata normally assumes for data.

Stata provides eight default file extensions that are used by various commands. They are

`.ado`	automatically loaded do-files
`.dct`	ASCII data dictionary
`.do`	do-file
`.dta`	Stata-format dataset
`.gph`	graph image
`.log`	log file
`.out`	file saved by `outsheet`
`.raw`	ASCII-format dataset

You do not have to name your data files with the `.dta` extension—if you type an explicit file extension, it will override the default. For instance, if your data was stored as `myfile.dat`, you could type `use myfile.dat`. If your data was stored as simply `myfile` with no file extension, type the period at the end of the filename to indicate that you are explicitly specifying the null extension. You type `use myfile.` to use this data.

All operating systems except DOS and Windows 3.1 allow blanks in filenames and so does Stata. However, if the filename includes a blank, you must enclose the filename in double quotes:

```
. save "my data"
```

would create the file `my data.dta`. Typing

```
. save my data
```

would be an error.

18.6.1 A special note for Macintosh users

Have you seen the notation `myfolder:myfile` before? It refers to the file `myfile` in the folder `myfolder`, which folder is contained in the current folder.

You do not have to use this notation if you do not like it. You could instead restrict yourself to using files only in the current directory. If that turns out to be too restricting, Stata for Macintosh provides enough pulldowns that you can probably get by. You may, however, find the notation convenient. In case you do, here is the rest of the definition.

`..:myfile` refers to `myfile` in the folder containing the current folder.

Thus, `..:nextdoor:myfile` refers to `myfile` in the folder `nextdoor` in the folder containing the current folder.

The notation `~:mydata:myfile` refers to `myfile` in the folder `mydata`, which is located in the folder containing the Stata folder.

Thus, `~:Stata:auto.dta` refers to the automobile data in the Stata folder.

18.6.2 A special note for Unix users

Stata understands ~ to mean your home directory. Stata understands this even if you do not use csh(1) as your shell.