Stata 12 **Tutorial 1**

TOPIC: **Getting Started with *Stata*: An Introduction or Review**

DATA: **auto1.raw** and **auto1.txt** (two *text-format* data files)

TASKS: ***Stata 12 Tutorial 1*** is intended to introduce you to some of the basic features and commands of *Stata* that you will need to do subsequent tutorials and your research assignments. It begins by showing you how to create and document a *Stata*-format dataset from a text-format data file. It then illustrates several *Stata* commands for describing, summarizing and tabulating data, and for creating new variables from existing variables. The *Stata* commands included in this review should work in both *Stata 11* (*Stata Release 11*) and *Stata 12* (*Stata Release 12*).

Topics introduced in *Tutorial 1* include:

- Accessing and exploring the ECON 452 web site
- Downloading a text-format data file from the ECON 452 web site
- Starting and recording a *Stata* session using the **log using** and related commands
- Inputting a text-format data file into *Stata* using the **infile** and **insheet** commands
- Displaying and summarizing data in *Stata* using the **describe**, **list**, **summarize** and **codebook** commands
- Creating a *Stata*-format dataset using the **save** command
- Removing the current dataset in memory using the **clear** command
- Loading a *Stata*-format dataset into memory using the **use** command
- Labeling variables and *Stata*-format datasets using the **label variable** and **label data** commands
- Creating new variables in *Stata* using the **generate** and **replace** commands
- Producing one-way and two-way frequency tables for categorical variables using the **tabulate**, **tab1** and **tab2** commands
- Producing one-way tables of statistics for the various categories (values) of categorical variables using the **table** command
- Ending a *Stata* session using the **exit** command

- The ***Stata*** **commands** introduced in this tutorial are:

log using	Opens and names a **log file** to record your *Stata* session.
log close	Closes the **log file** of your *Stata* session.
logoff	Temporarily stops recording your *Stata* session to the currently open log file.
logon	Resumes recording your *Stata* session to the currently open log file.
infile	Loads an unformatted text-format data file into memory.
insheet	Loads certain text-format data files into memory
cd	Changes the *Stata* working directory
dir	Lists the contents of the current *Stata* working directory
describe	Displays a summary of the current dataset in memory.
list	Displays values of some or all variables in the current dataset.
summarize	Displays descriptive summary statistics for numeric variables.
save	Saves to disk the data currently in memory as a new *Stata*-format dataset and gives the *Stata*-format dataset a name.
save, replace	Saves or resaves to disk under the same name the *Stata*-format dataset currently in memory.
clear	Removes or clears the current dataset from memory.
use	Loads or reads into memory an existing *Stata*-format dataset that currently resides on disk.
label variable	Assigns a brief user-supplied descriptive label to a variable.
label data	Assigns a short descriptive label to a *Stata*-format dataset.
codebook	Displays the characteristics, or properties, of any variable(s) in the current data set.
generate	Creates new variables from expressions containing existing variables, operators, and functions.
replace	Used to modify the contents (values) of existing variables.
compare	Compares two variables; reports the differences and similarities between two variables.
cf	Compares two datasets
drop	Eliminates or deletes variables or observations from the dataset in memory.
correlate	Displays correlation matrix for two or more numeric variables.
tabulate	Produces one-way and two-way frequency tables for categorical variables.

table	Produces one-way tables of statistics for the various categories (values) of categorical variables.
exit	Ends a *Stata* session.

NOTE: *Stata* commands are *case sensitive*. All *Stata command names* must be typed in the Command window in ***lower case*** **letters**.

HELP: *Stata* has an extensive on-line **Help** facility that provides fairly detailed information (including examples) on all *Stata* commands. Students should become familiar with the *Stata* on-line **Help** system. In the course of doing this tutorial, take the time to browse the **Help** information on some of the above *Stata* commands. To access the on-line **Help** for any *Stata* command:

- choose (click on) **Help** from the *Stata* main menu bar
- click on **Stata Command…** in the **Help** drop down menu
- type the full name of the *Stata* command in the *Stata* command dialog box and click **OK**

Introduction to the ECON 452 Web Site

This first part of the tutorial introduces you to the ECON 452 web site. You will need to use this web site for several purposes: to obtain the data files required for tutorials and assignments; to learn when assignments are due; to get another copy of the course outline and reading list; and to find out about the *Stata* tutorial schedule.

❑ Accessing and Browsing the ECON 452* Web Site

Three widely used web browsers are **Netscape**, **Microsoft Internet Explorer** and **Mozilla Firefox**. **Netscape** is the browser installed on many computers located on the Queen's campus.

- **On the computers in Dunning 350**, **Mozilla Firefox** is the installed web browser. (**Microsoft Internet Explorer** is also installed on the PCs in Dunning 350.)

This section outlines how to use the **Firefox** web browser to locate and browse the ECON 452 web site. But the **Netscape** and **Internet Explorer** browsers work similarly.

1. **Start your web browser** by ***double-clicking*** the **Firefox icon** on the Windows XP desktop.

2. **Proceed to the homepage of the ECON 452 web site.**

   How you find your way to the ECON 452 homepage depends on where you start from when beginning a browser session.

   - **On the computers in Dunning 350**, the **Firefox** and **Internet Explorer** web browsers begin at the QED (Queen's Economics Department) homepage, the URL for which is **http://www.econ.queensu.ca**.

   - The **most direct way to get to the ECON 452 homepage** from wherever your browser starts is to type or copy in your browser's web address window the address (URL) of the ECON 452 web site, which is:

http://www.econ.queensu.ca/pub/faculty/abbott/econ452/

Once you have typed or copied the above address into the browser's web address window, press the Enter key on the keypad.

3. **Browse the links listed on the homepage of the ECON 452 Web Site.**

Now that you are at the ECON 452 homepage, take a few minutes to explore the various pages of the web site. To do this, simply click on the links listed on the ECON 452 homepage.

❑ Downloading Data Files from the ECON 452 Web Site

You will frequently need to obtain data files for the tutorials and assignments in ECON 452. These data files can all be downloaded to the computer you are working at from the ECON 452 web site. This section explains how to download a text-format data file using either the **Firefox** or **Internet Explorer** web browser.

What is a **text-format (or ASCII-format) data file**? A text-format file is one that contains no special characters or formatting. It can be viewed and edited with a text editor (such as Windows Notepad or Windows Wordpad) or with any document processor (such as MS-Word or Corel WordPerfect).

Each of the text-format data files **auto1.raw** and **auto1.txt** contains the raw data you will use in this tutorial. The data file **auto1.raw** can be downloaded, or copied, from the ECON 452 web site by proceeding through the following steps.

1. Start your web browser, and **make your way to the ECON 452 homepage**, the address for which is

   http://www.econ.queensu.ca/pub/faculty/abbott/econ452/.

2. Under the heading **Information on STATA** on the ECON 452 homepage, click on **Download Text-Format Dataset 'auto1.raw'**.

   In the **Firefox** web browser, clicking on **Download Text-Format Dataset 'auto1.raw'** displays on the screen the contents of the text-format data file

auto1.raw. Use the vertical scroll bar to browse the contents of data file **auto1.raw**.

In the **Internet Explorer** web browser, clicking on **Download Text-Format Dataset 'auto1.raw'** opens the **File Download** dialog box. Click the **Open** button to display the file contents on the screen; you can then use the vertical scroll bar to browse the contents of data file **auto1.raw**. Alternatively, click the **Save** button to open the **Save As** dialog box and proceed directly to Step 4 below.

3. To save the text-format data file **auto1.raw** to disk, click on the **File** menu at the top of your browser screen. From the **File** menu select (click on) either the **Save Page As…** or **Save As…** item. The **Save As** dialog box appears.

4. In the **Save As** dialog box, select the drive and/or directory to which you want to save the data file.

- ***To download the data file to a flash memory stick in the E:-drive***, click on the down arrow beside the **Save in:** window; then click on the **E:** icon.

- ***To download the data file to a directory on the hard drive***, for example, **C:\data**, click on the **C:** icon in the **Save in:** window. Next, scroll through the folder list and double-click on the folder to which you want to copy the data file, for example the **data** folder.

5. Click on the **Save** button in the **Save As** dialog box. This copies the data file **auto1.raw** to the directory and drive selected in step 4 above. Close the **Save As** dialog box.

6. Click the **Back** button (←) in the **Firefox** button bar to return to the ECON 452 homepage.

7. Follow essentially the same procedure you used to download the data file **auto1.raw** to download the text-format data file **auto1.txt**, specifically steps 2-6 above.

❑ Preparing for Your *Stata* Session

Before you start your Stata session, make sure that you have downloaded the data files **auto1.raw** and **auto1.txt** and know where on the hard drive of your PC you have placed them. Once you have started your *Stata* session, you will want to change the ***Stata working directory*** to the folder in which you have placed your data files.

What is the default *Stata* working directory?

- **On the computers in Dunning 350**, the default *Stata* working directory is usually **C:\data**.

  When working in Dunning 350, you may download your data files to this directory/folder, or if you prefer to some other directory such as **C:\courses**. The important thing is that you keep track of the directory in which you placed the downloaded data files; this is the directory in which you will probably find it most convenient to work during your *Stata* session.

- **On the computers in MC B111**, the default *Stata* working directory is usually **D:\courses**.

❑ Starting a *Stata* Session

There are two ways to start a *Stata* session.

- If you see a ***Stata 11*** **or** ***Stata 12*** **icon** on the Windows desktop, simply ***double-click*** it.

- If there is no *Stata 11* or *Stata 12* icon on the Windows desktop of the computer at which you are working, click on the **Start** button located at the left end of the Windows XP or Windows 7 taskbar along the bottom of the desktop window. From the **All Programs** menu, select (click on) the ***Stata 11*** **or** ***Stata 12*** **icon**.

After you start your *Stata* session, the first screen you will see contains four *Stata* windows (or five *Stata* windows of you're using *Stata 12*):

- the ***Stata Command window***, in which you type all *Stata* commands.
- the ***Stata Results window***, which displays the results of each *Stata* command as you enter it.
- the ***Review window***, which displays the past commands you have issued during the current *Stata* session.
- the ***Variables window***, which lists the variables in the currently-loaded data file.
- (in *Stata 12* only) the ***Properties window***, which displays the properties of the variables and dataset currently in memory.

❑ Record Your *Stata* Session – log using

Stata is an interactive software program that executes commands as you enter them during a *Stata* session. Being interactive is cool, but it also means that none of your commands or the results they produce will be saved for future reference unless you create a ***log file*** – that is, a file that has the extension ".log". Failure to record your *Stata* session often means wasting a lot of your time. So I strongly recommend that you always record your *Stata* sessions in *Stata* log files.

- **To record your current Stata session**, including all the *Stata* commands you enter and the results (output) produced by these commands, make a text-format log file named **452tutorial1.log**. To open (begin) the log file **452tutorial1.log**, enter in the Command window the following command:

  ```
  log using 452tutorial1.log
  ```

  or

  ```
  log using e:452tutorial1.log
  ```

  The first command opens a ***text-format*** **file** called **452tutorial1.log** in the current *Stata* working directory on the hard drive of your PC. The second command opens a text-format file called **452tutorial1.log** on a portable electronic storage device in the E:-drive. Once you have opened it, a copy of all the commands you enter during your *Stata* session and of all the results they produce is recorded in that **452tutorial1.log** file. A log file with file extension **.log** is a *text-format file*, which means it can be viewed with any text editor such as Windows Notepad or

Windows Wordpad. You can also view a text-format file using your favorite document processor, such as Microsoft Word or Corel WordPerfect.

Note: It is very important that you include the file extension **.log** in the name of your log file. Otherwise, *Stata* will open a ***formatted*** **log file** that has file extension **.smcl**; this is the default file type for *Stata* log files in *Stata 11* and *Stata 12*. The major limitation of **.smcl** log files is that you must have the *Stata 11* or *Stata 12* program installed on your computer in order to read **.smcl** log files once you have finished your *Stata* session.

- **An alternative way to open (start) the log file 452tutorial1.log** is outlined in the following steps:

  1. Click on the **Log** button in the button bar near the top of the *Stata* window; this opens the **Begin Logging *Stata* Output** dialog box.

  2. In the **Begin Logging *Stata* Output** dialog box:

     - click on **Save as type:** and select **`Log (*.log)`**;
     - click on the **File name:** box and type the file name **`452tutorial1`**;
     - click on the **Save** button.

- If the log file **452tutorial1.log** already exists in the current working directory of your C:-drive or E:-drive, you can overwrite it by simply adding the **replace** option to the **log using** command:

  ```
  log using 452tutorial1.log, replace
  ```

  or

  ```
  log using e:452tutorial1.log, replace
  ```

- **To temporarily stop recording your current *Stata* session** while leaving the log file open, enter in the Command window:

  ```
  log off
  ```

- **To resume recording your *Stata* session** to the currently open log file, enter in the Command window:

```
log on
```

- **To stop recording your current *Stata* session and close the log file**, enter in the Command window:

```
log close
```

- **To add the results for your current *Stata* session to the end of an existing log file**, add the **append** option to the **log using** command. For example, to start writing the results of your *Stata* session to the end of the file **452tutorial1.log**, which you have just closed, enter in the Command window one of the following two commands:

```
log using 452tutorial1.log, append
```

or

```
log using e:452tutorial1.log, append
```

❑ Changing the Working Directory – cd

Stata expects to find the files it needs (such as *Stata* datasets) and to write the files it creates (such as *Stata* log files) in the current working directory. The name of the current working directory is displayed in the status bar of the *Stata* Command window. You may want to change the *Stata* working directory during a *Stata* session. For instance, if the datasets you wish to use are stored in a directory different from the current working directory, you will want to switch to that directory before loading the datasets into memory. The command you use for this purpose is the **cd** command.

Basic Syntax

- **cd**

  Displays the name of the current working directory.

- **cd *drive:***

  Changes the current working directory to a new drive, such as the A:-drive or the C:-drive.

- **cd *directory***

  Changes the current working directory to a new directory in the same drive as the original directory.

- **cd *drive:directory***

  Changes the current working directory to the new drive and directory specified.

- **cd "*drive: directory name*"**

  Changes the current working directory to the new drive and directory specified, where the new directory name contains embedded spaces.

  Usage Note: Windows XP allows directory names with embedded spaces. To change the working directory to a directory with spaces in its name, enclose the entire drive and directory specification in double quotation marks.

- **cd ..**

  Moves the current working directory one level up the directory tree from the original directory.

Simple Examples

- **`cd`** Displays current working directory
- **`cd e:`** Changes working directory to the flash memory stick in the E:-drive
- **`cd c:\courses`** Changes working directory to *C:\courses*
- **`cd c:\data`** Changes working directory to *C:\data*

An Extended Example

Enter the following sequence of **cd** commands and take the time to observe the current working directory that each produces. Notice that in some cases there is more than one way to enter the **cd** command.

- Enter the command: **`cd`**

`C:\data`

- Enter the command: **`cd \`** *or* **`cd ..`**

`C:\`

- Enter the command: **`cd courses`** *or* **`cd c:\courses`**

`C:\courses`

- Enter the command: **`cd e:`**

`E:\`

- Enter the command: **`cd c:`**

`C:\`

- Enter the command: **`cd data`**

`C:\data`

❑ Displaying the Contents of Directories – dir

From time to time you will probably want to see what files are contained in the current directory (or folder). The **dir** command is used for this purpose; it displays a list of the names of some or all of the files in the current *Stata* working directory.

Simple Examples

- To display in the Results window the names of all the files in the current working directory, enter in the Command window:

```
dir
```

- To display in the Results window the names of all the files in the current working directory that have the extension **.dta**, enter in the Command window:

```
dir *.dta
```

Note that the asterisk "*****" can be used to indicate any string of characters.

- To display in the Results window the names of all the files in the current working directory that have the extension **.log**, enter in the Command window:

```
dir *.log
```

- To display in the Results window the names of all the files in the current working directory that begin with the character string 'auto', enter in the Command window:

```
dir auto*.*
```

❑ Loading a Text-Format Data File into *Stata*: Method 1 – infile

Before starting your *Stata* session, you downloaded the text-format (or ASCII-format) data file **auto1.raw** from the ECON 452 web site and placed it in a directory or folder on the C:-drive of your computer. This section demonstrates how to input that data file into *Stata* using the **infile** command.

When can the *infile* command be used?

A text-format (or ASCII-format) data file must exhibit certain properties if it is to be correctly read by an **infile** command.

1. The data can be **space-separated**, **tab-separated**, or **comma-separated**. Another term for **space-separated** is **space-delimited**; another term for **tab-separated** is **tab-delimited**; and another term for **comma-separated** is **comma-delimited**.
2. Strings with embedded spaces or commas *must be enclosed in single or double quotes*.
3. The text-format data file *must contain only variable values*. This means that the first line of the text-format data file *cannot contain variable names*.
4. A single observation can be on more than one line, or there can be more than one observation on each line.

To use the **infile** command correctly, you need to know four things about the text-format data file you are inputting:

(1) the *number of variables* in the data file;
(2) the *order* in which the values of the variables are written in the data file for each observation or record;
(3) which of the variables are *numeric variables* and which are *string variables*; and
(4) for each *string* variable, the *maximum length* of the values it takes, i.e., the maximum number of characters (including embedded spaces) its values contain.

Basic Syntax

infile *varlist* using *filename*

Loads the data in the text file ***filename*** on the variables named in ***varlist***, where the data values for the variables are separated by one or more spaces, by tabs, or by commas. Such a file is called an unformatted, or free format, text file.

1. Note that the ***varlist*** **is required**, not optional. This means that you must know both the *number* and *order* of the variables for each observation.
2. The text file ***filename*** **must not contain variable names in the first line**.

What does the text-format data file auto1.raw look like?

- Too see what the data file **auto1.raw** looks like, enter in the Command window:

```
type auto1.raw, showtabs
```

Here is the listing of the first 10 lines and last 3 lines of data file **auto1.raw** produced by the above **type** command:

```
  "AMC Concord"      4099      22      2930      0
    "AMC Pacer"      4749      17      3350      0
   "AMC Spirit"      3799      22      2640      0
"Buick Century"      4816      20      3250      0
"Buick Electra"      7827      15      4080      0
"Buick LeSabre"      5788      18      3670      0
   "Buick Opel"      4453      26      2230      0
  "Buick Regal"      5189      20      3280      0
"Buick Riviera"     10372      16      3880      0
"Buick Skylark"      4082      19      3400      0
```

(output omitted)

```
  "VW Rabbit"       4697          25       1930          1
"VW Scirocco"       6850          25       1990          1
  "Volvo 260"      11995          17       3170          1
```

Note the following properties of the data file **auto1.raw**:

(1) The number of variables in the data file is 5.
(2) The order of the variables by name is: **make**, **price**, **mpg**, **wgt**, **foreign**.
(3) The first variable in each record, **make**, is a ***string variable***; its values are contained in double quotation marks.
(4) The other four variables are ***numeric variables***.
(5) The maximum length of the string variable **make** is 18 characters (though this is not apparent from the partial listing of the data file given above).

- To load the text-format data file **auto1.raw** into memory, enter in the Command window one of the following two commands:

```
infile str18 make price mpg wgt foreign using auto1.raw
```

or

```
infile str18 make price mpg wgt foreign using auto1
```

Note two things about these **infile** commands:

- The filename can be given as either **auto1.raw** or **auto1**. If the filename has the extension **.raw**, *Stata* knows that the data file is an unformatted text file and automatically assumes that filename **auto1** actually refers to the file **auto1.raw**. If the text-format data file *does not* have the extension **.raw**, then the full filename must be given after the keyword **using** in the **infile** command.

- The keyword **str18** appears in the ***varlist*** immediately before the variable name **make**. This keyword must appear in the ***varlist*** **immediately in front of each *string* variable name**; the number **18** specifies the maximum length taken by the values of the string variable **make**.

❑ Summarizing the contents of the current data set – describe

The **describe** command displays a summary of the contents of the current data set in memory. In the present case, the current data set in memory is the text-format data file **auto1.raw**. To summarize the contents of the current data set, enter in the Command window the following two commands:

```
describe
describe, short
```

❑ Displaying the values of variables – list

The **list** command displays the values of some or all of the variables in the current data set. There are several available forms for the **list** command.

- To display in the Results window the values of all variables for all observations in data set **auto1.raw**, enter in the Command window:

```
list
```

- To display the values of all variables only for the first 20 observations in **auto1.raw**, enter in the Command window:

```
list in 1/20
```

- To display the values of all variables for observations 50 to last, enter in the Command window:

```
list in 50/l
```

- To display the values of the variables **make**, **price**, **mpg**, and **wgt** for all observations in data set **auto1.raw**, enter in the Command window:

```
list make price mpg wgt
```

- To display the values of the variables **make**, **price**, **mpg**, and **wgt** for the first 25 observations in **auto1.raw**, enter in the Command window:

```
list make price mpg wgt in 1/25
```

- To temporarily stop recording your *Stata* session to the currently open log file **452tutorial1.log**, enter in the Command window:

```
log off
```

- To display the values of the variables **mpg**, **wgt**, and **foreign** only for those observations for which the value of **mpg** is greater than 20, enter in the Command window:

```
list mpg wgt foreign if mpg > 20
```

- To resume recording your *Stata* session to the log file **452tutorial1.log**, enter in the Command window:

```
log on
```

- To display the values of the variables **mpg**, **wgt** and **foreign** only for those observations for which the value of **mpg** is greater than 20 and **foreign** equals 1, enter in the Command window:

```
list mpg wgt foreign if mpg > 20 & foreign==1
```

  Describe the results of this command; that is, explain what it does. Note the use of the double equality "==" in the **if** statement; *Stata* uses "==" for the equal sign in all *logical* or *relational* statements, but not in any other kind of statements.

❑ Calculating descriptive summary statistics – summarize

The **summarize** command calculates and displays descriptive summary statistics for some or all of the ***numeric variables*** in the current data set. The basic descriptive summary statistics produced for each numeric variable by the **summarize** command are: the number of observations (Obs); the mean (Mean); the standard deviation (Std. Dev.); and the range, specifically the minimum value (Min) and the maximum

value (Max). These summary statistics obviously cannot be computed for ***string variables***, since string variables do not take numerical values.

- To calculate and display basic descriptive summary statistics for all variables and all observations in the **auto1.raw** data set, enter in the Command window:

```
summarize
```

  Note that this command does not calculate summary statistics for the variable **make**. The reason is that **make** is a ***string variable***; the summarize command can only compute descriptive statistics for ***numeric variables***.

- To calculate and display descriptive summary statistics for the numeric variables **price**, **wgt**, **mpg** and **foreign** for all observations in the **auto1.raw** data set, enter in the Command window:

```
summarize price wgt mpg foreign
```

- To calculate and display basic descriptive summary statistics for the variables **price**, **mpg** and **wgt** for those observations in the current data set **auto1.raw** for which the value of the variable **mpg** is greater than 20, enter the command:

```
summarize price mpg wgt if mpg > 20
```

- To calculate and display a table of basic descriptive summary statistics for the variables **price**, **mpg** and **wgt** for as many subsets of observations as there are distinct values of the categorical variable **foreign**, enter the commands:

```
sort foreign
by foreign: summarize price mpg wgt
```

  The **sort** command sorts the observations in the current data set in ascending order according to the values of the variable **foreign**. Since the variable **foreign** takes only two distinct values (i.e., 1 for "Foreign" cars and 0 for "Domestic" cars), the second command produces two tables of summary statistics for the variables **price**, **mpg** and **wgt**.

- An alternative way to calculate and display separately for domestic and foreign cars basic descriptive summary statistics for the variables **price**, **mpg** and **wgt** is to use the **if *option*** on the summarize command. Enter the commands:

```
summarize price mpg wgt if foreign==0
summarize price mpg wgt if foreign==1
```

- Enter the following **summarize** command preceded by the **bysort foreign:** option, and interpret the results it produces:

```
bysort foreign: summarize price mpg wgt if mpg > 20
```

  The **bysort foreign:** option sorts the sample data by the values of the variable **foreign** if it is not already sorted by **foreign** before attempting to execute the summarize command.

- The **summarize** command has a **detail** option. To see how it works, enter the following command:

```
summarize price mpg wgt, detail
```

❑ Creating or Saving a *Stata*-format Dataset – save

Stata-format Data Files -- .dta files

Stata-format data files are ***binary format*** files, which means that you cannot read them with a text editor or word processor. Only the *Stata* program can read a *Stata*-format data file.

All *Stata*-format data files have the filename extension **.dta**. Thus, a generic *Stata*-format data file has the name ***filename*.dta**, where ***filename*** is a user-supplied name of 32 characters or less.

Creating or saving a *Stata*-format data file

1. **To save a *new* dataset** to disk (or to save an old dataset under a new name):

   - either enter in the Command window: **save *filename***

- *or* pull down the **File** menu and choose **Save As**

This command saves to the current working directory on disk the *Stata*-format data file ***filename*.dta**, where ***filename*** is a user-supplied file name.

2. **To resave a data set that has been changed** (and overwrite the original **.dta** file):

   - enter in the Command window: **save *filename*, replace**
   - *or* simply enter in the Command window: **save, replace**
   - *or* pull down the **File** menu and choose **Save**

   Note that this command overwrites the original *Stata*-format data file of the same name. If you do not wish to lose the original **.dta** file, give the changed data set a new name using the save command given in **1.** above.

- To save the data file you have been working on as a *Stata*-format dataset, enter in the Command window:

```
save auto1
```

This command saves on disk in the *Stata* working directory the *Stata*-format dataset **auto1.dta**. Note that the *Stata* **save** command automatically attaches the extension **.dta** to the ***filename*** you specify. Note too that the text-format data file **auto1.raw** you originally read into *Stata* is left unchanged on disk by the *Stata* **save** command.

- The current data set in memory is now the *Stata*-format dataset **auto1.dta**. To summarize the contents of this current dataset, enter in the Command window:

```
describe
```

Inspect the results of this command.

- To confirm that the *Stata*-format dataset **auto1.dta** has been saved to the current *Stata* working directory, enter in the Command window:

```
dir *.dta
```

Among the listed filenames you should see **auto1.dta**.

❑ Removing a Data Set From *Stata* Memory – clear

Stata can have only one data set in memory at a time. To remove or clear the current data set **auto1.dta** from memory, enter in the Command window:

```
clear
```

Once *Stata* executes this **clear** command, there is no current data set in memory. But of course the *Stata*-format dataset **auto1.dta** is still sitting on your computer's C:-drive since you previously put it there with the **save** command.

Stata now has no dataset in memory on which to work. In a later section of this tutorial, you will learn how to use the **use** command to read an existing *Stata*-format dataset into memory.

❑ Loading a Text-Format Data File into *Stata*: Method 2 – insheet

Before starting your *Stata* session, you should have downloaded the text-format (or ASCII-format) data file **auto1.txt** from the ECON 452 web site and placed it in a directory or folder on the C:-drive of your computer. This section demonstrates how to input that data file into *Stata* using the **insheet** command.

When can the *insheet* command be used?

The **insheet** command is intended to read text files created by a spreadsheet or database program. A text-format (or ASCII-format) data file must exhibit certain properties if it is to be correctly read by an **insheet** command.

1. The data must be **tab-separated** or **comma-separated**; the **insheet** command cannot load data files that are space-separated.
2. A single observation must be on only one line. In other words, **there must be one, and only one, observation on each line** of the data file.
3. The **first line** of the text-format data file **can optionally contain *variable names***.

To use the **insheet** command correctly, you need to know relatively little about the text-format data file; *Stata* will figure out many of the file's characteristics and correctly read the data into memory. However, if the first line of the data file does not contain variable names, you should know at least the order of the variables in each observation. You should also know that the data file is **tab-delimited** or **comma-delimited**, and that each line of the file contains one, and only one, observation.

Basic Syntax

insheet [*varlist*] using *filename* [, names tab comma clear]

Loads the data in the text file ***filename*** on the variables named in ***varlist***, where the data values for the variables are separated (or delimited) by tabs or by commas. Such a file is called an unformatted, or free format, text file.

1. Note that the ***varlist*** **is optional**. If the first line of the text file ***filename*** contains variable names, the ***varlist*** can be omitted and the file will still be correctly read by the **insheet** command. In other words, when the first line of the text file ***filename*** contains variable names, you need not know either the number or order of the variables for each observation.
2. The text file ***filename*** **may contain** ***variable names*** **in the first line**. If it does not contain variable names in the first line, the variable names can optionally be given in a ***varlist***.
3. The **names** ***option*** tells *Stata* that the first line of the data file contains ***variable names***. If the data file does in fact include variable names in the first line, the **names** option simply reads the file more quickly.
4. The **tabs** ***option*** tells *Stata* that the data values are ***tab-separated***, or ***tab-delimited***. If the data file is in fact tab-separated, the **tabs** option simply reads the file more quickly.
5. The **comma** ***option*** tells *Stata* that the data values are ***comma-separated***, or ***comma-delimited***. If the data file is in fact comma-separated, the **comma** option simply enables **insheet** to read the file more quickly.
6. The **clear** ***option*** tells *Stata* to remove the current dataset from memory before attempting to load the new dataset ***filename***. Remember that *Stata* can have only one dataset in memory at a time.

Usage: To read many data files, you need only enter the following simple form of the **insheet** command:

insheet using *filename*

What does the text-format data file auto1.txt look like?

- Assuming the text-format data file **auto1.txt** is in the current working directory, you can view its contents by entering the following **type** command:

```
type auto1.txt, showtabs
```

The lines returned by the above **type** command are:

```
make,price,mpg,wgt,foreign
AMC Concord,4099,22,2930,0
AMC Pacer,4749,17,3350,0
AMC Spirit,3799,22,2640,0
Buick Century,4816,20,3250,0
Buick Electra,7827,15,4080,0
Buick LeSabre,5788,18,3670,0
Buick Opel,4453,26,2230,0
Buick Regal,5189,20,3280,0
Buick Riviera,10372,16,3880,0
Buick Skylark,4082,19,3400,0
```

(output omitted)

```
VW Rabbit,4697,25,1930,1
VW Scirocco,6850,25,1990,1
Volvo 260,11995,17,3170,1
```

Note the following properties of the text-format data file **auto1.txt**:

(1) The ***first line*** of the data file **auto1.txt** ***contains the variable names***, which are separated by commas.
(2) The variable values are ***comma-separated***, or ***comma-delimited***.
(3) Each observation is entirely contained in one line; there is ***only one observation per line***.
(4) The number of variables in the data file is 5.
(5) The order of the variables by name is: **make**, **price**, **mpg**, **wgt**, **foreign**.

(6) The values of the ***string variable*** **make** are not enclosed in single or double quotation marks.
(7) The other four variables are ***numeric variables***.

This is exactly the sort of data file that the **insheet** command can read very easily.

- To load the text-format data file **auto1.txt** into memory, type in the Command window the following command:

```
insheet using auto1.txt
```

- To display a summary of the contents of the dataset you have just loaded into memory, enter in the Command window the **describe** command:

```
describe
```

- To display in the Results window the values of all variables for all observations in the dataset **auto1.txt**, enter in the Command window:

```
list
```

Here is part of the display returned by the above **list** command:

```
                make      price        mpg        wgt    foreign
 1.       AMC Concord       4099         22       2930          0
 2.         AMC Pacer       4749         17       3350          0
 3.        AMC Spirit       3799         22       2640          0
 4.     Buick Century       4816         20       3250          0
 5.     Buick Electra       7827         15       4080          0
 6.     Buick LeSabre       5788         18       3670          0
 7.        Buick Opel       4453         26       2230          0
 8.       Buick Regal       5189         20       3280          0
 9.     Buick Riviera      10372         16       3880          0
10.     Buick Skylark       4082         19       3400          0
```

(output omitted)

```
72.         VW Rabbit       4697         25       1930          1
73.       VW Scirocco       6850         25       1990          1
74.         Volvo 260      11995         17       3170          1
```

- To compute descriptive summary statistics for the four *numeric variables* in the dataset **auto1.txt**, enter in the Command window:

```
summarize price mpg wgt foreign
```

❑ Comparing Two Datasets – cf

You currently have in memory the dataset **auto1.txt**. You previously saved the dataset **auto1.raw** in the *Stata*-format dataset **auto1.dta**. You can compare the contents of **auto1.txt** and **auto1.dta** using the *Stata* **cf** command.

Basic Syntax

cf *varlist* using *filename* [, verbose]

Compares the values of the variables in ***varlist*** in the dataset in memory with the values of the corresponding variables in ***filename***.

The **cf** command returns nothing if the specified variables are identical in the two datasets, and a return code of 9 if there are differences in the variables.

The **verbose** option provides a variable-by-variable comparison of the two datasets.

Simple Examples

- To compare all the variables in the datasets **auto1.txt** and **auto1.dta**, enter in the Command window:

```
cf _all using auto1
```

or

```
cf _all using auto1.dta
```

- To do a more detailed comparison of all the variables in the datasets **auto1.txt** and **auto1.dta**, add the **verbose *option*** to the above **cf** command. Enter in the Command window:

```
cf _all using auto1, verbose
```

or

```
cf _all using auto1.dta, verbose
```

- To compare only the variables **price** and **wgt** in the datasets **auto1.txt** and **auto1.dta**, enter in the Command window:

```
cf price wgt using auto1
```

or

```
cf price wgt using auto1.dta
```

- The results of the previous **cf** commands should be sufficient to persuade you that all the variables in the datasets **auto1.txt** and **auto1.dta** are identical. You can therefore discard the current dataset in memory, which is **auto1.txt**, by entering the following clear command in the Command window:

```
clear
```

  There is now no dataset in memory.

❑ Loading a Stata-format Dataset – use

The **use** command is *Stata*'s way of loading into memory a previously-created *Stata*-format dataset.

To load a Stata-format data file:

1. type in the Command window: **use *filename***

or

2. pull down the **File** menu and choose **Open**

This command loads into memory the *Stata*-format dataset ***filename*.dta**, where ***filename*** is a user-supplied filename of 32 characters or less.

Simple Examples

- **`use auto1`** *or* **`use auto1.dta`**
- **`use C:\courses\auto1`**

- To load or read into memory the *Stata*-format dataset **auto1.dta**, do either of the following:

  1. enter in the Command window: **`use auto1`**

  or

  2. pull down the **File** menu, choose **Open**, then choose **auto1.dta**.

  This command loads into memory the *Stata*-format dataset **auto1.dta**.

- To see that the *Stata*-format dataset **auto1.dta** is now the current dataset in memory, type in the Command window the following two commands:

```
describe
summarize
```

❑ Labeling Variables in a Stata-format Dataset – label variable

It is often helpful to give the variables in a *Stata*-format dataset variable labels that provide brief definitions of the variables. The command you use for this purpose is the **label variable** command.

Basic Syntax

label variable ***varname*** **"*user-suppled variable label*"**

where ***varname*** is the name of the variable you want to label and ***user-supplied variable label*** is the label you wish to give the variable ***varname***. Note that ***user-supplied variable label*** must be enclosed in double quotation marks.

- To assign labels to the five variables in the *Stata*-format dataset **auto1.dta**, type in the Command window the following commands:

```
label variable make "Make of car"
label variable price "Price (US dollars)"
label variable mpg "Miles per gallon"
label variable wgt "Weight (pounds)"
label variable foreign "Foreign car indicator"
```

- You can also give your dataset a short descriptive label using the **label data** command. The **label data** command works just like the **label variable** command. Type in the Command window:

```
label data "North American Car Prices, 1978"
```

  This command assigns the label **North American Car Prices, 1978** to the *Stata*-format dataset **auto1.dta**.

- You have to this point made some changes to the original dataset **auto1.dta**. In particular, you have assigned labels to the dataset and its variables. To save these changes, enter the following **save** command with the **replace** option:

```
save, replace
```

  This command saves on disk the revised dataset in the *Stata*-format data file **auto1.dta**. It actually overwrites the original disk copy of the dataset **auto1.dta**, and replaces it with the revised dataset that currently resides in memory.

- Finally, you can summarize the contents of the revised dataset **auto1.dta** by entering in the Command window:

```
describe
```

  Inspect the results of this command to see how the new labels you have created are displayed.

❑ Displaying variable characteristics – codebook

The **codebook** command displays the characteristics, or properties, of any variable(s) in the current data set.

- To display the characteristics of the variable **make**, enter in the Command window:

  ```
  codebook make
  ```

  Examine carefully the screen display: it tells you that **make** is a *string variable* with maximum length of 17 characters, that **make** takes 74 distinct values in the data set, and that there are no missing values for the variable **make**. It also displays some examples of the string values taken by the variable **make**.

- Display the characteristics of the *numeric variables* in the data set by entering in the Command window:

  ```
  codebook price mpg wgt foreign
  ```

  Again, examine carefully the screen display for each of these *numeric variables*. How is the displayed information for the *indicator variable* **foreign** different from that for the other numeric variables?

❑ Creating new variables from existing variables – generate & replace

The *Stata* **generate** command creates new variables from expressions that are combinations of existing ***variables***, ***operators***, and ***functions***.

- **Variables.** *Stata* handles two different types of *variables*: ***numeric variables***, and ***string variables***.

  - ***Numeric variables*** are variables whose values are only real numbers.

  - ***String variables*** are variables whose values are strings (or combinations) of alphabetic and/or numeric characters.

  Note: The **generate** command works only on ***numeric variables***.

- **Operators.** *Stata* has four different classes of *operators*: *arithmetic* operators, *relational* operators, *logical* operators, and *string* operators. In this course, you will probably make use only of the first three types of operators.

  - The ***arithmetic operators*** in *Stata* are:

+	addition
–	subtraction
*	multiplication
/	division
^	raise to a power

  - The ***relational operators*** in *Stata* are:

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
==	equal to (the relational operator for equality is a pair of equal signs)
~=	not equal to

  - The ***logical operators*** in *Stata* are:

&	and
|	or
~	not

- **Functions.** *Stata* has a long list of ***mathematical functions***. Among the most commonly used mathematical functions in econometrics are the following:

abs(*x*)	absolute value
exp(*x*)	exponential
ln(*x*)	natural logarithm
log(*x*)	natural logarithm
log10(*x*)	logarithm to base 10
sqrt(*x*)	square root

1. <u>Creating new continuous variables from existing continuous variables</u>. The generate command is often used to create new continuous variables from existing continuous variables that have either been loaded into memory from a data file or been previously created during the current *Stata* session.

Examples

- Create the new variable **wgtsq** to equal the squared value of the existing variable **wgt**. Enter in the Command window *either*:

```
generate wgtsq = wgt^2
```

  or

```
generate wgtsq = wgt*wgt
```

- To display the values of the newly created variable **wgtsq** and compare it with the values of the original variable **wgt**, enter the following **list** command:

```
list wgt wgtsq
```

- Generate the new variables $\ln(\text{price}_i)$, $\ln(\text{mpg}_i)$ and $\ln(\text{wgt}_i)$, the natural logarithms of the existing variables **price**, **mpg** and **wgt**. Give the variable name **lnprice** to the variable $\ln(\text{price}_i)$, the variable name **lnmpg** to the variable $\ln(\text{mpg}_i)$, and the variable name **lnwgt** to the variable $\ln(\text{wgt}_i)$. Enter in the Command window the following series of **generate** commands:

```
generate lnprice = ln(price)
generate lnmpg = ln(mpg)
gen lnwgt = ln(wgt)
```

  Note that the **generate** command can be abbreviated as **gen**; however, the use of abbreviations for command names is <u>not</u> recommended.

- To display the values of the newly created variables **lnprice**, **lnmpg**, and **lnwgt**, enter the following **list** command:

```
list lnprice lnmpg lnwgt
```

- To see the relationship between the natural logarithm function ***ln(x)*** and the exponential function ***exp(x)***, create new variables equal to the exponential function of the natural logarithms $\ln(price_i)$, $\ln(mpg_i)$ and $\ln(wgt_i)$. Enter in the Command window the following sequence of **generate** commands:

```
generate price1 = exp(lnprice)
generate mpg1 = exp(lnmpg)
generate wgt1 = exp(lnwgt)
```

- Enter the following **list** and **summarize** commands to compare the values of the newly created variables **price1**, **mpg1**, and **wgt1** with the values of the original variables **price**, **mpg**, and **wgt**:

```
list price price1 in 1/20
summarize price price1
list mpg mpg1 in 1/20
summarize mpg mpg1
list wgt wgt1 in 1/20
summarize wgt wgt1
```

  Inspect the results displayed by these **list** commands. They illustrate the fact that $\exp(\ln(x)) = x$ for $x > 0$.

- You can also use the *Stata* **compare** command to confirm that the variables **price** and **price1** are identical or equal to each other. You can do the same thing to confirm the equivalence of the two fuel efficiency variables **mpg** and **mpg1**, and the equivalence of the two weight variables **wgt** and **wgt1**. Enter in the Command window the following sequence of three **compare** commands:

```
compare price price1
compare mpg mpg1
compare wgt wgt1
```

- Now that you have verified by example the relationship between the natural logarithm function ***ln(x)*** and the exponential function ***exp(x)***, you can use the *Stata* **drop** command to eliminate from the dataset the redundant variables **price1**, **mpg1**, and **wgt1**. Enter the command:

```
drop price1 mpg1 wgt1
```

2. <u>Converting continuous to indicator variables</u>. You can use the *Stata* **generate** command to create a set of ***indicator (or dummy) variables*** which identify different ranges of values of a continuous variable. Indicator variables are ***binary variables*** that are defined to take only two values, 0 and 1. The value 1 indicates the presence of some characteristic or attribute; the value 0 indicates the absence of that characteristic or attribute. Indicator variables are used extensively in applied econometrics.

Suppose you want to create three indicator variables to identify the following three ranges of values of the continuous variable **mpg**:

(1) mpg < 18;

(2) 18 ≤ mpg ≤ 22 ;

(3) mpg > 22.

Define the indicator variable **mpglt18** to equal 1 if mpg < 18, and 0 otherwise. Define the indicator variable **mpg1822** to equal 1 if mpg ≥ 18 <u>and</u> mpg ≤ 22, and 0 otherwise. Finally, define the indicator variable **mpggt22** to equal 1 if mpg > 22, and 0 otherwise.

- To create the three indicator variables **mpglt18**, **mpg1822**, and **mpggt22**, enter the following three **generate** commands:

```
generate mpglt18 = mpg < 18
generate mpg1822 = mpg >= 18 & mpg <= 22
generate mpggt22 = mpg > 22
```

- To inspect the results of these three commands, enter the following **list** command:

```
list mpg mpglt18 mpg1822 mpggt22
```

Examine the results displayed by this **list** command.

3. <u>Converting continuous to categorical variables</u>. A ***categorical variable*** identifies groups to which observations belong. For example, you may want to create a categorical variable that indicates which of the following three ranges of values are taken by the continuous variable **mpg** for each observation: (1) mpg < 18; (2) $18 \leq \text{mpg} \leq 22$; and (3) mpg > 22.

- ***<u>Method 1</u>:*** The following sequence of **generate** and **replace** commands is one way of creating the categorical variable **mpgcat1**, the values of which are defined as follows: mpgcat1 = 1 if mpg < 18; mpgcat1 = 2 if $18 \leq \text{mpg} \leq 22$; and mpgcat1 = 3 if mpg > 22.

```
generate mpgcat1=1 if mpg < 18
replace mpgcat1=2 if mpg >= 18 & mpg <= 22
replace mpgcat1=3 if mpg > 22
```

- ***<u>Method 2</u>:*** An alternative way of creating a categorical variable identifying the three ranges of values of the continuous variable **mpg** makes use of the *Stata* **recode( )** function. The following **generate** command with the **recode( )** function creates the categorical variable **mpgcat2**, the values of which are defined as follows: mpgcat2 = 17 if mpg ≤ 17; mpgcat2 = 22 if $18 \leq \text{mpg} \leq 22$; and mpgcat2 = 99 if mpg > 22.

```
generate mpgcat2 = recode(mpg,17,22,99)
```

  The **recode ( )** function has three or more arguments, the first of which must be the <u>*continuous*</u> *variable* (in this case **mpg**) from which the <u>*categorical*</u> *variable* (in this case **mpgcat2**) is being constructed. The above command works as follows. For each observation, **recode( )** asks if **mpg** is less than or equal to 17; if so, the value of **mpgcat2** is set equal to 17. If not, **recode( )** then asks if the value of **mpg** is less than or equal to 22; if so, the value of **mpgcat2** is set equal to 22. If not, the value of **mpgcat2** is set equal to 99, which in this case indicates a value of **mpg** greater than 22.

- Use the following **list** command to display the values of the continuous variable **mpg** and the two categorical variables **mpgcat1** and **mpgcat2**.

```
list mpg mpgcat1 mpgcat2
```

- Use the following **compare** command to see if the two categorical variables **mpgcat1** and **mpgcat2** you have created are identical or equal to each other.

```
compare mpgcat1 mpgcat2
```

❑ Re-saving an existing *Stata*-format Dataset – save, replace

You have to this point made several changes to the original dataset **auto1.dta**. In particular, you have created several new variables. Suppose you want to save to disk the expanded *Stata*-format dataset you have created without changing its name, i.e., you want to save to disk the expanded dataset currently in memory and give it the name **auto1.dta**. You do this using the Stata save command with the replace option. Enter the following **save** command with the **replace** option:

```
save, replace
```

This command saves on disk the new expanded dataset in the *Stata*-format dataset **auto1.dta**. It overwrites the original dataset **auto1.dta**, meaning that the newly saved *Stata* dataset **auto1.dta** now contains the new variables you have created since the most recent **save** command you issued during your *Stata* session. To see exactly what variables are now contained in the dataset **auto1.dta** on disk, enter in the Command window the *Stata* **describe** command:

```
describe
```

❑ Computing sample correlations and covariances – correlate

The *Stata* **correlate** command calculates and displays the ***correlation* matrix** or ***covariance* matrix** for a group of two or more *numeric variables*.

- To calculate and display the ***correlation matrix*** for the numeric variables **price**, **mpg**, **wgt**, and **foreign**, enter in the Command window:

```
correlate price mpg wgt foreign
```

Examine carefully the results of this command.

- Each *diagonal element* of the correlation matrix equals 1 because each variable is perfectly positively correlated with itself (no surprise there).

- Each *off-diagonal element* in the correlation matrix contains the sample correlation coefficient for the two variables that correspond to the row and column in question.

- To calculate and display the ***covariance matrix*** for the variables **price**, **mpg**, **wgt**, and **foreign**, use the **covariance *option*** on the **correlate** command. Type in the Command window:

```
correlate price mpg wgt foreign, covariance
```

- To calculate separate correlation matrices for the variables **price**, **mpg** and **wgt** for the two values of **foreign**, enter in the Command window:

```
bysort foreign: correlate price mpg wgt
```

- See what you get when you enter the following **correlate** commands with the **means *option***:

```
correlate mpg wgt, means
bysort foreign: correlate mpg wgt, means
```

❑ Computing frequency tables for numeric variables – tabulate, tab1, tab2

The *Stata* **tabulate** command computes ***one-way*** and ***two-way*** tables of frequency counts for *numeric variables*, together with various measures of association such as the common Pearson chi-square statistic. Note that the **tabulate** command may be used only with *numeric variables*. In practice, the **tabulate** command is most commonly used with *discrete* or *categorical variables***,** not with continuous variables.

♦ ***One-way frequency tables.*** Use the **tabulate** command to compute and display a one-way table of frequency counts for a single categorical variable.

- To calculate and display a one-way frequency table for the variable **foreign**, enter the following **tabulate** and **tab1** commands:

```
tabulate foreign
tab1 foreign, nolabel
```

Note the use of the **nolabel** ***option*** on the **tab1** command; it causes the numeric values for the variable **foreign** to be displayed rather than the value labels. Note too that **tab** can be used as an abbreviation of the command names **tabulate** and **tab1**.

- To calculate and display one-way frequency tables for the variables **mpgcat1** and **mpgcat2**, enter the following **tabulate** commands:

```
tabulate mpgcat1
tab mpgcat2
tab1 mpgcat1 mpgcat2
```

Compare the frequency tables for **mpgcat1** and **mpgcat2**; they should be identical. Note that the third command above, the **tab1** command, computes and displays one-way frequency tables for both the categorical variables **mpgcat1** and **mpgcat2**; the **tab1** command can thus be used to compute *one-way* frequency tables for a set of two or more discrete variables.

♦ ***Two-way frequency tables.*** Use the **tabulate** command to compute and display a two-way table of frequency counts for a pair of categorical variables.

- To calculate and display a two-way frequency table for the categorical variables **mpgcat1** and **foreign**, enter the following **tabulate** command:

```
tabulate mpgcat1 foreign
```

Note that the first variable in the variable list following the command name **tabulate**, **mpgcat1**, identifies the variable whose values are displayed in the rows of the two-way frequency table; the second variable following the **tabulate**

command name, **foreign**, identifies the variable whose values are displayed in the columns of the two-way frequency table. Since the categorical variable **mpgcat1** takes three distinct values (1, 2 and 3) and the binary indicator variable **foreign** takes two distinct values (0 and 1), the two-way frequency table produced by the above **tabulate** command has three rows and two columns.

- To display a two-way frequency table that reverses the roles of the two categorical variables **mpgcat1** and **foreign** in the preceding **tabulate** command, enter the following **tabulate** command:

```
tabulate foreign mpgcat1
```

  Note that the values of the binary indicator variable **foreign** now correspond to the rows, and the values of the categorical variable **mpgcat1** to the columns, of the two-way frequency table displayed by this **tabulate** command.

- Use the above **tabulate** command with the **chi2** ***option*** to calculate and display a two-way frequency table for the categorical variables **foreign** and **mpgcat1** and to calculate the Pearson chi-square statistic for testing the null hypothesis that the categorical variables **foreign** and **mpgcat1** are statistically independent.

```
tabulate mpgcat1 foreign, chi2
```

- To calculate and display a two-way frequency table for the categorical variables **mpgcat1** and **mpgcat2**, enter either of the following **tabulate** commands:

```
tab mpgcat1 mpgcat2
tab2 mpgcat1 mpgcat2
```

  Examine the results of these two commands; they are identical. Note too that the results of these two identical commands demonstrate that the two categorical variables **mpgcat1** and **mpgcat2** are identical.

- To confirm that the two categorical variables **mpgcat1** and **mpgcat2** are identical to each other, enter in the Command window the following *Stata* **compare** command:

```
compare mpgcat1 mpgcat2
```

- Commonly-used ***options*** for two-way frequency tables are **column** and **row**. The **column *option*** displays in each cell of a two-way table the ***relative frequency*** (or ***percentage***) of that cell within its column. The **row *option*** displays in each cell of a two-way table the ***relative frequency*** (or ***percentage***) of that cell within its row. To see the effect of these two options, enter the following commands and examine the results they produce.

```
tab mpgcat1 foreign, column
tab mpgcat1 foreign, row
tab mpgcat1 foreign, row column
```

❑ Displaying tables of descriptive statistics for numeric variables – table

The *Stata* **table** command computes and displays tables of descriptive statistics for *numeric* variables. Like the **tabulate** command, the **table** command may be used only with *numeric variables*.

Basic Syntax

table *rowvar* [*colvar*] [if *exp*] [in *range*] [*weight*], contents(*clist*) row col center left

where the elements of ***clist*** may be:

`freq`	frequency
`mean` *varname*	mean of *varname*
`sd` *varname*	standard deviation of *varname*
`sum` *varname*	sum of *varname*
`rawsum` *varname*	sum of *varname* ignoring optionally specified weight
`count` *varname*	count of nonmissing observations for *varname*
`n` *varname*	same as **`count`**
`max` *varname*	maximum of *varname*
`min` *varname*	minimum of *varname*
`median` *varname*	median of *varname*
`p1` *varname*	1st percentile of *varname*
`p2` *varname*	2nd percentile of *varname*
…	
`p50` *varname*	50th percentile (median) of *varname*

…

`p98` *varname*	98th percentile of *varname*
`p99` *varname*	99th percentile of *varname*
`iqr` *varname*	interquartile range of *varname*

Common Options for 'table' command

contents(*clist*) specifies the contents of the table's cells; if this option is not specified, the default is to use **`contents(freq)`**, i.e., to produce a table of frequency counts. NOTE: No more than **five statistics** may be specified for a single table.

row adds a row to the table reflecting the total across rows.

col adds a column to the table reflecting the total across columns.

center specifies that the statistics are to be centered in the table's cells. The default is to right align the results in the table's cells.

left specifies that the column labels are to be left aligned. The default is to right align the column labels.

In practice, the variables that determine the table's rows (***rowvar***) and columns (***colvar***) are almost always *discrete* or *categorical variables***,** not *continuous* variables.

Examples

- Calculate and display a table containing the number of nonmissing observations, the mean, the standard deviation, and the minimum and the maximum values of the variable **price** for the two types of cars distinguished by the binary indicator variable **foreign**. Enter the following **table** command:

```
table foreign, contents(n price mean price sd price min price
max price)
```

- Calculate and display a table containing the number of nonmissing observations, the mean, the standard deviation, and the minimum and the maximum values of the variable **wgt** for the two types of cars distinguished by the binary indicator variable **foreign**. Enter the following **table** command:

```
table foreign, contents(n wgt mean wgt sd wgt min wgt max
wgt)
```

- Calculate and display a table containing the number of nonmissing observations, the mean, the standard deviation, and the minimum and the maximum values of the variable **mpg** for the two types of cars distinguished by the binary indicator variable **foreign**. Enter the following **table** command:

```
table foreign, contents(n mpg mean mpg sd mpg min mpg max
mpg)
```

- To calculate and display a table containing the sample means of the variables **price**, **wgt** and **mpg** for the two types of cars distinguished by the binary indicator variable **foreign**, enter the following **table** command:

```
table foreign, contents(mean price mean wgt mean mpg)
```

- Add the option **row** to the preceding table command, and note how it changes the table. Enter the command:

```
table foreign, contents(mean price mean wgt mean mpg) row
```

- To compute and display a table that contains the frequency counts (number of sample observations) and the sample means of the variables **price**, **wgt**, **mpg** and **foreign** for the three categories of cars distinguished by the categorical variable **mpgcat1**, enter the following **table** command:

```
table mpgcat1, contents(freq mean price mean wgt mean mpg
mean foreign) row
```

- To compute and display a table that contains the 25-th, 50-th and 75-th percentiles and the interquartile range (the 75-th percentile minus the 25-th percentile) of the variable **price** for the three categories of cars distinguished by

the categorical variable **mpgcat1**, enter the following **table** command:

```
table mpgcat1, contents(p25 price p50 price p75 price iqr
price) row
```

How would you interpret the values in the last row of the table produced by the above **table** command?

❑ Preparing to End Your *Stata* Session

Before you end your *Stata* session, you should do two things.

- First, you will want to **save the current data set**. Enter the following **save** command with the **replace** option to save the current data set as *Stata*-format data set **auto1.dta**:

```
save auto1, replace
```

- Second, **close the log file** you have been recording. Enter the command:

```
log close
```

  Alternatively, you could have closed the log file by:

  - clicking on the **Log** button in the *Stata* button bar;
  - clicking on **Close log file** in the ***Stata* Log Options** dialog box;
  - clicking the **OK** button.

❑ End Your *Stata* Session – exit

- **To end your *Stata* session**, use the **exit** command. Enter the command:

```
exit      or   exit, clear
```

❑ Cleaning Up and Clearing Out

After returning to Windows, you should copy all the files you have used and created during your *Stata* session to your own portable storage device such as a flash memory stick. These files will be found in the ***Stata working directory***, which is usually **C:\data** on the computers in Dunning 350. There are four files you will want to be sure you have: the text-format data files **auto1.raw** and **auto1.txt**, the *Stata*-format dataset **auto1.dta**, and the *Stata* log file **452tutorial1.log**. Use the Windows **copy** command to copy these files to your own flash memory stick.

Finally, **as a courtesy to other users** of the computing classroom, please delete all the files you have used or created from the *Stata* working directory.