Stata 10 **Tutorial 2**

TOPIC: **Introduction to Selected *Stata* Commands**

DATA: **auto1.dta** (the *Stata*-format data file you created in *Stata Tutorial 1*)
or
auto1.raw (the original text-format data file)

TASKS: ***Stata 10 Tutorial 2*** is intended to introduce you to some of *Stata*'s data display capabilities and to some important data management and statistical commands of the *Stata* statistical software program. It demonstrates several important *Stata* commands that students of introductory econometrics will use frequently.

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

generate	Creates new numeric variables from expressions containing existing numeric 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
drop	Eliminates or deletes variables or observations from the dataset in memory.
label values	Assigns a single value label to a variable.
label define	Assigns a value label to each distinct value of a variable.
graph bar	Draws bar charts of sample means of numeric variables.
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.

NOTE: *Stata* commands are *case sensitive*. All *Stata* commands 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. 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**

❑ Preparing for Your *Stata* Session

In *Stata Tutorial 1*, you created and saved on your own diskette the *Stata*-format dataset **auto1.dta**. This is the dataset you want to use to begin *Stata Tutorial 2*.

Before beginning your *Stata* session, use Windows Explorer to copy the *Stata*-format dataset **auto1.dta** to the *Stata* ***working directory*** on the C:-drive or D:-drive of the computer at which you are working.

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

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

❑ Start Your *Stata* Session

To start your *Stata* session, double-click on the ***Stata 10*** **icon** on the Windows desktop.

After you double-click the ***Stata 10*** **icon**, the first screen you will see contains four *Stata* windows:

- 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.

❑ Record Your *Stata* Session – log using

To record your *Stata* session, including all the *Stata* commands you enter and the results (output) produced by these commands, make a **.log** file named **351tutor2.log**. To open (begin) the **.log** file **351tutor2.log**, enter in the Command window one of the following two commands:

```
log using 351tutor2.log
```

or

```
log using e:351tutor2.log
```

The first command opens a text format file called **351tutor2.log** in the current *Stata* working directory. The second command opens a text format file called **351tutor2.log** on a portable electronic storage device (such as a flash memory stick) in the E:-drive. Once you have opened the **351tutor2.log** file, a copy of all the commands you enter during your *Stata* session and of all the results they produce is recorded in that **351tutor2.log** file.

An alternative way to open the **.log** file **351tutor2.log** is to click on the **Log** button; click on **Save as type:** and select **`Log (*.log)`**; click on the **File name:** box and type the file name **351tutor2**; and click on the **Save** button.

❑ Loading a *Stata*-Format Dataset into *Stata* – use

In *Stata Tutorial 1*, you created a *Stata*-format dataset **auto1.dta**. If you saved the dataset **auto1.dta** on your own portable electronic storage device in *Stata Tutorial 1* and copied this dataset to the *Stata* working directory before beginning your *Stata* session, you can simply use the **use** command to read or load **auto1.dta** into memory where *Stata* can go to work on it. If, however, you did not have the foresight to save the *Stata*-format dataset **auto1.dta** you created during *Stata Tutorial 1* and bring it with you on a portable electronic storage device, you will have to repeat most of that section of *Stata Tutorial 1* entitled "Getting Started with *Stata*".

To load, or read, into memory the *Stata*-format dataset auto1.dta, enter in the Command window:

```
use auto1
```

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

To summarize the contents of the current dataset, use the **describe** command. Recall from *Stata Tutorial 1* that the **describe** command displays a summary of the contents of the current dataset in memory, which in this case is the *Stata*-format data file **auto1.dta**. Enter in the Command window:

```
describe
```

- Examine the results of this command. In particular, check to see if the dataset is sorted according to the values of the binary indicator variable **foreign**. If it is not sorted, enter the command

  ```
  sort foreign
  ```

  This command sorts the observations (rows) of the current dataset in ascending order of the values of the variable **foreign**, from smallest to largest.

- To save the sorted dataset and overwrite or replace the original disk copy of **auto1.dta**, enter the command

  ```
  save, replace
  ```

  The 74 observations in the *Stata*-format dataset **auto1.dta** are now sorted by the values of the variable **foreign**.

Displaying values of variables – list. To refresh your memory on just what the *Stata*-format dataset **auto1.dta** looks like, use the **list** command to display the values of some or all of the variables in **auto1.dta**.

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

  ```
  list
  ```

- To display the values of all variables only for observations 40 to 65 inclusive, enter in the Command window:

```
list in 40/65
```

❑ Calculating descriptive summary statistics – summarize

Recall from *Stata Tutorial 1* that the **summarize** command calculates and displays descriptive summary statistics for some or all of the *numeric variables* in the current dataset.

- To calculate and display basic descriptive summary statistics for all variables and all observations in the current dataset **auto1.dta**, enter in the Command window:

```
summarize
```

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

```
by foreign: summarize price mpg weight
```

  Since the variable **foreign** takes only two distinct values (i.e., 1 = "Foreign" and 0 = "Domestic"), this command produces two tables of summary statistics for the variables **price**, **mpg** and **weight**.

  From inspection of the results of the previous command, it is apparent that the sample means of the variables **price**, **mpg** and **weight** differ numerically between domestic cars (for which the indicator variable $\text{foreign}_i = 0$) and foreign cars (for which the indicator variable $\text{foreign}_i = 1$). Later in this tutorial you will learn how *Stata* can be used to make bar charts that graphically display the differences in sample means between domestic and foreign cars. But before that, you will learn in the next section how to assign labels to the values of a variable.

❑ Labeling Variable Values – label values *and* label define

The ***categorical variable*** **foreign** is a binary ***indicator (or dummy) variable*** that takes only two values, 0 and 1. The value 0 identifies a domestic car (i.e., a car assembled in North America); the value 1 identifies a foreign car (i.e., a car assembled outside North America). When working with indicator variables, it is often helpful and convenient to assign short descriptive labels to the two values that such variables take.

In the case of the indicator variable **foreign**, suppose you want to assign the label "Domestic" to the value 0 and the label "Foreign" to the value 1. Two steps are required to do this.

- **Step 1:** Attach a value label to the variable **foreign** -- **label values**.

  The **label values** command assigns a single value label to a variable. Its basic syntax is

  label values *varname lblname*

  where ***varname*** is the name of the variable whose values you wish to label, and ***lblname*** is the value label you want to assign to the variable ***varname***.

- To assign the value label ***cartype*** to the variable **foreign**, enter the command

```
label values foreign cartype
```

- **Step 2:** Create value labels for each distinct value of the variable **foreign**.

  The **label define** command assigns a value label to each distinct value of a variable. Its basic syntax is

  label define *lblname* # "*text*" # "*text*" …

  where ***lblname*** is the value label of the variable whose values are to labeled, **#** denotes each distinct value that the variable takes in the dataset, and ***text*** is the user-supplied label given to the preceding value **#**. An example is the easiest to way to understand how the **label define** command works.

- To assign the label ***Domestic*** to the value 0 and the label ***Foreign*** to the value 1 of the indicator variable **foreign** whose value label is ***cartype***, enter the command

```
label define cartype 0 "Domestic" 1 "Foreign"
```

- Enter the following commands to see the results of the **label values** and **label define** commands you have just issued:

```
describe
list price mpg weight foreign in 40/65
```

  Note that the **list** command displays the value labels you have just assigned to the values 0 and 1 of the variable **foreign**.

- To display the numeric values of the variable **foreign** rather than their corresponding value labels, use the **nolabel** *option* on the **list** command. Enter the command

```
list price mpg weight foreign in 40/65, nolabel
```

- To save the current dataset with the new value labels you have created for the indicator variable **foreign** and replace the original disk copy of **auto1.dta**, enter the command

```
save, replace
```

❑ Drawing Bar Charts – graph bar

The *Stata* **graph** command is a very powerful tool for drawing a variety of graphs and charts. In this section, you learn how to use the **graph bar** command with the **(mean)** and **over( )** ***options*** to draw simple bar charts that display and compare the sample means of the three continuous variables **price**, **mpg**, and **weight** for the two types of cars – domestic and foreign – identified by the categorical variable **foreign**.

Basic Syntax

graph bar (mean) ***varname*****, over(*****category*****)**

where ***varname*** is the name of the variable you want to graph and ***category*** is the name of the categorical variable that identifies the observation subsets for which you want to draw separate bars.

The two options used in this **graph bar** command are:

(mean) specifies that the bars are to have heights equal to the sample means of the variable ***varname*** for the categories identified by ***category***;

over(***category*****)** specifies the observation categories for which separate bars are to be drawn.

Examples

- Draw a bar chart of the sample mean values of the variable **price** for domestic and foreign cars. Enter the command

```
graph bar (mean) price, over(foreign)
```

  The results of this command are displayed in the Graph window, which covers the Results window. To put the Results window back on top, use the mouse pointer to click on the **Results** button in the *Stata* button bar near the top of the screen. To switch back to the Graph window and display the results of the most recently issued **graph** command, click on the **Graph** button. The **Results** and **Graph** buttons allow you to switch back and forth between the Results window and the Graph window.

- Add the **relabel ( )** *sub-option* to the **over( )** *option* of the above **graph** command to override the default labeling of the values of the categorical variable **foreign** on the x-axis. This sub-option is used to assign your own labels to the bars for domestic and foreign cars. Enter the command

```
graph bar (mean) price, over(foreign, relabel (1 “Domestic
Cars” 2 “Foreign Cars”))
```

The result of this command is to relabel the first value of the categorical variable **foreign** to be "Domestic Cars", and the second value of foreign to be "Foreign Cars". Note that the values "1" and "2" refer to the category numbers as determined by the current sorting of the sample observations according to the values of the variable **foreign**: the value "1" refers to the first value of the variable **foreign**, which is 0; the value "2" refers to the second value of the variable **foreign**, which is 1.

- Use the **title( )** and **subtitle( )** *options* to add an appropriate title and subtitle to the bar chart created by the above **graph** command. Enter in the Command window *on one line* the command

```
graph bar (mean) price, over(foreign, relabel (1 "Domestic
Cars" 2 "Foreign Cars")) title ("Average Price of Domestic
and Foreign Cars") subtitle ("North America, 1978")
```

- To further enhance the bar chart produced by the above **graph** command, use the **ytitle( )** *option* to assign your own label to the vertical y-axis of the bar chart. Enter in the Command window *on one line* the command

```
graph bar (mean) price, over(foreign, relabel (1 "Domestic
Cars" 2 "Foreign Cars")) title ("Average Price of Domestic
and Foreign Cars") subtitle ("North America, 1978")
ytitle("Average car price, U.S. dollars")
```

- Now draw a bar chart of the sample mean values of the variable **mpg** for domestic and foreign cars. Enter the command

```
graph bar (mean) mpg, over(foreign, relabel (1 "Domestic
Cars" 2 "Foreign Cars"))
```

- Enhance the bar chart produced by the above **graph** command for the variable **mpg** by adding appropriate **title( )**, **subtitle( )** and **ytitle( )** *options*. Enter the command

```
graph bar (mean) mpg, over(foreign, relabel (1 "Domestic
Cars" 2 "Foreign Cars")) title ("Average Fuel Efficiency of
Domestic and Foreign Cars") subtitle ("North America, 1978")
ytitle("Average fuel efficiency" "in miles per gallon")
```

- Finally, draw a bar chart of the sample mean values of the variable **weight** for domestic and foreign cars. Enter the command

```
graph bar (mean) weight, over(foreign)
```

- Enhance the bar chart produced by the above **graph** command for the variable **weight** by adding appropriate **title( )**, **subtitle( )** and **ytitle( )** *options*. Enter the command

```
graph bar (mean) weight, over(foreign, relabel (1 "Domestic
Cars" 2 "Foreign Cars")) title ("Average Weight of Domestic
and Foreign Cars") subtitle ("North America, 1978")
ytitle("Average car weight, pounds")
```

❑ Creating new variables from existing variables – generate

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 elementary 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. Creating new continuous variables from existing continuous variables. 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 **weightsq** to equal the squared value of the existing variable **weight**. Enter in the Command window *either*:

```
generate weightsq = weight^2
```

or

```
generate weightsq = weight*weight
```

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

```
list weight weightsq
```

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

```
gen lnprice = ln(price)
gen lnmpg = ln(mpg)
gen lnweight = ln(weight)
```

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

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

```
list lnprice lnmpg lnweight
```

- 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 $\ln(\text{price}_i)$, $\ln(\text{mpg}_i)$ and $\ln(\text{weight}_i)$. Enter in the Command window the following sequence of **generate** commands:

```
generate price1 = exp(lnprice)
generate mpg1 = exp(lnmpg)
generate weight1 = exp(lnweight)
```

Enter the following **list** and **summarize** commands to compare the values of the newly created variables **price1**, **mpg1**, and **weight1** with the values of the original variables **price**, **mpg**, and **weight**:

```
list price price1 in 1/20
summarize price price1
```

```
list mpg mpg1 in 1/20
summarize mpg mpg1
list weight weight1 in 1/20
summarize weight weight1
```

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 **weight** and **weight1**. Enter in the Command window the following sequence of three **compare** commands:

```
compare price price1
compare mpg mpg1
compare weight weight1
```

- 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 **weight1**. Enter the command:

```
drop price1 mpg1 weight1
```

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 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) $18 \leq$ 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 and 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) $18 \leq$ mpg ≤ 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 $18 \leq$ mpg ≤ 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
```

- ***Method 2:*** 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 $18 \leq$ mpg ≤ 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 *continuous variable* (in this case **mpg**) from which the *categorical 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
  ```

❑ Creating a new *Stata*-format Dataset -- save

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 dataset you have created, but you do not want to lose the original dataset **auto1.dta**, which is stored on disk in your working directory. One way you can do this is to use the *Stata* **save** command to give the expanded dataset

a new name that is different from the name of the original dataset **auto1.dta**, and save the new expanded dataset to disk. Enter the following **save** command:

```
save auto2
```

This command saves on disk the new expanded dataset in the *Stata*-format data file **auto2.dta**. It does not overwrite the original dataset **auto1.dta**, which remains stored on disk. Note, however, that the *current* dataset – that is, the dataset that currently resides in memory – is the new expanded dataset **auto2.dta**.

❑ 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**, **weight**, and **foreign**, enter in the Command window:

  ```
  correlate price mpg weight 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**, **weight**, and **foreign**, use the **covariance *option*** on the **correlate** command. Enter in the Command window:

  ```
  correlate price mpg weight foreign, covariance
  ```

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

  ```
  bysort foreign: correlate price mpg weight
  ```

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

```
correlate mpg weight, means
bysort foreign: correlate mpg weight, 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.

- The **plot *option*** can be used with the **tabulate** command to produce a bar chart of the relative frequencies in a one-way frequency table. Enter the command

```
tab1 mpgcat1, plot
```

♦ ***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 **mpgcat1** and **foreign**

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.

- 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 **weight** for the two types of cars distinguished by the binary indicator variable **foreign**. Enter the following **table** command:

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

- 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**, **weight** 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 weight 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 weight 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**, **weight**, **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 weight 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, use the following **save** command with the **replace *option*** to save any changes you have made to your new expanded dataset **auto2.dta**:

```
save, replace
```

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

```
log close
```

❑ End Your *Stata* Session -- exit

You are now ready to conclude your *Stata* session.

- **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 electronic storage device. 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 take with you: the text-format data files **auto1.raw**, the original *Stata*-format dataset **auto1.dta**, the new *Stata*-format dataset **auto2.dta**, and the *Stata* log file **351tutor2.log**. Use the Windows **copy** command to copy any files you want to keep to your own portable electronic storage device (e.g., flash memory stick) in the E:-drive (or to a diskette in the A:-drive).

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.