Exercise No. 4: Data Management -- Part 1

New Stata Commands		**Old Commands Reviewed**	
rename	label values	describe	
label variable	label define	help	
label data	drop	summarize	
destring	histogram *var,* start(.) width(.)	graph box *var,* over(.)	
generate	graph7 *var,* oneway		
egen			
list			
replace			

This exercise begins a semester-long exploration of data-handling facilities in Stata. You will learn how to manipulate data, create new variables and data sets and create subsets of data that can be subjected to independent analysis. This exercise covers roughly the first half of Chapter 2 (pages 13 - 38) of *Statistics with Stata 12.* You should study this section carefully in order to understand the commands covered there that are not used in this exercise.

As you complete your exercise, be sure to include in the exercise the material requested in paragraphs that are check-marked (✓).

{Be sure to open a log and keep it for your records}

```
26 27 27 27 27 29 30 30 30 30 31 31 31 32 32 33 33 33 33 33 34 34 34 35 35 36
36 36 37 37 37 37 37 37 37 39 39 39 39 39 39 39 40 41 42 42 42 42 42 43 43 43
43 43 43 43 43 44 44 44 44 44 44 45 45 45 45 45 45 45 46 46 46 46 46 46 47 47
47 47 47 47 48 48 48 48 48 48 48 48 49 49 49 49 50 50 51 51 51 51 51 52 52 52
52 52 53 53 53 53 53 54 54 54 54 54 55 55 55 56 56 56 56 56 57 57 57 57 58 58
58 58 58 58 58 58 59 59 59 59 60 60 60 60 60 60 61 61 61 61 61 61 62 62 62 63
63 64 65 66 66 66 67 67 67 67 68 68 69 69 69 69 69 69 69 69 71 71 72 73 74 74
74 75 75 76 76 78 80 80 80 80 81 81 81 82 82 83 83 83 83 84 84 84 84 84 84 84
90 90 90 91 91 91 92 92 92 93 93 93 93 95 95
```

1. In the Sanitary District of Chicago, operating engineers are hired on the basis of a competitive civil-service examination. In 1966, there were 223 applicants for 15 jobs. The exam was held on March 12; the test scores are shown above, arranged in increasing order. On the basis of these data, the examiners were charged with rigging the exam. Why? To answer this question, you need to do the following:

{ Enter the scores into a Stata spreadsheet. You want to put all 223 scores into a single column. Note: *Do not* put a variable name in the first row of the spreadsheet. If you do, Stata will think that you're entering character (or "string") data for that column. All the numbers you type in after that will be treated as characters rather than numbers.

{ Rename the variable *var1* to *score* and label the variable "Examination Score." Use the following commands:

```
rename var1 score

label variable score "Examination Score"
```

{ You can also add information to the dataset, such as giving the dataset as a whole a name:

```
label data "Chicago Civil Service Exam"
```

✓ Now **describe** the data set and paste the results into your exercise. Check the results of describe to determine if *score* really is being treated as numeric data rather than character data. Suppose looking at the column labelled "storage type" you don't see "float" (or "byte" or "int" or "long" or "double") but "str21" or something else that starts with "str…". What do you do then, start over and reenter all those numbers? No, you don't have to do that. Instead, use *either* the command **destring** *or* the command **generate** *newvar* = real(*score*) to create a numeric variable that you can work with. Go to **help** *destring* to use the easiest of two commands. Go to the end of the exercise to see a description of Stata data types.

{ How can you be sure that you did not make a mistake entering the data into Stata? (Hint: **summarize** *score* and see if your results match mine. If they do, fine. If not, find the error.) How would you insure the accuracy of your data if I hadn't given you the information below?

```
             Examination Score
-------------------------------------------------------------
      Percentiles      Smallest
 1%           27             26
 5%           31             27
10%           34             27       Obs                 223
25%           43             27       Sum of Wgt.         223

50%           54                      Mean           56.18834
                        Largest       Std. Dev.      17.58609
75%           68             93
90%           83             93       Variance       309.2707
95%           91             95       Skewness       .4672641
99%           93             95       Kurtosis       2.367891
```

✓ Now, paste your (correct) **summarize** results into your exercise, so we can agree that we're all working on the same data.

✓ Now, let's see if we can understand why the Civil Service Commission thought the examiners had rigged the exam. First, generate a **oneway** scatter diagram of the data and paste it into your exercise.[1] Second, generate a histogram using options **start(25**) and

[1] See the command "graph7 ..." in the table on page 1. This is a Stata 7 command. Stata 12 graphics do not have a one way scatter plot. Type "help graph7" in Stata to read more about this and other Stata 7 commands.

width(5),[2] x-axis labels 25 30 to 100, use actual counts rather than proportions (option **frequency**), more complete numbering of the y-axis, and an explanatory title for the histogram. Paste the resulting graph into your exercise.

- ✓ Using these two graphs, explain why the Civil Service Commission concluded that the exam was rigged.

2. Later on in the course, we're going to need to create "standardized" scores from raw scores. In the present case the *score* variable has a mean of 56.2 and a standard deviation of 17.6. We'll want to convert these raw scores to standardized scores that have a mean = zero and a standard deviation = 1.0. They're often called *zscores* because of their association with the standard normal distribution that we'll be encountering later.

- { So, let's create a standardized version of *score* by using the following command:

```
egen zscore=std(score)
```

The **egen** command creates a new variable *zscore* by standardizing *score.*

- { Next, let's list a few rows of data to see what the new *zscore* looks like in comparison with the raw scores.

```
list if (score <30 | score >86)
```

The **list** command is an enormously useful command. Here we've decided to list all those raw test scores that are below 30 and greater than 86. Notice that the "|" symbol stands for "or" in Stata's vocabulary, so this command asks Stata to list observations if the values of *score* are less than 30 *or* greater than 86. Be sure to study the section in Chapter 2 entitled "Specifying Subsets of the Data: `in` and `if` Qualifiers" in *Statistics with Stata 12* in order to learn more about how to use the **list** command.

- ✓ Paste the results of the **list** command into your exercise.

- ✓ Finally, **summarize** (with detail) the *zscore* variable and paste the result into your exercise. How do the two versions of *score* compare?

3. Let's categorize the *score* variable by assigning each test-taker a rating and creating a new variable to contain that numerical rating.

[2] See Hamilton, *Statistics with Stata Version 12*, Chapter 3 ("Graphs") Section "Histograms" for more information.

```
gen group=.
replace group=1 if score < 30
replace group=2  if (score >29 & score < 43)
replace group=3  if (score >42 & score <68)
replace group=4  if (score >67 & score <90)
replace group=5  if (score >89)
label values  group grouplbl
label define grouplbl 1 "Cretin" 2 "Poor" 3 "Average" 4 "Good" 5 "Crook"
graph box score, over(group)
```

The group of Stata commands above does the following things:

- The **generate** command creates a variable named *group* and sets all of its values equal to "missing" (.).

- The **replace** command replaces missing values in *group* with a "1" if the value of *score* is less than 30.

- The next **replace** command replaces missing values in *group* with a "2" if the value of *score* is greater than 42 and less than 68. (Why have I chosen the end points that I have in these **replace** commands?)

- Now, I label each value of variable *group*. The first **label** command sets up a location to store the labels for variable *group*. The second **label** command assigns a term to each value of *group*.

- Finally, create a set of boxplots for the *score* variable separated by values of the group variable.

✓ Paste the boxplot into your exercise.

- **Save** your dataset.

4. Now, let's get a better idea of the true scores of the civil service applicants by getting rid of the crooks.

- To do this, we can use the **drop** command as follows:

```
drop if group==5
```

The command tells Stata to **drop** any observation that has a *group* value equal to 5.

- Now, let's **list** the lowest and highest values again:

```
list if (score <30 | score >80)
```

- ✓ Paste the results of the **list** command into your exercise.

- { Finally let's reduce the data set even further by getting rid of the cretins:

```
drop if group==1

list if (score <35 | score >80)
```

- { Save the revised data set under a new name, say *chicago_purged*.

- ✓ Finally **summarize** *score* now that it's been purged of crooks and cretins, and paste your results into your exercise.

15.2.2 Numeric storage types

Numbers can be stored in one of five variable types: `byte`, `int`, `long`, `float` (the default), or `double`. `bytes` are, naturally, stored in 1 byte. `int`s are stored in 2 bytes, `long`s and `float`s in 4 bytes, and `double`s in 8 bytes. The table below shows the minimum and maximum values for each storage type and the encoding used internally by Stata to record *missing*.

Storage Type	Minimum	Maximum	Closest to 0 without being 0	missing
`byte`	−127	126	±1	127
`int`	−32,767	32,766	±1	32,767
`long`	−2,147,483,647	2,147,483,646	±1	2,147,483,647
`float`	-10^{36}	10^{36}	$\pm10^{-36}$	2^{128}
`double`	-10^{308}	10^{308}	$\pm10^{-323}$	2^{1023}

Do not confuse the term *integer*, which is a characteristic of a number, with `int`, which is a storage type. For instance, the number 5 is an integer no matter how it is stored; thus, if you read that an argument is required to be an integer, that does not mean that it must be stored as an `int`.

15.4.3 Strings containing numeric data

If a string variable contains the character representation of a number—for instance, `myvar` contains `"1"`, `"1.2"`, and `"-5.2"`—you can convert it directly into a numeric variable using the `real()` function, e.g., `generate newvar=real(myvar)`.

Similarly, if you want to convert a numeric variable to its string representation, you can use the `string()` function: `generate str10 as_str=string(newvar)`.

See [U] **16.3.5 String functions**.

15.4.4 String storage types

Strings are stored in string variables with storage types `str1`, `str2`, . . . , `str80`. The storage type merely sets the maximum length of the string, not its actual length; thus, `"example"` has length 7 whether it is stored as a `str7`, a `str10`, or even a `str80`. On the other hand, an attempt to assign the string `"example"` to a `str6` would result in `"exampl"`.

The maximum length of a string in Stata is 80 characters. String literals may exceed 80 characters, but only the first 80 characters are significant.

15.5 Formats: controlling how data are displayed

Formats describe how a number or string is to be presented. For instance, how is the number 325.24 to be presented? As `325.2`, or `325.24`, or `325.240`, or `3.2524e+2`, or `3.25e+2`, or how? The *display format* tells Stata exactly how you want this done. You do not have to specify display formats, since Stata always makes reasonable assumptions on how to display a variable, but you always have the option.