8

Stata Basics

This tutorial covers **how to**:

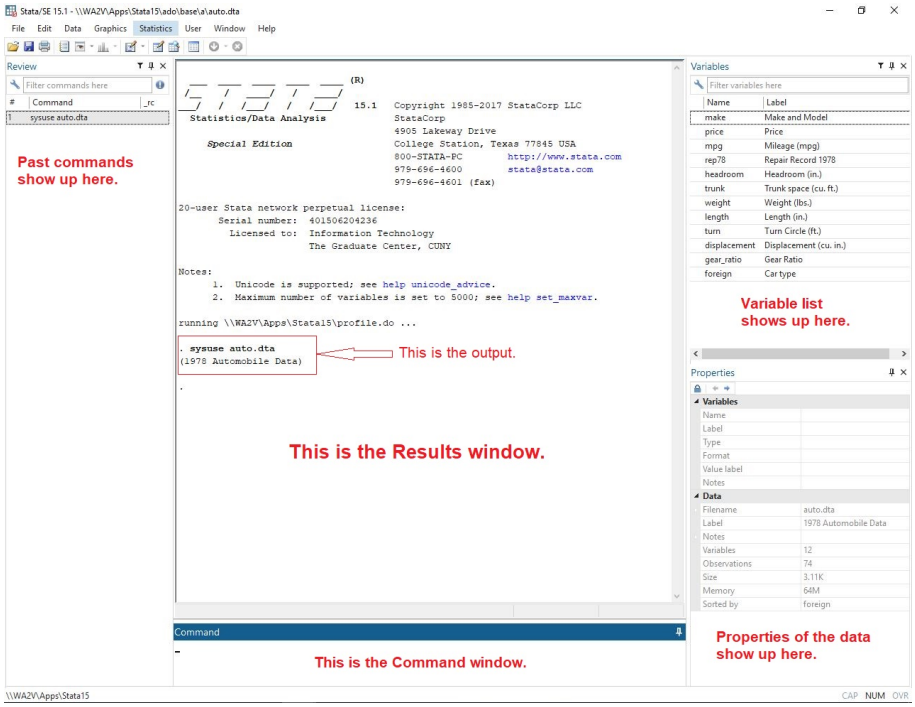
- describe the data
- produce summary statistics
- list a subset of observations
- create one-way and two-way frequency tables
- plot graphs
- run regressions
- generate variables

1 Loading a Data Set

Stata is one of the most commonly used software programs by researchers in Economics. This tutorial is going to introduce some basic features and commands in Stata. To get started, let's open `auto`, one of the example data sets that's part of the Stata installation. `auto` contains data on cars sold in the United States in 1978. You can load (or read) the `auto` data by typing

```
sysuse auto
```

in Stata's command window and pressing *enter*. (`sysuse` always loads one of Stata's example data sets; in this case, the example data set is `auto`.) Stata responds to this command by writing the following output to the results window:



You'll usually work with a data set that's not part of the Stata installation. For instance, you might have downloaded the `auto` data set from `http://www.stata-press.com/data/r9/auto.dta` and stored it in `D:\yourname\tutorial` on your computer. To load this file into Stata, first change Stata's working directory to that folder.

```
cd "d:\yourname\tutorial"
```

Second, instruct Stata to `use` the `auto` data.

```
use auto, clear
```

Stata looks for a file named `auto.dta` in its working directory and opens it.

2 Describing the Data

To reveal the contents (e.g., number of observations and variables, variable names) of the data set, type

```
describe
```

Stata writes the following output to the results window:

```
. describe

Contains data from \\WA2V\Apps\Stata15\ado\base/a/auto.dta
  obs:            74                          1978 Automobile Data
 vars:            12                          13 Apr 2016 17:45
 size:         3,182                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
rep78           int     %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           int     %8.0g                 Trunk space (cu. ft.)
weight          int     %8.0gc                Weight (lbs.)
length          int     %8.0g                 Length (in.)
turn            int     %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      float   %6.2f                 Gear Ratio
foreign         byte    %8.0g       origin    Car type
-------------------------------------------------------------------------------
Sorted by: foreign
```

The `auto` data contains 74 observations on 12 variables. For each variable, we see the variable name, storage type, display format, value label (although the value labels for most variables here are blank), and variable label. For example, the name of the first variable in the listing is `make`, a string variable, and its label is "Make and Model." The second variable is `price`, it's a numeric variable, and its label is "Price."

Stata stores variables as either string type or numeric type. String variables contain characters; numeric variables contain numbers. String variables' storage types are str#, such as str1, str2, str3, ..., str2045, or strL. Numeric variables' storage types are byte, int, long, float, or double. For more about data types, type `help data types`.

3 Producing Summary Statistics

To present the summary statistics of all the variables in the data set, submit

```
summarize
```

The following table appears in the results window:

```
. summarize

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
        make |          0
       price |         74    6165.257    2949.496       3291      15906
         mpg |         74     21.2973    5.785503         12         41
       rep78 |         69    3.405797    .9899323          1          5
    headroom |         74    2.993243    .8459948        1.5          5
-------------+---------------------------------------------------------
       trunk |         74    13.75676    4.277404          5         23
      weight |         74    3019.459    777.1936       1760       4840
      length |         74    187.9324    22.26634        142        233
        turn |         74    39.64865    4.399354         31         51
displacement |         74    197.2973    91.83722         79        425
-------------+---------------------------------------------------------
  gear_ratio |         74    3.014865    .4562871       2.19       3.89
     foreign |         74    .2972973    .4601885          0          1
```

Notice that the variable `rep78` has fewer observations than other variables; `rep78` has some missing observations. Also, the variable `make` doesn't have any observations to compute summary statistics because it's a string (rather than numeric) variable.

4 Listing a Subset of Observations

Which makes of cars have missing repair records? To answer this question, type

```
list make if missing(rep78)
```

And Stata replies with

```
. list make if missing(rep78)

     +-------------+
     |        make |
     |-------------|
  3. | AMC Spirit  |
  7. | Buick Opel  |
 45. | Plym. Sapporo |
 51. | Pont. Phoenix |
 64. | Peugeot 604 |
     +-------------+
```

The `if` qualifier allows us to look at a subset of observations.

We could also list the first few observations using the `in` qualifier. For instance, to list the first 5 observations, type

```
list in 1/5
```

And the output is

```
. list in 1/5

     +-----------------------------------------------------------------------------------------------+
     |        make | price | mpg | rep78 | headroom | trunk | weight | length | turn | displa~t | gear_r~o | foreign |
     |-------------|-------|-----|-------|----------|-------|--------|--------|------|----------|----------|---------|
  1. | AMC Concord | 4,099 |  22 |     3 |      2.5 |    11 |  2,930 |    186 |   40 |      121 |     3.58 | Domestic |
  2. | AMC Pacer   | 4,749 |  17 |     3 |      3.0 |    11 |  3,350 |    173 |   40 |      258 |     2.53 | Domestic |
  3. | AMC Spirit  | 3,799 |  22 |     . |      3.0 |    12 |  2,640 |    168 |   35 |      121 |     3.08 | Domestic |
  4. | Buick Century | 4,816 |  20 |     3 |      4.5 |    16 |  3,250 |    196 |   40 |      196 |     2.93 | Domestic |
  5. | Buick Electra | 7,827 |  15 |     4 |      4.0 |    20 |  4,080 |    222 |   43 |      350 |     2.41 | Domestic |
     +-----------------------------------------------------------------------------------------------+
```

To list the makes of the first 3 observations, submit

```
list make in 1/3
```

And the output is

```
. list make in 1/3

     +--------------+
     |         make |
     |--------------|
  1. | AMC Concord  |
  2. | AMC Pacer    |
  3. | AMC Spirit   |
     +--------------+
```

5 Creating One-Way and Two-Way Frequency Tables

To produce a one-way frequency table for any variable in the data set, type `tabulate varname`. For example, type

```
tabulate foreign
```

And the output is

```
. tabulate foreign

   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
   Domestic |         52       70.27       70.27
    Foreign |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00
```

Of the 74 cars in the data, 52 are domestic, and 22 are foreign. Nearly 30% of the cars are foreign imports.

A one-way frequency tabulation of repair records `rep78` is as follows:

```
. tabulate rep78

     Repair |
Record 1978 |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |          2        2.90        2.90
          2 |          8       11.59       14.49
          3 |         30       43.48       57.97
          4 |         18       26.09       84.06
          5 |         11       15.94      100.00
------------+-----------------------------------
      Total |         69      100.00
```

Repair records are coded from 1 to 5, and higher numbers associate with better repair records. The repair records of 5 cars are missing.

To cross-tabulate two variables, `foreign` and `rep78`, submit

```
tabulate rep78 foreign, row
```

And the output is

```
. tabulate rep78 foreign, row
```

Key
frequency *row percentage*

Repair Record 1978	Car type		
	Domestic	Foreign	Total
1	2	0	2
	100.00	0.00	100.00
2	8	0	8
	100.00	0.00	100.00
3	27	3	30
	90.00	10.00	100.00
4	9	9	18
	50.00	50.00	100.00
5	2	9	11
	18.18	81.82	100.00
Total	48	21	69
	69.57	30.43	100.00

Here `rep78` is the row variable, `foreign` is the column variable, and the table displays within-row relative frequencies since the command includes the `row` option.

To produce cross-tabulations with `foreign` as the row variable and `rep78` as the column variable (and displaying the within-column relative frequencies), switch the order of the two variables and use the `column` option.

```
tabulate foreign rep78, column
```

The resulting output is

```
. tabulate foreign rep78, column
```

Key
frequency *column percentage*

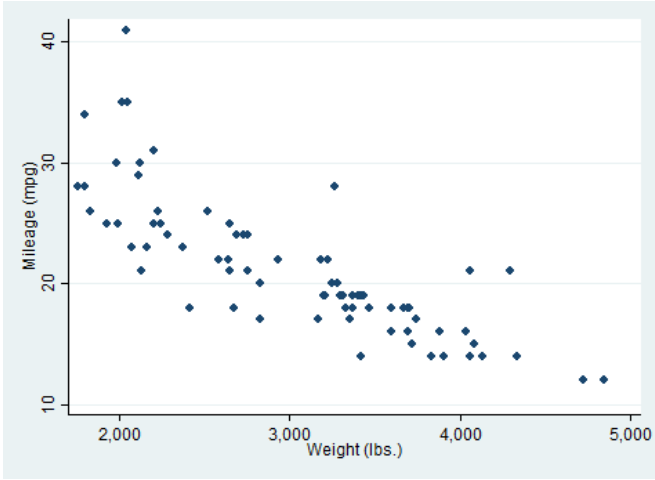
	Repair Record 1978					
Car type	1	2	3	4	5	Total
Domestic	2	8	27	9	2	48
	100.00	100.00	90.00	50.00	18.18	69.57
Foreign	0	0	3	9	9	21
	0.00	0.00	10.00	50.00	81.82	30.43
Total	2	8	30	18	11	69
	100.00	100.00	100.00	100.00	100.00	100.00

6 Plotting the Data

To plot the data for two variables, `mpg` and `weight`, type

```
scatter mpg weight
```

Stata displays the following graph:



The variables `mpg` and `weight` seem to be negatively related, and the relationship might be nonlinear.

7 Running Regressions

To find factors that influence a car's fuel efficiency, we regress `mpg` on variables like `weight`. Since the relationship between `mpg` and `weight` appears to be nonlinear, let's model `mpg` as a quadratic function of `weight`. To create the quadratic term, submit

```
generate wtsq = weight^2
```

which is `weight` squared.

To fit the quadratic relationship between `mpg` and `weight`, type

```
regress mpg weight wtsq
```

The regression output is

```
. regress mpg weight wtsq
```

Source	SS	df	MS	Number of obs	=	74
				F(2, 71)	=	72.80
Model	1642.52197	2	821.260986	Prob > F	=	0.0000
Residual	800.937487	71	11.2808097	R-squared	=	0.6722
				Adj R-squared	=	0.6630
Total	2443.45946	73	33.4720474	Root MSE	=	3.3587

mpg	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
weight	-.0141581	.0038835	-3.65	0.001	-.0219016	-.0064145
wtsq	1.32e-06	6.26e-07	2.12	0.038	7.67e-08	2.57e-06
_cons	51.18308	5.767884	8.87	0.000	39.68225	62.68392

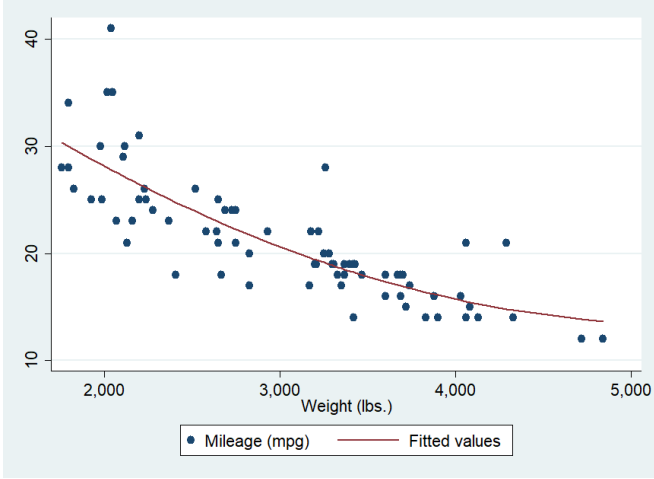
To add the predicted values of `mpg` to the scatterplot, first create the predicted values.

```
predict mpghat
```

Next graph the data with the predicted values by typing

```
twoway (scatter mpg weight) (line mpghat weight, sort)
```

Using the `sort` option ensures that the line segments linking the predicted values connect adjacent points from smallest to largest values of `weight`.



8 Generating Variables

Recall that the variable `length` is in inches. To create a new variable that has the length in feet, type

```
generate length_ft = length/12
```

To create a new variable that is the natural log of `length`, type

```
generate llength = log(length)
```

Suppose we want to break `weight` into three categories.

```
generate weight3 = .
replace weight3 = 1 if weight <= 2000
replace weight3 = 2 if weight > 2000 & weight <= 4000
replace weight3 = 3 if weight > 4000 & weight < .
```

Alternatively, we could convert `weight` into three categories using `recode`.

```
generate weight3a = weight
recode weight3a (min/2000=1) (2001/4000=2) (4001/max=3)
tab weight3 weight3a
```

```
. gen weight3 = .
(74 missing values generated)

. replace weight3 = 1 if weight <= 2000
(7 real changes made)

. replace weight3 = 2 if weight > 2000 & weight <= 4000
(58 real changes made)

. replace weight3 = 3 if weight > 4000 & weight < .
(9 real changes made)

. gen weight3a = weight

. recode weight3a (min/2000=1) (2001/4000=2) (4001/max=3)
(weight3a: 74 changes made)
```

To confirm that the two methods are equivalent, check the cross tabulations.

```
. tab weight3 weight3a
```

	weight3a			
weight3	1	2	3	Total
1	7	0	0	7
2	0	58	0	58
3	0	0	9	9
Total	7	58	9	74

To regress `mpg` on the weight categories, submit

```
regress mpg i.weight3
```

```
. regress mpg i.weight3
```

Source	SS	df	MS	Number of obs	=	74
				F(2, 71)	=	12.16
Model	623.357927	2	311.678963	Prob > F	=	0.0000
Residual	1820.10153	71	25.6352329	R-squared	=	0.2551
				Adj R-squared	=	0.2341
Total	2443.45946	73	33.4720474	Root MSE	=	5.0631

mpg	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
weight3						
2	-6.603448	2.025873	-3.26	0.002	-10.64293	-2.563972
3	-12.55556	2.551575	-4.92	0.000	-17.64325	-7.467859
_cons	28	1.913681	14.63	0.000	24.18423	31.81577

The indicator prefix "i" instructs Stata to create one dummy variable for each value of `weight`. By default, `weight3==1` is the reference group, and the two dummy variables are indicators of `weight3==2` and `weight3==3`.

Summary List of Commands

```
sysuse auto.dta
describe
summarize
list make if missing(rep78)
list in 1/5
list make in 1/3
tabulate foreign
tabulate rep78 foreign, row
tabulate foreign rep78, column
scatter mpg weight
generate wtsq = weight^2
regress mpg weight wtsq
predict mpghat
twoway (scatter mpg weight) (line mpghat weight, sort)
```

Additional Resources

Introducing Stata sample session
`https://www.stata.com/manuals/gsw1.pdf`

Opening and saving Stata data sets
`https://www.stata.com/manuals/gsw5.pdf`

Creating dummy variables
`https://www.stata.com/support/faqs/data-management/creating-dummy-variables/`

Creating and recoding variables
`https://stats.idre.ucla.edu/stata/modules/creating-and-recoding-variables/`

Acknowledgements

Shaoying Ma created this and other tutorials in the *Math for Economics* series. Kenneth McLaughlin supervised her work and edited the final product.