Stata How-to:
Explore and Describe Data

Patrick Blanchenay

2021-12-27

Contents

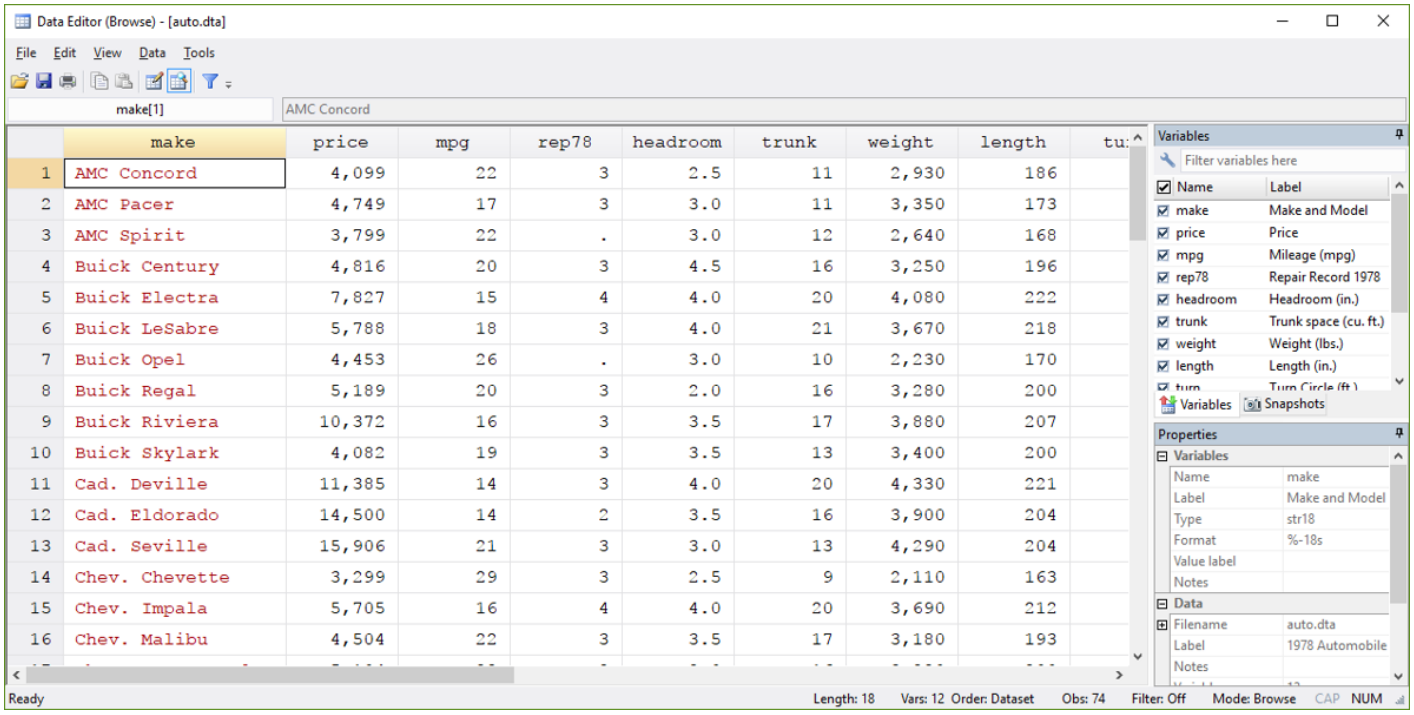
1. Browsing the dataset

Let's use the system dataset `auto`:

```
sysuse auto, clear
```

The data in memory can be viewed/browsed in a spreadsheet-like data browser by clicking on the data viewer button  or by using the **`browse`** command in the Command window.[1] A window will open:

[1] **`browse`** is one of the few commands that is more useful in the Command window than it would be in a do-file.



This is often useful to check that data manipulating commands had the effect intended. It also possible to browse only a few variables by specifying them with the **browse** command:

```
browse price mpg weight // browsing a small set of variables
```

In the data browser, colours represent different types of variables:

- black for numeric variables;
- red for string variables;
- blue for categorical variables, numeric variables that represent labelled categories (read section 4)

The data can be sorted using the drop down menus in the window above or using the **sort** or **gsort** commands:

```
sort price              //sorts the data by ascending price
gsort -mpg price        //sorts the data by descending mpg, then by ascending price
```

2. Missing values

Sometimes datasets have missing values for some variables under some observations. Stata records missing values in numeric variables as **.** (a dot) and missing values in string variables as "" (empty string). Some commands automatically ignore observations with missing values in variables listed in the command (e.g. the **tabulate** command excludes them by default and the **regress** command drops them from the regression input data). Some commands such as **tabulate** have options that allow you to choose how Stata handles these values.

> **Warning**
>
> In comparisons, Stata treats missing numeric values **.** as infinitely large numbers when comparing values. See Stata How-to: Conditions, Subsetting for more details.

3. Exploring the data

3.1. What is in the dataset: **describe**

The **describe** command outputs information about the dataset such as the number of observations, as well as information on the variables and their formats. Stata variables can either be numeric or strings (can only take on text values). The command can be executed as follows:

```
// Describe the dataset in memory
describe
```

which produces:

```
  obs:            74                          1978 Automobile Data
 vars:            12                          13 Apr 2014 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 output tells us, among other things, that the dataset has 74 observations, that the "make" variable is a str18 format (18 character string), and that the remaining variables are numeric (the int and byte formats both hold whole numbers but the byte format takes on a much smaller range of values; float and double formats both hold decimal numbers although doubles have greater precision). We are also given the variable labels and the display format (this specifies how many digits should be displayed, whether thousands should be separated by commas, etc.).

3.2. What information is in these variables: **codebook**

The **codebook** command lists one (if specified) or all variables of the dataset, and offers important information for each variable such as: the range of values, the number of unique values, the number of missing values.

```
price                                                                 Price
-------------------------------------------------------------------------------

            type:  numeric (int)

           range:  [3291,15906]                 units:  1
   unique values:  74                       missing .:  0/74

            mean:   6165.26
       std. dev:   2949.5

     percentiles:        10%       25%       50%       75%       90%
                        3895      4195    5006.5      6342     11385
```

3.3. Summary statistics of a variable: **summarize**

The **summarize** command outputs summary statistics for the specified *numeric* variables:

```
//print summary statistics for several variables
summarize price mpg weight
```

Variable	Obs	Mean	Std. Dev.	Min	Max
price	74	6165.257	2949.496	3291	15906
mpg	74	21.2973	5.785503	12	41
weight	74	3019.459	777.1936	1760	4840

By default, **summarize** provides the mean, standard deviation, minimum and maximum value. When using the **detail** option, **summarize** also provides variance, skewness, various percentiles, as well as the 4 smallest and 4 largest values that the variable takes (useful to spot outliers):

```
// Print more detailed summary statistics for the price variable
summarize price, detail
```

```
                            Price
-------------------------------------------------------------
      Percentiles      Smallest
 1%          3291          3291
 5%          3748          3299
10%          3895          3667       Obs                  74
25%          4195          3748       Sum of Wgt.          74

50%        5006.5                     Mean           6165.257
                        Largest       Std. Dev.      2949.496
75%          6342         13466
90%         11385         13594       Variance       8699526
95%         13466         14500       Skewness       1.653434
99%         15906         15906       Kurtosis       4.819188
```

3.4. How many observations take such and such value: **tabulate**

The **tabulate** command produces tabulations and cross-tabulations for categorical variables. That is, it displays for each unique value that the variable takes, the number of observations that take that value:

```
// Tabulate the foreign variable
tabulate foreign
```

Car type	Freq.	Percent	Cum.
Domestic	52	70.27	70.27
Foreign	22	29.73	100.00
Total	74	100.00	

This tells us that in our dataset, 52 cars are domestic (US) cars, and 22 are foreign cars.

We can also perform cross-tabulations, that is, do the same but for two variables at once. The table displays the number of observations that take each combination of the two variables. Missing values can be included in the table by adding the **`missing`** option:

```
// Cross-tabulate rep78 vs. foreign, with missing values included
tabulate rep78 foreign, missing
```

Repair Record 1978	Car type		Total
	Domestic	Foreign	
1	2	0	2
2	8	0	8
3	27	3	30
4	9	9	18
5	2	9	11
.	4	1	5
Total	52	22	74

which tells us that among domestic cars, 27 had 3 repair records, and 4 do not have any information recorded about repair.

4. Confusing: Categorical variables and value labels

Categorical variables are numeric variables used to record labelled categories; they are a common source of confusion for students. Suppose for instance that you have dataset that records the Canadian province where individuals live. This is categorical information. You *could* have a string variable called `province`, and for each individual, the string would be "Alberta" or "Ontario" or "British Columbia", etc. depending on where they live. But string variables take a lot of space in your dataset.

To be more memory efficient, it is common for such categorical information to be stored numerically: as an integer, for instance, 1 for Alberta, 2 for British Columbia, etc. The problem is that it would be hard for researchers to remember that 1 is for Alberta, 2 is for BC... To remember this, Stata uses *value labels*, which provides meaningful labels to numeric variables. Such labelled variables appear in blue when browsing the dataset. A value label is essentially correspondence table between the numeric values of a variable, and easy-to-understand labels.

Let's look at an example from the system dataset `nlsw88`, an extract from a labour survey on young women. The dataset contains individual characteristics (such as education, whether they have been married) and labour market outcomes for those indivduals (their hourly wage, whether they are unionized, etc.).

```
sysuse nlsw88, clear // loads the NLSW 1988 extract
```

In this dataset, the variable `industry` records the industry for which the individuals work. When browsing, you will industry explicitly listed, such as "Professional Services" or "Construction". But underneath these are numeric values.

How to know what numeric value correspond to what label?

Typing **`label list`** will list all value labels (correspondence tables) used in the dataset, and for each the values and their corresponding label:

```
label list
```

The output can be long, but you will see that one of these value labels is called `indlb`, and this gives us our answer:

```
indlbl:
           1 Ag/Forestry/Fisheries
           2 Mining
           3 Construction
           4 Manufacturing
           5 Transport/Comm/Utility
           6 Wholesale/Retail Trade
           7 Finance/Ins/Real Estate
           8 Business/Repair Svc
           9 Personal Services
          10 Entertainment/Rec Svc
          11 Professional Services
          12 Public Administration
```

From this we see that "Construction" is coded as **`industry==3`** and that "Professional Services" is coded as **`industry==11`**.

> **Warning**
>
> Categorical variables appear to be string variables but are actually numeric variables. In the example above, to filter observations that work in Professional Services, one would write: **`browse if (industry==11)`**. Writing **`browse if (industry == "Professional Services")`** would generate an error.

5. Flexible summary tables using `table`

The **`table`** command can be used to create more complex tables of summary statistics. The general syntax is

```
  table [row variable] [optional column variable], [contents of each cell]
```

To practice this, we can use the system dataset `nlsw88`, an extract from a labour survey on young women. The dataset contains individual characteristics (such as education, whether they have been married) and labour market outcomes for those indivduals (their hourly wage, whether they are unionized, etc.). Suppose we want to create a table with mean hourly wages (`wage`) for workers who have a college degree versus those who do not (dummy `collgrad`):

```
sysuse nlsw88, clear // loads the NLSW 1988 extract
table collgrad, contents(mean wage) // average wage, college graduates vs. not
```

displays a table reporting, for each value of `collgrad`, the mean of `wage`; so we will obtain the mean wage for women where `collgrad == 1` (college graduates), and the mean wage for women where `collgrad == 0` (non college graduates).

college graduate	mean(wage)
not college grad	6.910561
college grad	10.52606

Note that the variable that defines rows comes first, while the content of the table comes as part of the **`contents()`** option. To look at average wages based on the industry these women work in:

```
table industry, contents(mean wage) // average wage by industry
```

industry	mean(wage)
Ag/Forestry/Fisheries	5.621121
Mining	15.34959
Construction	7.564934
Manufacturing	7.501578
Transport/Comm/Utility	11.44335
Wholesale/Retail Trade	6.125896
Finance/Ins/Real Estate	9.843174
Business/Repair Svc	7.51579
Personal Services	4.401093
Entertainment/Rec Svc	6.724409
Professional Services	7.871186
Public Administration	9.148407

In addition to rows defined by a variable, we can also have a variable for columns. For instance, to get mean wages by industry, depending on whether they are college graduate or not:

```
table industry collgrad, contents(mean wage) // average wage by industry, college graduates vs. not
```

	college graduate	
industry	not college grad	college grad
Ag/Forestry/Fisheries	5.312225	7.062636
Mining	15.34959	
Construction	6.399225	17.66774
Manufacturing	6.88612	13.73076
Transport/Comm/Utility	11.19514	13.22596
Wholesale/Retail Trade	5.852632	8.247295
Finance/Ins/Real Estate	9.496175	11.8756
Business/Repair Svc	6.47788	11.72848
Personal Services	4.360305	5.679114
Entertainment/Rec Svc	5.970389	8.534057
Professional Services	6.270573	10.09349
Public Administration	8.213759	12.03929

Other summary statistics than the mean can also be obtained and combined in the table: the **`contents()`** should contain the statistic followed by the variable, for as many times as cells of info you want.[2]

[2] See **`help table`** for an exhaustive list of statistics that can be computed.

```
table industry, contents(mean wage p50 wage sd wage) // avg, median and std. dev. of wage, by industry
```

industry	mean(wage)	med(wage)	sd(wage)
Ag/Forestry/Fisheries	5.621121	4.53301	3.226642
Mining	15.34959	8.091784	16.67932
Construction	7.564934	6.688963	5.095233
Manufacturing	7.501578	6.191625	5.368414
Transport/Comm/Utility	11.44335	10.1248	6.127745
Wholesale/Retail Trade	6.125896	4.53301	5.510996
Finance/Ins/Real Estate	9.843174	7.053139	8.334663
Business/Repair Svc	7.51579	5.334139	6.44801
Personal Services	4.401093	3.887958	2.607814
Entertainment/Rec Svc	6.724409	4.227053	4.072037
Professional Services	7.871186	6.69887	5.114011
Public Administration	9.148407	8.397742	5.030466

See also Stata How-to: Conditions, Subsetting on how to count observations that satisfy a certain condition.

6. Correlations between variables

The **correlate** command outputs a table of pairwise correlations for specified variables:

```
sysuse auto, clears
correlate price mpg weight
```

	price	mpg	weight
price	1.0000		
mpg	-0.4686	1.0000	
weight	0.5386	-0.8072	1.0000

For a more substantive analysis of linear relationships between variables, economists use linear regressions. See Stata How-to: OLS regressions .

7. Practice

Create a folder on your computer, and save the dataset `CPS_2016.dta` to that folder. Open Stata, and create a new do-file in that folder. Use that do-file to answer the questions below; ensure that your do-file can always run from the beginning without generating errors.

1. ☐ At the top, write a comment to describe what this do-file does.
2. ☐ Set the working directory to the folder you just created, and load the dataset.
3. ☐ For how many observations is education level missing?
4. ☐ What is the average and median wage in the dataset?
5. ☐ How many individuals are Divorced? How is that marital status coded?
6. ☐ Create a table that provides the average age by marital status.
7. ☐ Among what marital status is the difference in wages between men and women the biggest? Create a summary table to answers.