Stata Learning Modules

Ethan Deng

Version 1.0

April 20, 2015

1 Fundamentals of Using Stata (part I)

1.1 A Sample Stata Session

Please refer to Manuals13.

1.2 Descriptive information and statistics

This module shows common commands for showing descriptive information and descriptive statistics about data files.

1.2.1 Getting an overview of your file

The sysuse command loads a specified Stata-format dataset that was shipped with Stata. Here we will use the auto data file.

```
sysuse auto
```

The describe command shows you basic information about a Stata data file. As you can see, it tells us the number of observations in the file, the number of variables, the names of the variables, and more.

```
describe
 Contains data from auto.dta
  obs:            74
 vars:            12                          17 Feb 1999 10:49
 size:         3,108 (99.6% of memory free)
-------------------------------------------------------------------------------
   1. make      str17   %17s
   2. price     int     %9.0g
   3. mpg       byte    %9.0g
   4. rep78     byte    %9.0g
   5. hdroom    float   %9.0g
   6. trunk     byte    %9.0g
   7. weight    int     %9.0g
   8. length    int     %9.0g
   9. turn      byte    %9.0g
  10. displ     int     %9.0g
  11. gratio    float   %9.0g
  12. foreign   byte    %9.0g
-------------------------------------------------------------------------------
Sorted by:
```

The codebook command is a great tool for getting a quick overview of the variables in the data file. It produces a kind of electronic codebook from the data file. Have a look at what it produces below.

```
codebook
make --------------------------------------------------------------- (unlabeled)
                  type:  string (str17)

         unique values:  74                   coded missing:  0 / 74

              examples:  "Cad. Deville"
                         "Dodge Magnum"
                         "Merc. XR-7"
                         "Pont. Catalina"

               warning:  variable has embedded blanks

price -------------------------------------------------------------- (unlabeled)
                  type:  numeric (int)

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

                  mean:   6165.26
              std. dev:    2949.5

           percentiles:        10%       25%       50%       75%       90%
                              3895      4195    5006.5      6342     11385
//(omitted)
```

Another useful command for getting a quick overview of a data file is the inspect command. Here is what the inspect command produces for the auto data file.

```
inspect
price:                                          Number of Observations
--------                                                            Non-
                                                Total  Integers     Integers
|  #                            Negative            -         -          -
|  #                            Zero                -         -          -
|  #                            Positive           74        74          -
|  #                                            -----     -----      -----
|  #                            Total              74        74          -
|  #   #   .   .   .            Missing             -
+----------------------                         -----
3291              15906                            74
  (74 unique values)

mpg:                                            Number of Observations
------                                                              Non-
                                                Total  Integers     Integers
|      #                        Negative            -         -          -
|      #                        Zero                -         -          -
|      #                        Positive           74        74          -
|  #   #                                        -----     -----      -----
|  #   #   #                    Total              74        74          -
|  #   #   #   #   .            Missing             -
+----------------------                         -----
12                   41                            74
```

```
  (21 unique values)
//(omitted)
```

The `list` command is useful for viewing all or a range of observations. Here we look at *make*, *price*, *mpg*, *rep78* and *foreign* for the first 10 observations.

```
list make price mpg rep78 foreign in 1/10
                   make      price        mpg      rep78     foreign
  1.      Dodge Magnum       5886        16          2           0
  2.        Datsun 510       5079        24          4           1
  3.      Ford Mustang       4187        21          3           0
  4.  Linc. Versailles      13466        14          3           0
  5.     Plym. Sapporo       6486        26          .           0
  6.       Plym. Arrow       4647        28          3           0
  7.     Cad. Eldorado      14500        14          2           0
  8.        AMC Spirit       3799        22          .           0
  9.    Pont. Catalina       5798        18          4           0
 10.        Chev. Nova       3955        19          3           0
```

1.2.2 Creating tables

The `tabulate` command is useful for obtaining frequency tables. Below, we make a table for *rep78* and a table for *foreign*. The command can also be shortened to `tab`.

```
tabulate rep78
      rep78 |      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
tabulate foreign
     foreign |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         52       70.27       70.27
          1 |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00
```

The `tab1` command can be used as a shortcut to request tables for a series of variables (instead of typing the `tabulate` command over and over again for each variable of interest).

```
tab1 rep78 foreign
-> tabulation of rep78

      rep78 |      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

-> tabulation of foreign

    foreign |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         52       70.27       70.27
          1 |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00
```

We can use the `plot` option to make a plot to visually show the tabulated values.

```
tabulate rep78, plot
      rep78 |      Freq.
------------+------------+-----------------------------------------------------
          1 |          2 |**
          2 |          8 |********
          3 |         30 |******************************
          4 |         18 |******************
          5 |         11 |***********
------------+------------+-----------------------------------------------------
      Total |         69
```

We can also make crosstabs using `tabulate`. Let's look at the repair history broken down by *foreign* and *domestic* cars.

```
tabulate rep78 foreign
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         1 |         2          0 |         2
         2 |         8          0 |         8
         3 |        27          3 |        30
         4 |         9          9 |        18
         5 |         2          9 |        11
-----------+----------------------+----------
     Total |        48         21 |        69
```

With the `column` option, we can request column percentages. Notice that about 86% of the foreign cars received a rating of 4 or 5. Only about 23% of domestic cars were rated that highly.

```
tabulate rep78 foreign, column
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         1 |         2          0 |         2
           |      4.17       0.00 |      2.90
-----------+----------------------+----------
         2 |         8          0 |         8
           |     16.67       0.00 |     11.59
-----------+----------------------+----------
         3 |        27          3 |        30
           |     56.25      14.29 |     43.48
-----------+----------------------+----------
         4 |         9          9 |        18
           |     18.75      42.86 |     26.09
-----------+----------------------+----------
```

```
         5 |         2          9 |        11
           |      4.17      42.86 |     15.94
-----------+----------------------+----------
     Total |        48         21 |        69
           |    100.00     100.00 |    100.00
```

We can use the `nofreq` option to suppress the frequencies, and just focus on the percentages.

```
tabulate rep78 foreign, column nofreq
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         1 |      4.17       0.00 |      2.90
         2 |     16.67       0.00 |     11.59
         3 |     56.25      14.29 |     43.48
         4 |     18.75      42.86 |     26.09
         5 |      4.17      42.86 |     15.94
-----------+----------------------+----------
     Total |    100.00     100.00 |    100.00
```

Note that the order of the options does not matter. Just remember that the options must come after the comma.

```
tabulate rep78 foreign, nofreq column
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         1 |      4.17       0.00 |      2.90
         2 |     16.67       0.00 |     11.59
         3 |     56.25      14.29 |     43.48
         4 |     18.75      42.86 |     26.09
         5 |      4.17      42.86 |     15.94
-----------+----------------------+----------
     Total |    100.00     100.00 |    100.00
```

1.2.3 Generating summary statistics with summarize

For summary statistics, we can use the `summarize` command. Let's generate some summary statistics on *mpg*.

```
summarize mpg
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
     mpg |      74     21.2973   5.785503         12         41
```

We can use the `detail` option of the `summarize` command to get more detailed summary statistics.

```
summarize mpg, detail
                              mpg
-------------------------------------------------------------
      Percentiles      Smallest
 1%           12             12
 5%           14             12
10%           14             14       Obs                  74
25%           18             14       Sum of Wgt.          74

50%           20                      Mean            21.2973
                        Largest       Std. Dev.      5.785503
75%           25             34
90%           29             35       Variance       33.47205
```

```
95%          34             35        Skewness      .9487176
99%          41             41        Kurtosis      3.975005
```

To get these values separately for *foreign* and *domestic*, we could use the `by foreign:` prefix as shown below. Note that we first had to `sort` the data before using `by foreign:`.

```
sort foreign
by foreign: summarize mpg
 -> foreign= 0
Variable |    Obs        Mean    Std. Dev.       Min        Max
---------+---------------------------------------------------
     mpg |     52    19.82692    4.743297         12         34

-> foreign= 1
Variable |    Obs        Mean    Std. Dev.       Min        Max
---------+---------------------------------------------------
     mpg |     22    24.77273    6.611187         14         41
```

This is not the most efficient way to do this. Another way, which does not require the data to be sorted, is by using the `summarize( )` option as part of the `tabulate` command.

```
tabulate foreign, summarize(mpg)
            |          Summary of mpg
    foreign |       Mean   Std. Dev.       Freq.
------------+----------------------------------
          0 |  19.826923   4.7432972          52
          1 |  24.772727   6.6111869          22
------------+----------------------------------
      Total |  21.297297   5.7855032          74
```

Here is another example, showing the average price of cars for each level of repair history.

```
tabulate rep78, summarize(price)
            |         Summary of price
      rep78 |       Mean   Std. Dev.       Freq.
------------+----------------------------------
          1 |     4564.5   522.55191           2
          2 |   5967.625   3579.3568           8
          3 |  6429.2333   3525.1398          30
          4 |     6071.5   1709.6083          18
          5 |       5913   2615.7628          11
------------+----------------------------------
      Total |  6146.0435   2912.4403          69
```

1.2.4 Summary

- `describe`: provide information about the current data file, including the number of variables and observations and a listing of the variables in a data file.
- `codebook`: produce codebook like information for the current data file.
- `inspect`: provide a quick overview of data file.
- `list make mpg`: list out the variables make and mpg.
- `tabulate mpg`: make a table of mpg.
- `tabulate rep78 foreign`: make a two way table of rep78 by foreign.
- `summarize mpg price`: produce summary statistics of mpg and price.
- To produce summary statistics for mpg separately for foreign and domestic cars,use

```
sort foreign
by foreign: summarize(mpg)
```

- `tabulate foreign, summarize(mpg)`: produce summary statistics for mpg by foreign (prior sorting not required).

1.3 Getting help using Stata

This module shows resources you can use to help you learn and use Stata.

1.3.1 Stata online help

When you know the name of the command you want to use (e.g., `summarize`), you can use the Stata help to get a quick summary of the command and its syntax. You can do this in two ways:

1. type `help summarize` in the command window, or
2. click **Help**, **Stata Command**, then type `summarize`.

Here is what help summarize looks like.

```
help summarize
help summarize                                     dialog:  summarize
----------------------------------------------------------------------

Title

    [R] summarize -- Summary statistics


Syntax

        summarize [varlist] [if] [in] [weight] [, options]

    options            description
    ----------------------------------------------------------------
    Main
      detail           display additional statistics
      meanonly         suppress the display; only calculate the
                         mean; programmer's option
      format           use variable's display format
      separator(#)     draw separator line after every # variables;
                         default is separator(5)
    ----------------------------------------------------------------
    varlist may contain time-series operators; see tsvarlist.
    by may be used with summarize; see by.
    aweights, fweights, and iweights are allowed.  However,
      iweights may not be used with the detail option; see weight.
//(omitted)
```

If you use the pull-down menu to get help for a command, it shows the same basic information but related commands and topics are hotlinks you can click.

When you want to search for a keyword, e.g. `memory`, you can use Stata to search for help topics that contain that keyword. You can do this in two ways:

1. Type search `memory` in the command window, or
2. Click **Help**, **Search**, then **memory**.

Here is what search memory looks like.

```
search memory

GS       . . . . . . . . . . . . . . . . . . . . . . . . Getting Started manual

[U]      Chapter 7  . . . . . . . . . . . . . . . .  Setting the size of memory
         (help memory)

[R]      compress . . . . . . . . . . . . . . . . . . . . Compress data in memory
         (help compress)
//(omitted)
```

As you can see, there are lots of help topics that refer to memory. Some of the topics give you a command, and then you can get help for that command. Notice that those topics start with **GS[U]** or **[R]**. Those are indicating which Stata manual you could find the command (GS=Getting Started, U=Users Guide, R=Reference Guide).

The next set of topics all start with **FAQ** because these are Frequently Asked Questions from the Stata web site. You can see the title of the FAQ and the address of the FAQ. Lastly, there is a topic that starts with **STB** which stands for Stata Technical Bulletin. These refer to add-on programs that you can install into Stata. There are dozens, if not hundreds of specialized and useful programs that you can get from the Stata Technical Bulletin.

You can access this same kind of help from the pull-down menus by clicking **Help** then **Search** then type `memory`. Note how the related commands, the FAQs, and the STB all have hotlinks you can click. For example, you can click on a FAQ and it will bring up that FAQ in your web browser. Or, you could click on an STB and it would walk you through the steps of installing that STB into your copy of Stata. As you can see, there are real advantages to using the pull-down menus for getting help because it is so easy to click on the related topics.

1.3.2 Stata sample data files

Stata has some very useful data files available to you for learning and practicing Stata. For example, you can type

```
sysuse auto
```

to use the auto data file that comes with Stata. You can type

```
sysuse dir
```

to see the entire list of data files that ship with Stata. You can type

```
help dta_contents
```

to see all of the sample data files that you can easily access from within Stata.

1.3.3 Stata web pages

The Stata web page is a wonderful resource. You can visit the main page at http://www.stata.com.

The User Support page (click **User Support** from main page) has a great set of resources, including

- FAQs
- NetCourses
- StataList: How to subscribe
- StataList: Archives
- Statalist ado-file Archives
- Stata Bookstore

In the bookstore, you can find books on Stata. A good intro book on Stata is **Statistics with Stata**.

2 Fundamentals of Using Stata (part II)

2.1 Using IF with Stata commands

This module shows the use of if with common Stata commands.

Let's use the auto data file.

```
sysuse auto
```

For this module, we will focus on the variables *make*, *rep78*, *foreign*, *mpg*, and *price*. We can use the keep command to keep just these five variables.

```
keep make rep78 foreign mpg price
```

Let's make a table of *rep78* by *foreign* to look at the repair histories of the foreign and domestic cars.

```
tabulate rep78 foreign
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         1 |         2          0 |         2
         2 |         8          0 |         8
         3 |        27          3 |        30
         4 |         9          9 |        18
         5 |         2          9 |        11
-----------+----------------------+----------
     Total |        48         21 |        69
```

Suppose we wanted to focus on just the cars with repair histories of four or better. We can use if suffix to do this.

```
tabulate rep78 foreign if rep78 >=4
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         4 |         9          9 |        18
         5 |         2          9 |        11
-----------+----------------------+----------
     Total |        11         18 |        29
```

Let's make the above table using the column and nofreq options. The command column requests column percentages while the command nofreq suppresses cell frequencies. Note that column and nofreq come after the comma. These are options on the tabulate command and options need to be placed after a comma.

```
tabulate rep78 foreign if rep78 >= 4, column nofreq
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         4 |     81.82      50.00 |     62.07
         5 |     18.18      50.00 |     37.93
-----------+----------------------+----------
     Total |    100.00     100.00 |    100.00
```

The use of if is not limited to the tabulate command. Here, we use it with the list command.

```
list if rep78 >= 4
                    make       price         mpg       rep78     foreign
  3.          AMC Spirit       3799          22           .           0
  5.     Buick Electra       7827         15          4          0
  7.        Buick Opel       4453         26          .          0
 15.      Chev. Impala       5705         16          4          0
 20.        Dodge Colt       3984         30          5          0
 24.       Ford Fiesta       4389         28          4          0
 29.      Merc. Bobcat       3829         22          4          0
 30.      Merc. Cougar       5379         14          4          0
//(omitted)
```

Did you see that some of the observations had a value of '.' for rep78? These are missing values. For example, the value of *rep78* for the AMC Spirit is missing. **Stata treats a missing value as positive infinity**, the highest number possible. So, when we said `list if rep78 >= 4`, Stata included the observations where *rep78* was '.' as well.

If we wanted to include just the valid (non-missing) observations that are greater than or equal to 4, we can do the following to tell Stata we want only observations where `rep78 >= 4` and *rep78* is not missing.

```
list if rep78 >= 4  &  !missing(rep78)
                   make      price         mpg      rep78    foreign
  5.     Buick Electra       7827         15          4          0
 15.      Chev. Impala       5705         16          4          0
 20.        Dodge Colt       3984         30          5          0
 24.       Ford Fiesta       4389         28          4          0
 29.      Merc. Bobcat       3829         22          4          0
 30.      Merc. Cougar       5379         14          4          0
 33.        Merc. XR-7       6303         14          4          0
 35.           Olds 98       8814         21          4          0
//(omitted)
```

This code will also yield the same output as above.

```
list if rep78 >= 4 & rep78 != .
```

We can use `if` with most Stata commands. Here, we get summary statistics for *price* for cars with repair histories of 1 or 2. Note the double equal (==) represents **IS EQUAL TO** and the pipe (|) represents **OR**.

```
summarize price if rep78 == 1 | rep78 == 2
Variable |     Obs         Mean    Std. Dev.       Min        Max
---------+-----------------------------------------------------
   price |      10         5687    3216.375       3667      14500
```

A simpler way to say this would be ...

```
summarize price if rep78 <= 2
Variable |     Obs         Mean    Std. Dev.       Min        Max
---------+-----------------------------------------------------
   price |      10         5687    3216.375       3667      14500
```

Likewise, we can do this for cars with repair history of 3, 4 or 5.

```
summarize price if rep78 == 3 | rep78 == 4 | rep78 == 5
Variable |     Obs         Mean    Std. Dev.       Min        Max
---------+-----------------------------------------------------
   price |      59     6223.847    2880.454       3291      15906
```

Additionally, we can use this code to designate a range of values. Here is a summary of *price* for the values 3 through 5 in *rep78*.

```
summarize price if inrange(rep78,3,5)
Variable |       Obs         Mean     Std. Dev.        Min         Max
```

```
----------+-------------------------------------------------------
   price |        59     6223.847     2880.454        3291       15906
```

Let's simplify this by saying `rep78 >= 3`.

```
summarize price if rep78 >= 3
Variable |     Obs         Mean    Std. Dev.        Min         Max
---------+-----------------------------------------------------
   price |      64     6239.984    2925.843        3291      15906
```

Did you see the mistake we made? We accidentally included the missing values because we forgot to exclude them. We really needed to say.

```
summarize price if rep78 >= 3 & !missing(rep78)
Variable |     Obs         Mean    Std. Dev.        Min         Max
---------+-----------------------------------------------------
   price |      59     6223.847    2880.454        3291      15906
```

2.1.1 Taking a random sample

It is also possible to take a simple random sample of your data using the sample command. This information can be found on our STATA FAQ page: How can I draw a random sample of my data?

2.1.2 Summary

Most Stata commands can be followed by `if`, for example

```
summarize if rep78 == 2
summarize if rep78 >= 2
summarize if rep78 >  2
summarize if rep78 <= 2
summarize if rep78 <2
summarize if rep78 != 2
```

`if` expressions can be connected with `|` for **OR**, `&` for **AND**.

2.1.3 Missing Values

Missing values are represented as '.' and are the highest value possible. Therefore, when values are missing, be careful with commands like

```
summarize if rep78 >  3
summarize if rep78 >= 3
summarize if rep78 != 3
```

to omit missing values, use

```
summarize if rep78 >  3 & !missing(rep78)
summarize if rep78 >= 3 & !missing(rep78)
summarize if rep78 != 3 & !missing(rep78)
```

2.2 A statistical sampler in Stata

Version info: Code for this page was tested in Stata 12.

This module will give a brief overview of some common statistical tests in Stata. Let's use the `auto` data file that we will use for our examples.

```
auto
```

```
sysuse auto
```

2.2.1 t-tests

Let's do a t-test comparing the miles per gallon (*mpg*) of foreign and domestic cars.

```
ttest mpg , by(foreign)
Two-sample t test with equal variances

------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
       0 |      52    19.82692     .657777    4.743297    18.50638    21.14747
       1 |      22    24.77273     1.40951    6.611187    21.84149    27.70396
---------+--------------------------------------------------------------------
combined |      74     21.2973    .6725511    5.785503     19.9569    22.63769
---------+--------------------------------------------------------------------
    diff |           -4.945804    1.362162               -7.661225   -2.230384
------------------------------------------------------------------------------
Degrees of freedom: 72

                      Ho: mean(0) - mean(1) = diff = 0

     Ha: diff <0 Ha: diff ~="0" Ha: diff> 0
       t =  -3.6308                t =  -3.6308               t =  -3.6308
   P < t =   0.0003          P > |t| =   0.0005           P > t =   0.9997
```

As you see in the output above, the domestic cars had significantly lower *mpg* (19.8) than the foreign cars (24.7).

2.2.2 Chi-square

Let's compare the repair rating (*rep78*) of the foreign and domestic cars. We can make a crosstab of *rep78* by *foreign*. We may want to ask whether these variables are independent. We can use the `chi2` option to request a chi-square test of independence as well as the crosstab.

```
tabulate rep78 foreign, chi2
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         1 |         2          0 |         2
         2 |         8          0 |         8
         3 |        27          3 |        30
         4 |         9          9 |        18
         5 |         2          9 |        11
-----------+----------------------+----------
     Total |        48         21 |        69

          Pearson chi2(4) =  27.2640   Pr = 0.000
```

The chi-square is not really valid when you have empty cells. In such cases when you have empty cells, or cells with small frequencies, you can request Fisher's exact test with the exact option.

```
tabulate rep78 foreign, chi2 exact
           |        foreign
     rep78 |         0          1 |     Total
-----------+----------------------+----------
```

```
         1 |         2          0 |         2
         2 |         8          0 |         8
         3 |        27          3 |        30
         4 |         9          9 |        18
         5 |         2          9 |        11
-----------+----------------------+----------
     Total |        48         21 |        69

          Pearson chi2(4) =   27.2640   Pr = 0.000
          Fisher's exact =                 0.000
```

2.2.3 Correlation

We can use the `correlate` command to get the correlations among variables. Let's look at the correlations among *price mpg weight* and *rep78*. (We use *rep78* in the correlation even though it is not continuous to illustrate what happens when you use correlate with variables with missing data.)

```
correlate price mpg weight rep78
 (obs=69)

         |    price      mpg   weight    rep78
---------+------------------------------------
   price |   1.0000
     mpg |  -0.4559   1.0000
  weight |   0.5478  -0.8055   1.0000
   rep78 |   0.0066   0.4023  -0.4003   1.0000
```

Note that the output above said (obs=69). The `correlate` command drops data on a listwise basis, meaning that if any of the variables are missing, then the entire observation is omitted from the correlation analysis.

We can use `pwcorr` (pairwise correlations) if we want to obtain correlations that deletes missing data on a pairwise basis instead of a listwise basis. We will use the obs option to show the number of observations used for calculating each correlation.

```
pwcorr price mpg weight rep78, obs
          |    price      mpg   weight    rep78
----------+------------------------------------
    price |   1.0000
          |       74
          |
      mpg |  -0.4686   1.0000
          |       74       74
          |
   weight |   0.5386  -0.8072   1.0000
          |       74       74       74
          |
    rep78 |   0.0066   0.4023  -0.4003   1.0000
          |       69       69       69       69
          |
```

Note how the correlations that involve *rep78* have an N of 69 compared to the other correlations that have an N of 74. This is because *rep78* has five missing values, so it only had 69 valid observations, but the other variables had no missing data so they had 74 valid observations.

2.2.4 Regression

Let's look at doing regression analysis in Stata. For this example, let's drop the cases where *rep78* is 1 or 2 or missing.

```
drop if (rep78 <= 2) | (rep78 ==.)
 (15 observations deleted)
```

Now, let's predict *mpg* from *price* and *weight*. As you see below, *weight* is a significant predictor of *mpg*, but *price* is not.

```
regress mpg price weight

  Source |       SS       df       MS                  Number of obs =      59
---------+------------------------------               F(  2,    56) =   47.87
   Model |  1375.62097     2  687.810483               Prob > F      =  0.0000
Residual |  804.616322    56  14.3681486               R-squared     =  0.6310
---------+------------------------------               Adj R-squared =  0.6178
   Total |  2180.23729    58  37.5902981               Root MSE      =  3.7905

------------------------------------------------------------------------------
     mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   price |  -.0000139   .0002108     -0.066   0.948     -.0004362    .0004084
  weight |   -.005828   .0007301     -7.982   0.000     -.0072906   -.0043654
   _cons |   39.08279   1.855011     21.069   0.000      35.36676    42.79882
------------------------------------------------------------------------------
```

What if we wanted to predict *mpg* from *rep78* as well. *rep78* is really more of a categorical variable than it is a continuous variable. To include it in the regression, we should convert *rep78* into dummy variables. Fortunately, Stata makes dummy variables easily using `tabulate`. The `gen`(rep) option tells Stata that we want to generate dummy variables from *rep78* and we want the stem of the dummy variables to be *rep*.

```
tabulate rep78, gen(rep)
      rep78 |      Freq.     Percent        Cum.
------------+-----------------------------------
          3 |         30       50.85       50.85
          4 |         18       30.51       81.36
          5 |         11       18.64      100.00
------------+-----------------------------------
      Total |         59      100.00
```

Stata has created *rep1* (1 if *rep78* is 3), *rep2* (1 if *rep78* is 4) and *rep3* (1 if *rep78* is 5). We can use the `tabulate` command to verify that the dummy variables were created properly.

```
tabulate rep78 rep1
           |  rep78==     3.0000
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         3 |         0         30 |        30
         4 |        18          0 |        18
         5 |        11          0 |        11
-----------+----------------------+----------
     Total |        29         30 |        59
tabulate rep78 rep2
           |  rep78==     4.0000
     rep78 |         0          1 |     Total
-----------+----------------------+----------
```

```
         3 |        30          0 |        30
         4 |         0         18 |        18
         5 |        11          0 |        11
-----------+----------------------+----------
     Total |        41         18 |        59
tabulate rep78 rep3
           |  rep78==     5.0000
     rep78 |         0          1 |     Total
-----------+----------------------+----------
         3 |        30          0 |        30
         4 |        18          0 |        18
         5 |         0         11 |        11
-----------+----------------------+----------
     Total |        48         11 |        59
```

Now we can include *rep1* and *rep2* as dummy variables in the regression model.

```
regress mpg price weight rep1 rep2

      Source |       SS       df       MS              Number of obs =      59
-------------+------------------------------           F(  4,    54) =   26.04
       Model |  1435.91975     4  358.979938           Prob > F      =  0.0000
    Residual |  744.317536    54  13.7836581           R-squared     =  0.6586
-------------+------------------------------           Adj R-squared =  0.6333
       Total |  2180.23729    58  37.5902981           Root MSE      =  3.7126


------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       price |  -.0001126   .0002133    -0.53   0.600    -.0005403    .0003151
      weight |   -.005107   .0008236    -6.20   0.000    -.0067584   -.0034557
        rep1 |  -2.886288   1.504639    -1.92   0.060    -5.902908    .1303314
        rep2 |   -2.88417   1.484817    -1.94   0.057    -5.861048    .0927086
       _cons |   39.89189   1.892188    21.08   0.000     36.09828     43.6855
------------------------------------------------------------------------------
```

2.2.5 Analysis of variance

If you wanted to do an analysis of variance looking at the differences in *mpg* among the three repair groups, you can use the oneway command to do this.

```
oneway mpg rep78
                         Analysis of Variance
    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between groups      506.325167      2   253.162583      8.47     0.0006
 Within groups      1673.91212     56   29.8912879
------------------------------------------------------------------------
    Total           2180.23729     58   37.5902981

Bartlett's test for equal variances:  chi2(2) =   9.9384  Prob>chi2 = 0.007
```

If you include the tabulate option, you get mean *mpg* for the three groups, which shows that the group with the best repair rating (*rep78* of 5) also has the highest *mpg* (27.3).

```
oneway mpg rep78, tabulate

```

```
            |           Summary of mpg
      rep78 |        Mean   Std. Dev.       Freq.
------------+------------------------------------
          3 |   19.433333   4.1413252          30
          4 |   21.666667   4.9348699          18
          5 |   27.363636   8.7323849          11
------------+------------------------------------
      Total |    21.59322   6.1310927          59

                        Analysis of Variance
    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between groups      506.325167      2   253.162583      8.47     0.0006
 Within groups      1673.91212     56   29.8912879
------------------------------------------------------------------------
    Total           2180.23729     58   37.5902981

Bartlett's test for equal variances:  chi2(2) =   9.9384  Prob>chi2 = 0.007
```

If you want to include covariates, you need to use the `anova` command. The continuous(price weight) option tells Stata that those variables are covariates.

```
anova mpg rep78 c.price c.weight
                           Number of obs =       59     R-squared     =  0.6586
                           Root MSE      = 3.71263     Adj R-squared =  0.6333

                  Source | Partial SS    df       MS           F     Prob > F
              -----------+----------------------------------------------------
                   Model |  1435.91975     4  358.979938      26.04     0.0000
                         |
                   rep78 |  60.2987853     2  30.1493926       2.19     0.1221
                   price |   3.8421233     1   3.8421233       0.28     0.5997
                  weight |  529.932889     1  529.932889      38.45     0.0000
                         |
                Residual |  744.317536    54  13.7836581
              -----------+----------------------------------------------------
                   Total |  2180.23729    58  37.5902981
```

2.3 An overview of Stata syntax

This module shows the general structure of Stata commands. We will demonstrate this using `summarize` as an example, although this general structure applies to most Stata commands.

Note: This code was tested in Stata 12.

Let's first use the `auto` data file.

```
use auto
```

As you have seen, we can type `summarize` and it will give us summary statistics for all of the variables in the data file.

```
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
  hdroom |      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
   displ |      74    197.2973    91.83722          79         425
  gratio |      74    3.014865    .4562871        2.19        3.89
 foreign |      74    .2972973    .4601885           0           1
```

It is also possible to obtain means for specific variables. For example, below we get summary statistics just for *mpg* and *price*.

```
summarize mpg price
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
     mpg |      74     21.2973    5.785503          12          41
   price |      74    6165.257    2949.496        3291       15906
```

We could further tell Stata to limit the summary statistics to just foreign cars by adding an if qualifier.

```
summarize mpg price if (foreign == 1)
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
     mpg |      22    24.77273    6.611187          14          41
   price |      22    6384.682    2621.915        3748       12990
```

The *if* qualifier can contain more than one condition. Here, we ask for summary statistics for the foreign cars which get less than 30 miles per gallon.

```
summarize mpg price if foreign == 1 & mpg <30
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
     mpg |      17    21.94118    3.896643          14          28
   price |      17    6996.235    2674.552        3895       12990
```

We can use the `detail` option to ask Stata to give us more detail in the summary statistics. Notice that the `detail` option goes after the comma. If the comma were omitted, Stata would give an error.

```
summarize mpg price if foreign == 1 & mpg <30 , detail
                              mpg
-------------------------------------------------------------
      Percentiles      Smallest
 1%           14             14
 5%           14             17
10%           17             17       Obs                  17
25%           18             18       Sum of Wgt.          17

50%           23                      Mean           21.94118
                        Largest       Std. Dev.      3.896643
75%           25             25
90%           26             25       Variance       15.18382
95%           28             26       Skewness      -.4901235
99%           28             28       Kurtosis       2.201759

                            price
-------------------------------------------------------------
      Percentiles      Smallest
 1%         3895           3895
 5%         3895           4296
10%         4296           4499       Obs                  17
25%         5079           4697       Sum of Wgt.          17

50%         6229                      Mean           6996.235
                        Largest       Std. Dev.      2674.552
75%         8129           9690
90%        11995           9735       Variance        7153229
95%        12990          11995       Skewness       .9818272
99%        12990          12990       Kurtosis       2.930843
```

Note that even though we built these parts up one at a time, they don't have to go together. Let's look at some other forms of the summarize command.

You can tell Stata which observation numbers you want using the in qualifier. Here we ask for summaries of observations 1 to 10. This is useful if you have a big data file and want to try out a command on a subset of observations.

```
summarize in 1/10
Variable |    Obs        Mean  Std. Dev.       Min       Max
---------+--------------------------------------------------
    make |      0
   price |     10      5517.4  2063.518       3799     10372
     mpg |     10        19.5   3.27448         15        26
   rep78 |      8       3.125  .3535534          3         4
  hdroom |     10         3.3  .7527727          2       4.5
   trunk |     10        14.7   3.88873         10        21
  weight |     10        3271  558.3796       2230      4080
  length |     10         194  19.32759        168       222
    turn |     10        40.2  3.259175         34        43
   displ |     10       223.9  71.77503        121       350
  gratio |     10       2.907  .3225264       2.41      3.58
 foreign |     10           0         0          0         0
```

Also, recall that you can ask Stata to perform summaries for foreign and domestic cars separately using by, as shown below.

```
sort foreign
by foreign: summarize
 -> foreign= 0
Variable |    Obs        Mean  Std. Dev.       Min       Max
---------+--------------------------------------------------
    make |      0
   price |     52    6072.423  3097.104       3291     15906
     mpg |     52    19.82692  4.743297         12        34
   rep78 |     48    3.020833   .837666          1         5
  hdroom |     52    3.153846  .9157578        1.5         5
   trunk |     52       14.75  4.306288          7        23
  weight |     52    3317.115  695.3637       1800      4840
  length |     52    196.1346  20.04605        147       233
    turn |     52    41.44231  3.967582         31        51
   displ |     52    233.7115  85.26299         86       425
  gratio |     52    2.806538  .3359556       2.19      3.58
 foreign |     52           0         0          0         0

-> foreign= 1
```

```
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
    make |       0
   price |      22    6384.682    2621.915      3748      12990
     mpg |      22    24.77273    6.611187        14         41
   rep78 |      21    4.285714    .7171372         3          5
  hdroom |      22    2.613636    .4862837       1.5        3.5
   trunk |      22    11.40909    3.216906         5         16
  weight |      22    2315.909    433.0035      1760       3420
  length |      22    168.5455    13.68255       142        193
    turn |      22    35.40909    1.501082        32         38
   displ |      22    111.2273    24.88054        79        163
  gratio |      22    3.507273    .2969076      2.98       3.89
 foreign |      22           1           0         1          1
```

Let's review all those pieces.

A command can be preceded with a `by` prefix, as shown below.

```
by foreign: summarize
```

There are many parts that can come after a command. They are each presented separately below.

For example, `summarize` followed by the names of variables.

```
summarize mpg price
```

`summarize` with in specifying a range of records to be summarized.

```
summarize in 1/10
```

`summarize` with simple `if` specifying records to summarize.

```
summarize if foreign == 1
```

`summarize` with complex `if` specifying records to summarize.

```
summarize if foreign == 1 & mpg > 30
```

summarize followed by option(s).

```
summarize , detail
```

So, putting it all together, the general syntax of the `summarize` command can be described as:

```
[by varlist:] summarize [varlist] [in range] [if exp] , [options]
```

Understanding the overall syntax of Stata commands helps you remember them and use them more effectively, and it also aids you understand the help files in Stata. All the extra stuff about `by`, `if` and `in` could be confusing. Let's have a look at the help file for `summarize`. It makes more sense knowing what the `by`, `if` and `in` parts mean.

```
help summarize
-------------------------------------------------------------------------------
help for summarize                                     (manual:  [R] summarize)
-------------------------------------------------------------------------------


Summary statistics
------------------


    [by varlist:]  summarize [varlist] [weight] [if exp] [in range]
                             [, { detail | meanonly } format ]
```

2.4 Missing data

2.4.1 Introduction

This module will explore missing data in STATA, focusing on numeric missing data. It will describe how to indicate missing data in your raw data files, as well as how missing data are handled in STATA logical commands and assignment statements.

We will illustrate some of the missing data properties in STATA using data from a reaction time study with eight subjects indicated by the variable *id* , and the subjects reaction times were measured at three time points (*trial1 trial2 trial3*). The input data file is shown below.

```
input id trial1 trial2 trial3
1 1.5 1.4 1.6
2 1.5 . 1.9
3 . 2.0 1.6
4 . . 2.2
5 1.9 2.1 2
6 1.8 2.0 1.9
7 . .  .
end
list
```

You might notice that some of the reaction times are coded using a single '.' as is the case for subject 2. The person measuring time for that trial did not measure the response time properly, therefore the data for the second trial is missing.

```
     +------------------------------+
     | id   trial1   trial2   trial3 |
     |------------------------------|
  1. |  1      1.5      1.4      1.6 |
  2. |  2      1.5        .      1.9 |
  3. |  3        .        2      1.6 |
  4. |  4        .        .      2.2 |
  5. |  5      1.9      2.1        2 |
     |------------------------------|
  6. |  6      1.8        2      1.9 |
  7. |  7        .        .        . |
     +------------------------------+
```

2.4.2 How STATA handles missing data in STATA procedures

As a general rule, STATA commands that perform computations of any type handle missing data by omitting the missing values. However, the way that missing values are omitted is not always consistent across commands, so let's take a look at some examples.

First, let's `summarize` our reaction time variables and see how STATA handles the missing values.

```
summarize trial1 trial2 trial3
```

As you see in the output below, `summarize` computed means using 4 observations for *trial1* and *trial2* and 6 observations for *trial3*. In short, the `summarize` command performed the computations on all the available data.

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      trial1 |         4       1.675    .2061553        1.5        1.9
      trial2 |         4       1.875    .3201562        1.4        2.1
      trial3 |         6    1.866667     .233809        1.6        2.2
```

A second example, shows how the tabulation or tab1 command handles missing data. Like summarize, tab1 uses just available data. Note that the percentages are computed based on the total number of non-missing cases.

```
tab1 trial1 trial2 trial3
-> tabulation of trial1

     trial1 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.5 |          2       50.00       50.00
        1.8 |          1       25.00       75.00
        1.9 |          1       25.00      100.00
------------+-----------------------------------
      Total |          4      100.00

-> tabulation of trial2

     trial2 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.4 |          1       25.00       25.00
          2 |          2       50.00       75.00
        2.1 |          1       25.00      100.00
------------+-----------------------------------
      Total |          4      100.00

-> tabulation of trial3

     trial3 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.6 |          2       33.33       33.33
        1.9 |          2       33.33       66.67
          2 |          1       16.67       83.33
        2.2 |          1       16.67      100.00
------------+-----------------------------------
      Total |          6      100.00
```

It is possible that you might want the percentages to be computed out of the total number of observations, and the percentage missing for each variable shown in the table. This can be achieved by including the missing option after the tabulation command,

```
tab1 trial1 trial2 trial3, m
-> tabulation of trial1

     trial1 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.5 |          2       28.57       28.57
        1.8 |          1       14.29       42.86
        1.9 |          1       14.29       57.14
          . |          3       42.86      100.00
------------+-----------------------------------
      Total |          7      100.00

-> tabulation of trial2

     trial2 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.4 |          1       14.29       14.29
          2 |          2       28.57       42.86
        2.1 |          1       14.29       57.14
          . |          3       42.86      100.00
------------+-----------------------------------
      Total |          7      100.00

-> tabulation of trial3

     trial3 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.6 |          2       28.57       28.57
        1.9 |          2       28.57       57.14
          2 |          1       14.29       71.43
        2.2 |          1       14.29       85.71
          . |          1       14.29      100.00
------------+-----------------------------------
      Total |          7      100.00
```

Let's look at how the correlate command handles missing data. We would expect that it would perform the computations based on the available data, and omit the missing values. Here is an example command.

```
corr trial1 trial2 trial3
```

The output is show below. Note how the missing values were excluded. For each pair variables, the corr command used the number of pairs that had valid data. For the pair formed by *trial1* and *trial2*, there were 3 pairs with valid data. For the pairing of *trial1* and *trial3* there were 4 valid pairs, and likewise there were 4 valid pairs for *trial3* and *trial2*. Using all of the valid pairs of data is called pairwise deletion of missing data.

```
             |   trial1   trial2   trial3
-------------+---------------------------
      trial1 |   1.0000
             |        4
             |
      trial2 |   0.9939   1.0000
             |        3        4
             |
      trial3 |   0.7001   0.6439   1.0000
             |        4        4        6
```

It is possible to ask STATA to only perform the correlations on the observations that had complete data for all of the variables on the var statement. For example, you might want the correlations of the reaction times just for the observations that had non-missing data on all of the trials. This is called listwise deletion of missing data meaning that when any of the variables are missing, the entire observation is omitted from the analysis. You can request listwise deletion within pwcorr as illustrated below.

```
pwcorr trial1 trial2 trial3, listwise obs
             |   trial1   trial2   trial3
-------------+---------------------------
      trial1 |   1.0000
             |        3
             |
      trial2 |   0.9939   1.0000
             |        3        3
             |
      trial3 |   1.0000   0.9939   1.0000
             |        3        3        3
```

2.4.3 Summary of how missing values are handled in STATA procedures

- `summarize`: For each variable, the number of non-missing values are used.
- `tabulation`: By default, missing values are excluded and percentages are based on the number of non-missing values. If you use the missing option on the `tab` command, the percentages are based on the total number of observations (non-missing and missing) and the percentage of missing values are reported in the table.
- `corr`: By default, correlations are computed based on the number of pairs with non-missing data (`pairwise` deletion of missing data). The `pwcorr` command can be used to request that correlations be computed only for observations that have non-missing data for all variables listed after the `pwcorr` command (`listwise` deletion of missing data).
- `reg`: If any of the variables listed after the `reg` command are missing, the observations missing that value(s) are excluded from the analysis (i.e., `listwise` deletion of missing data).
- For other procedures, see the STATA manual for information on how missing data are handled.

2.4.4 Missing values in assignment statements

It is important to understand how missing values are handled in assignment statements. Consider the example shown below.

```
gen sum1 = trial1 + trial2 + trial3
```

The `list` command below illustrates how missing values are handled in assignment statements. The variable *sum1* is based on the variables *trial1 trial2* and *trial3*. If any of those variables were missing, the value for *sum1* was set to missing. Therefore *sum1* is missing for observations 2, 3 and 4, as is the case for observation 7.

```
list
     +-------------------------------------+
     | id   trial1   trial2   trial3   sum1 |
     |-------------------------------------|
  1. |  1      1.5      1.4      1.6    4.5 |
  2. |  2      1.5        .      1.9      . |
  3. |  3        .        2      1.6      . |
  4. |  4        .        .      2.2      . |
  5. |  5      1.9      2.1        2      6 |
     |-------------------------------------|
  6. |  6      1.8        2      1.9    5.7 |
  7. |  7        .        .        .      . |
     +-------------------------------------+
```

As a general rule, computations involving missing values yield missing values. For example,

```
2 + 2 yields 4
2 + . yields .
2 / 2 yields 1
. / 2 yields .
2 * 3 yields 6
2 * . yields .
```

whenever you add, subtract, multiply, divide, etc., values that involve missing data, the result is missing.

In our reaction time experiment, the total reaction time *sum1* is missing for four out of seven cases. We could try totaling the data for the non-missing trials by using the `rowtotal` function as shown in the example below.

```
egen sum2 = rowtotal(trial1 trial2 trial3)
list
```

The results below show that sum2 now contains the sum of the non-missing trials.

```
     +---------------------------------------------+
     | id   trial1   trial2   trial3   sum1   sum2 |
     |---------------------------------------------|
  1. |  1      1.5      1.4      1.6    4.5    4.5 |
  2. |  2      1.5        .      1.9      .    3.4 |
  3. |  3        .        2      1.6      .    3.6 |
  4. |  4        .        .      2.2      .    2.2 |
  5. |  5      1.9      2.1        2      6      6 |
     |---------------------------------------------|
  6. |  6      1.8        2      1.9    5.7    5.7 |
  7. |  7        .        .        .      .      0 |
     +---------------------------------------------+
```

Note that the `rowtotal` function treats missing as a zero value. When summing several variables it may not be reasonable to treat missing as zero if an observations is missing on all variables to be summed. The `rowtotal` function with the missing option will return a missing value if an observation is missing on all variables.

```
egen sum3 = rowtotal(trial1 trial2 trial3) , missing

     +----------------------------------------------------+
     | id   trial1   trial2   trial3   sum1   sum2   sum3 |
     |----------------------------------------------------|
  1. |  1      1.5      1.4      1.6    4.5    4.5    4.5 |
  2. |  2      1.5        .      1.9      .    3.4    3.4 |
  3. |  3        .        2      1.6      .    3.6    3.6 |
  4. |  4        .        .      2.2      .    2.2    2.2 |
  5. |  5      1.9      2.1        2      6      6      6 |
     |----------------------------------------------------|
  6. |  6      1.8        2      1.9    5.7    5.7    5.7 |
  7. |  7        .        .        .      .      0      . |
     +----------------------------------------------------+
```

Other statements work similarly. For example, observed what happened when we try to create an average variable without using a function (as in the example below). If any of the variables *trial1*, *trial2* or *trial3* are missing, the value for *avg1* are set to missing.

```
gen avg1 = (trial1 + trial2 + trial3)/3
```

Alternatively, the `rowmean` function averages the data for the non-missing trials in the same way as the rowtotal function.

```
egen avg2 = rowmean(trial1 trial2 trial3)
```

Note: Had there been large number of trials, say 50 trials, then it would be annoying to have to type `avg=rowmean (trial1 trial2 trial3 trial4 ...)`. Here is a shortcut you could use in this kind of situation:

```
egen avg3 = rowmean(trial1 - trial3)
list
     +----------------------------------------------------+
     | id   trial1   trial2   trial3   avg1   avg2   avg3 |
     |----------------------------------------------------|
  1. |  1      1.5      1.4      1.6    1.5    1.5    1.5 |
  2. |  2      1.5        .      1.9      .    1.7    1.7 |
  3. |  3        .        2      1.6      .    1.8    1.8 |
  4. |  4        .        .      2.2      .    2.2    2.2 |
  5. |  5      1.9      2.1        2      2      2      2 |
     |----------------------------------------------------|
  6. |  6      1.8        2      1.9    1.9    1.9    1.9 |
  7. |  7        .        .        .      .      .      . |
     +----------------------------------------------------+
```

Finally, you can use the `rowmiss` and `rownonmiss` functions to determine the number of missing and the number of non-missing values, respectively, in a list of variables. This is illustrated below.

```
egen miss = rowmiss(trial1 - trial3)
egen nomiss = rownonmiss(trial1 - trial3)
list
```

For variable *nomiss*, observations 1, 5 and 6 had three valid values, observations 2 and 3 had two valid values, observation 4 had only one valid value and observation 7 had no valid values. The variable *miss* shows the opposite, it provides a count of the number of missing values.

```
     +-----------------------------------------------+
     | id   trial1   trial2   trial3   miss   nomiss |
     |-----------------------------------------------|
  1. |  1      1.5      1.4      1.6      0        3 |
  2. |  2      1.5        .      1.9      1        2 |
  3. |  3        .        2      1.6      1        2 |
  4. |  4        .        .      2.2      2        1 |
  5. |  5      1.9      2.1        2      0        3 |
     |-----------------------------------------------|
  6. |  6      1.8        2      1.9      0        3 |
  7. |  7        .        .        .      3        0 |
     +-----------------------------------------------+
```

2.4.5 Missing values in logical statements

It is important to understand how missing values are handled in logical statements. For example, say that you want to create a 0/1 variable for trial1 that is 1 if it is 1.5 or less, and 0 if it is over 1.5. We show this below (incorrectly, as you will see).

```
gen newvar1 =(trial2 <1.5)
list trial2 newvar1
```

It appears that something went wrong with our newly created variable *newvar1*! The observations with missing values for *trial2* were assigned a zero for *newvar1*.

```
     +------------------+
     | trial2   newvar1 |
     |------------------|
  1. |    1.4         1 |
  2. |      .         0 |
  3. |      2         0 |
  4. |      .         0 |
  5. |    2.1         0 |
     |------------------|
  6. |      2         0 |
  7. |      .         0 |
     +------------------+
```

Let's explore why this happened by looking at the frequency table of *trial2*.

As you can see in the output, missing values are at the listed after the highest value 2.1 This is because STATA treats a missing value as the largest possible value (e.g., positive infinity) and that value is greater than 2.1, so then the values for *newvar1* become 0.

```
tab trial2, missing
     trial2 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.4 |          1       14.29       14.29
          2 |          2       28.57       42.86
        2.1 |          1       14.29       57.14
          . |          3       42.86      100.00
------------+-----------------------------------
      Total |          7      100.00
```

Now that we understand how STATA treats missing values, we will explicitly exclude missing values to make sure they are treated properly, as shown below.

```
gen newvar2 =(trial2 <1.5) if trial2 !=.
list trial2 newvar1 newvar2
```

As you can see in the STATA output below, the new variable *newvar2* has missing values for observations that are also missing for *trial2*.

```
     +-----------------------------+
     | trial2   newvar1   newvar2 |
     |-----------------------------|
  1. |    1.4         1         1 |
  2. |      .         0         . |
  3. |      2         0         0 |
  4. |      .         0         . |
  5. |    2.1         0         0 |
     |-----------------------------|
  6. |      2         0         0 |
  7. |      .         0         . |
     +-----------------------------+
```

2.4.6 Missing values in logical statements

When creating or recoding variables that involve missing values, always pay attention to whether the variable includes missing values.

2.4.7 For more information

- See the STATA FAQ: How can I recode missing values into different categories?
- See the STATA FAQ: Can I quickly see how many missing values a variable has? for more information on examining the number of missing and non-missing values for a particular variable or set of variables.