Introduction to Geographic Information Science
Week 03 Stata Commands

Christopher G. Prener, Ph.D.

Spring, 2016

Contents

1 Arithmetic Operators

1.1 Syntax

+ (addition)
- (subtraction)
* (multiplication)
/ (division)
ˆ (raise to a power)

1.2 Examples

These examples use the display command. This command is used for a variety of purposes by Stata, including making mathematical calculations. Additional details can be found in the Stata documentation for the display command (link). Note that arithmetic operators may be used with a variety of commands, and that display is used for illustrative purposes only.

To add two numbers together:

```
.  display 10+10
20
```

To subtract one number from another:

```
.  display 20-10
10
```

To multiply two numbers:

```
.  display 10*10
100
```

To divide one number by another:

```
.  display 100/10
10
```

To raise a number by a power:

```
.  display 102̂
100
```

Stata allows the use of parantheses to specify the order in which calculations are conducted. Calculations within parantheses will be conducted first. Without parantheses, calculations will be evaluated using standard order of operations procedures (ˆ, /, *, -, and then +). Here is an example of this difference:

```
.  display (10+10)*5
100
.  display 10+10*5
60
```

2 Relational Operators

2.1 Syntax

`>` (greater than)
`>=` (greater than or equal to)
`<` (less than)
`<=` (less than or equal to)
`==` (equal to)
`!=` (not equal to)

2.2 Examples

These examples use the `summarize` command. This command was described in the Week 01 Stata command notes available on GitHub. Additional details can be found in the Stata documentation for the `summarize` command (link).

Note that arithmetic operators may be used with a variety of commands, and that `summarize` is used for illustrative purposes only. Any command that includes the `[if]` statement in its syntax documentation can use these operators. So far this semester, the only command that includes this statement that we have discussed is the `summarize` command.

In the `census.dta` dataset that comes installed with Stata, you could produce descriptive statistics for the urban population (`popurban`) of a state **only** if the state's urban population is greater than one million individuals:

```
.  sysuse census.dta
(1980 Census data by state)

.  summarize popurban if popurban > 1000000

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    popurban |         34     4638607     4391220    1179556   2.16e+07
```

Similarly, you could produce descriptive statistics for the urban population (`popurban`) **only** if a state's urban population is less than or equal to one million individuals:

```
.  summarize popurban if popurban <= 1000000

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    popurban |         16      543752      255768     172735     987859
```

These same statements would also work with `>=` or `<` operators as well. However, if we tried to use the `==` (equal to) operator, we would get the following result:

```
.  summarize popurban if popurban == 1000000

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    popurban |          0
```

We get this result because no state has an urban population of **exactly** one million individuals. A more appropriate use of the `==` (equal to) operator would be to obtain descriptive statistics for the urban population of **only** states in New England. The variable `region` is a categorical variable where a value of `1` represents states in New England. This value can be combined with the `==` (equal to) operator:

```
.  summarize popurban if region == 1

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    popurban |          9     4322838     4918176     172735   1.49e+07
```

We could use the `!=` (equal to) operator to get the compliment of this measurement - descriptive statistics for urban populations in states that are **not** in New England:

```
.  summarize popurban if region != 1

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    popurban |         41     3109930     3922319     258567   2.16e+07
```

3 Logical Operators

3.1 Syntax

`&` (and)
`|` (or)

3.2 Examples

These examples use the `summarize` command. This command was described in the Week 01 Stata command notes available on GitHub. Additional details can be found in the Stata documentation for the `summarize` command (link).

Note that logical operators may be used with a variety of commands, and that `summarize` is used for illustrative purposes only. Any command that includes the `[if]` statement in its syntax documentation can use these operators. So far this semester, the only command that includes this statement that we have discussed is the `summarize` command.

In the `census.dta` dataset that comes installed with Stata, you could produce descriptive statistics for the urban population (`popurban`) of a state **only** if that urban population is greater than one million individuals **and** a state is located in the West:

```
.  sysuse census.dta
(1980 Census data by state)

.  summarize popurban if popurban > 1000000 & region == 4

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    popurban |          6     5379105     7972992    1233060   2.16e+07
```

Note how the `summarize` command above combines a logical operator with two arithmetic operators. By adding parantheses and additional operators, these `if` statements can be used to produce complex conditions that Stata can evaluate.

A second example below also uses a logical operator. It produces descriptive statistics for urban populations **only** if states are located in New England **or** the South:

```
.  summarize popurban if region == 1 | region == 3

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    popurban |         25     3547259     3662622     172735   1.49e+07
```

If we had tried to use the `&` (and) logical operator, we would have returned no data because a state cannot simultaneously be located in New England and the South.

4 Listing Observations

4.1 Syntax

```
list [varlist] [if] [in] [, options]
```

4.2 Basic Examples

The `list` command will display the values of variables, observation by observation. This can be an invaluable command for understanding the structure of your data and for list observtions with particular characteristics. If no `varlist` is given, all variables will be listed:

```
.  sysuse census.dta
(1980 Census data by state)

.  list

     +-----------------------------------------------------------------------+
  1. | state         | state2 | region  |        pop |    poplt5 |   pop5_17 |
     | Alabama       | AL     | South   |  3,893,888 |   296,412 |   865,836 |
     |------------------------+----------------------------------+-----------|
     |     pop18p |    pop65p |   popurban  | medage  |   death  | marriage  |
     |  2,731,640 |   440,015 |  2,337,713  |  29.30  |  35,305  |   49,018  |
     |-----------------------------------------------------------------------|
     |                                divorce                                |
     |                                 26,745                                |
     +-----------------------------------------------------------------------+
```

(output omitted)

```
     +-----------------------------------------------------------------------+
 50. | state         | state2 | region  |        pop |    poplt5 |   pop5_17 |
     | Wyoming       | WY     | West    |    469,557 |    44,845 |   100,708 |
     |------------------------+----------------------------------+-----------|
     |     pop18p |    pop65p |   popurban  | medage  |   death  | marriage  |
     |    324,004 |    37,175 |    294,639  |  27.10  |   3,215  |    6,868  |
     |-----------------------------------------------------------------------|
     |                                divorce                                |
     |                                  4,003                                |
     +-----------------------------------------------------------------------+
```

With a `varlist` specified, the output will be restricted to those variables. The following output lists the two letter abbreviation for each state as well as the number of marriages in 1980:

```
.  list state2 marriage

     +-------------------+
     | state2   marriage |
     |-------------------|
  1. | AL         49,018 |
  2. | AK          5,361 |
  3. | AZ         30,223 |
  4. | AR         26,513 |
  5. | CA        210,864 |
     |-------------------|
```

(output omitted)

```
     |-------------------|
 46. | VA         60,210 |
 47. | WA         47,728 |
 48. | WV         17,391 |
 49. | WI         41,111 |
 50. | WY          6,868 |
     +-------------------+
```

4.3 Example with an `if` Statement

The list command includes the ability to use logical and relational operators in combination with an `if` statement. This is particularly useful for identifying observations that meet a specific set of criteria.

For example, we could produce a list of states that `only` contains states that had over one hundred thousand marriages in 1980:

```
.  sysuse census.dta
(1980 Census data by state)

.  list state2 marriage if marriage > 100000

     +-------------------+
     | state2   marriage |
     |-------------------|
  5. | CA        210,864 |
  9. | FL        108,344 |
 13. | IL        109,823 |
 28. | NV        114,333 |
 32. | NY        144,518 |
     |-------------------|
 43. | TX        181,762 |
     +-------------------+
```

4.4 Examples with an in Statement

The list command also includes the ability to list a limited number of observations. This is particularly useful for exploring the structure of your data.

The basic syntax for in statements is:

```
list [varlist] in firstObs/lastObs
```

For example, the first five observations of the census.dta dataset can be listed for the two letter abbreviation for each state as well as the number of marriages in 1980:

```
.  sysuse census.dta
(1980 Census data by state)

.  list state2 marriage in 1/5

     +-------------------+
     | state2   marriage |
     |-------------------|
  1. | AL         49,018 |
  2. | AK          5,361 |
  3. | AZ         30,223 |
  4. | AR         26,513 |
  5. | CA        210,864 |
     +-------------------+
```

The varlist as well as the observation numbers can vary. The following example lists observations ten to fourteen for three variables - the two letter abbreviation for each state as well as the number of marriages and divorces in 1980:

```
.  list state2 marriage divorce in 10/14

     +-----------------------------+
     | state2   marriage   divorce |
     |-----------------------------|
 10. | GA         70,638    34,743 |
 11. | HI         11,856     4,438 |
 12. | ID         13,428     6,596 |
 13. | IL        109,823    50,997 |
 14. | IN         57,853    40,006 |
     +-----------------------------+
```

Finally, the following example lists the same three observations as the last but for the final five observations. Rather than specifying observations 46 to 50, this code is written for maximum flexibility - it can be applied to any dataset even if the total sample size is unknown. It does this by using a `-5` for the `firstObs` value and the lowercase letter 'l' (`l`) for the `lastObs` value.

```
.  list state2 marriage divorce in -5/l

     +------------------------------+
     | state2   marriage   divorce |
     |------------------------------|
 46. | VA         60,210    23,615 |
 47. | WA         47,728    28,642 |
 48. | WV         17,391    10,273 |
 49. | WI         41,111    17,546 |
 50. | WY          6,868     4,003 |
     +------------------------------+
```

4.5 Options

There are two options that are worth briefly mentioning. The first is the `noobs` option. In the example above, you can see the notation on the left side of the table. These row numbers are based on the current sort order of the data. In small datasets they can sometimes be useful but in large ones they are rarely helpful. To supress them, use the `noobs` option:

```
.  sysuse census.dta
(1980 Census data by state)

.  list state2 marriage in -3/l, noobs

  +--------------------+
  | state2   marriage |
  |--------------------|
  | WV         17,391 |
  | WI         41,111 |
  | WY          6,868 |
  +--------------------+
```

If you prefer the tables without the lines, you can specify the `clean` option:

```
.  list state2 marriage in -3/l, clean

       state2   marriage
 48.   WV         17,391
 49.   WI         41,111
 50.   WY          6,868
```

5 Creating New Variables

5.1 Syntax

```
generate [type] newvar =expression [if] [in]
```

5.2 Examples

To see create a new, empty variable in the `census.dta` dataset:

```
.  sysuse census.dta
(1980 Census data by state)

.  generate varOne = 1

.  summarize varOne

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      varOne |         50           1           0          1          1
```

To create a copy of a pre-existing variable in the `census.dta` dataset:

```
.  generate popNew = pop

.  summarize pop

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         pop |         50     4518149     4715038     401851   2.37e+07


.  summarize popNew

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      popNew |         50     4518149     4715038     401851   2.37e+07
```

Finally, `if` statements can be used with the generate command to create variables that meet a particular set of criteria. For example, the following code creates a variable that contains populations only for states that have populations greater than one million individuals:

```
.  generate popHigh = pop if pop > 1000000
(12 missing values generated)

.  summarize popHigh

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
     popHigh |         38     5716400     4825525    1124660   2.37e+07
```

5.3 Notes

Additional details for the `generate` command can be found in the Stata documentation for that command (link).

6 Replacing Data in a Variable

6.1 Syntax

```
replace oldvar =expression [if] [in]
```

6.2 Examples

The replace command will always be used to overwrite data in an existing variable. For this reason, it is a best practice to always create a new variable first before using replace. This preserves all existing data for future use or reference.

To create, for example, a binary variable that represents low population states that have populations less than one million individuals, first use the generate command to create a new variable. Set the new variable equal to missing so that it is created with empty data:

```
.  sysuse census.dta
(1980 Census data by state)

.  generate lowPop = .
(50 missing values generated)
```

Once that variable has been created, two instances of the replace command can be used to define the value 0 as representing states with populations greater than one million individuals and the value 1 as representing states with populations less than or equal to one million individuals:

```
.  replace lowPop = 0 if pop > 1000000
(38 real changes made)

.  replace lowPop = 1 if pop <= 1000000
(12 real changes made)
```

6.3 Notes

Additional details for the replace command can be found in the Stata documentation for the generate command (link).

7 Creating Frequency Tables

7.1 Syntax

```
tabulate varlist [if] [in] [, options]
```

7.2 Options

For frequency tables, there are two key options to use. The first is the `nolabel` option. This option will remove value labels from the table, making it easier to identify the numeric structure of categorical and ordinal variables.

The second option is the `missing` option. This option will reverse the default behavior of the `tabulate` command, which is to suppress missing data from output.

7.3 Examples

To see the frequency table for a single variable, `region`, in the `census.dta` dataset:

```
.  sysuse census.dta
(1980 Census data by state)

.  tabulate region

     Census |
     region |      Freq.     Percent        Cum.
------------+-----------------------------------
         NE |          9       18.00       18.00
    N Cntrl |         12       24.00       42.00
      South |         16       32.00       74.00
       West |         13       26.00      100.00
------------+-----------------------------------
      Total |         50      100.00
```

To see the underlying numeric structure of the same variable:

```
.  tabulate region, nolabel

     Census |
     region |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |          9       18.00       18.00
          2 |         12       24.00       42.00
          3 |         16       32.00       74.00
          4 |         13       26.00      100.00
------------+-----------------------------------
      Total |         50      100.00
```

By specifying two variables in the varlist, you can compare categorical and/or ordinal level variables. Doing so with the census.dta dataset requires that we construct a new variable for illustrative purposes:

```
.  generate newEngland = region

.  replace newEngland = 0 if region > 1
(41 real changes made)

.  tabulate region newEngland

    Census |      newEngland
    region |         0          1 |     Total
-----------+----------------------+----------
        NE |         0          9 |         9
   N Cntrl |        12          0 |        12
     South |        16          0 |        16
      West |        13          0 |        13
-----------+----------------------+----------
     Total |        41          9 |        50
```

This process created a binary (0/1 or ”zero-one”) variable that represents the condition of being a New England state or not being a New England state.

The missing option can be used to identify missing values in categorical or ordinal level variables. For these examples, it is necessary to download the file missing.dta, which can be found in the GitHub repository week-03 in the commands-03 folder. These examples assume that you have already set your working directory to the folder where the file missing.dta is located.

Tabulating the variable z produces the following output:

```
.  tabulate z

              z |      Freq.     Percent        Cum.
----------------+-----------------------------------
             no |         37       37.76       37.76
            yes |         61       62.24      100.00
----------------+-----------------------------------
          Total |         98      100.00
```

To see missing values in the variable z:

```
.  tabulate z, missing

              z |      Freq.     Percent        Cum.
----------------+-----------------------------------
             no |         37       37.00       37.00
            yes |         61       61.00       98.00
not applicable  |          2        2.00      100.00
----------------+-----------------------------------
          Total |        100      100.00
```

Both options can be combined to see the underlying valid and missing values of the variable `z`:

```
.  tabulate z, nolabel missing

          z |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         37       37.00       37.00
          1 |         61       61.00       98.00
         .a |          2        2.00      100.00
------------+-----------------------------------
      Total |        100      100.00
```

Missing values are discussed more in the next section.

7.4 Notes

Additional details for the `tabulate` command with a single variable in the `varlist` can be found in the Stata documentation for `tabulate oneway` command (link).

Additional details for the `tabulate` command with two variable sin the `varlist` can be found in the Stata documentation for `tabulate twoway` command (link).

8 Missing Data

8.1 Syntax

`.` (period; system missing value)
`.a, .b, .c, ... .z` (extended missing values)

8.2 Examples

For these examples, it is necessary to download the file `missing.dta`, which can be found in the GitHub repository `week-03` in the `commands-03` folder. These examples assume that you have already set your working directory to the folder where the file `missing.dta` is located.

```
.  use missing.dta
(SOC 4650/5650 Intro to GISc Missing Data Examples)
```

Summarizing the `id` shows that there are 100 observations in the dataset. You can manually confirm this by opening the data viewer and scrolling to the bottom of the dataset.

```
.  summarize id

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
          id |        100        50.5    29.01149          1        100
```

However, if you summarize variable `x`, you will see that there are only 98 observations. This is an indication that there is some data missing in that dataset.

```
.  summarize x

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
           x |         98    156.7422    27.32463   103.9041   199.8077
```

With continuous data, the list command is the easiest way to identify which cases are missing. Below are two different ways for producing the same list:

```
.  list id x if x == .

     +---------+
     | id    x |
     |---------|
 24. | 24    . |
 49. | 49    . |
     +---------+
```

(Continued on next page)

```
.  list id x if missing(x)

     +--------+
     | id   x |
     |--------|
 24. | 24   . |
 49. | 49   . |
     +--------+
```

In addition to simple periods to identify missing data, Stata allows for 27 different extended missing value characters. These range from `.a` to `.z`. Each of these *must* be accompanied by a leading period. There cannot be a space between the leading period and the character, and characters cannot be combined (e.g. `.aa`). This approach to missing data can be found in the varable `y`, which contains a number of missing values:

```
.  summarize y

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           y |        96    7.546618    1.369873   5.088363   9.952309
```

```
.  list id y if missing(y)

     +---------+
     | id    y |
     |---------|
  7. |  7   .a |
 24. | 24   .b |
 32. | 32   .a |
 49. | 49   .b |
     +---------+
```

The values `.`, `.a`, and `.b` are distinct. Thus the following three examples produce different output from the previous example:

```
.  list id y if y == .
```

(*No output*)

```
.  list id y if y == .a

     +---------+
     | id    y |
     |---------|
  7. |  7   .a |
 32. | 32   .a |
     +---------+
```

(Continued on next page)

```
.  list id y if y == .b

     +---------+
     | id    y |
     |---------|
 24. | 24   .b |
 49. | 49   .b |
     +---------+
```

These different missing data values can be used to represent specific types of missing data. For example, in the variable `z`, which is a binary variable, `.a` represents ”not applicable”.

```
.  tabulate z, missing

             z |      Freq.     Percent        Cum.
---------------+-----------------------------------
            no |         37       37.00       37.00
           yes |         61       61.00       98.00
not applicable |          2        2.00      100.00
---------------+-----------------------------------
         Total |        100      100.00
```

8.3 The Numerical Meaning of Missing Values

Missing values in Stata retain a numerical meaning. They are essentially exceptionally large numbers (approaching infinity) that Stata handles in specific ways. Thus

```
all nonmissing numbers < .  < .a < .b < ...  < .z
```

It is therefore important to remember that commands must explicitly exclude missing data from output or from inclusion in recoded variables. For example, the following output lists the variable `x` for all cases where `x` is greater than 195. Since missing data are technically very large numbers, they are included in the resulting output:

```
.  list id x if x > 195

     +----------------+
     |  id          x |
     |----------------|
  6. |   6   197.9749 |
  9. |   9   199.9561 |
 24. |  24          . |
 39. |  39   196.4724 |
 47. |  47   195.7684 |
     |----------------|
 49. |  49          . |
100. | 100   197.7366 |
     +----------------+
```

An additional pair of logical and relational operators should be included in any case where missing data may be present. The following example still only list values of the variable `x` that are greater than 195, but it now also excludes the very large missing data values.

```
. list id x if x > 195 & x < .

     +----------------+
     |  id          x |
     |----------------|
  6. |   6   197.9749 |
  9. |   9   199.9561 |
 39. |  39   196.4724 |
 47. |  47   195.7684 |
100. | 100   197.7366 |
     +----------------+
```

In cases where extended missing values are used, a similar approach is necessary. Since `.a` is greater than a system missing value (the period - `.`), excluding any value greater than or equal to the period (`.`) is sufficient.

```
. list id y if y > 9.8

     +---------------+
     | id          y |
     |---------------|
  7. |  7         .a |
 24. | 24         .b |
 32. | 32         .a |
 33. | 33   9.961525 |
 49. | 49         .b |
     |---------------|
 52. | 52   9.866672 |
 89. | 89   9.830105 |
     +---------------+
```

```
. list id y if y > 9.8 & y < .

     +---------------+
     | id          y |
     |---------------|
 33. | 33   9.961525 |
 52. | 52   9.866672 |
 89. | 89   9.830105 |
     +---------------+
```

8.4 Notes

Additional details can be found in the Stata documentation for missing values (link).