1

Language Elements

In this chapter, the main topic will be the mode of representation of data in Stata and their manipulation. In particular, we will see how to represent numerical variables and categorical variables, how to operate on subsets of observations or how to only select parts that verify logical conditions, and finally the base syntax of Stata instructions (`if`, `in`, `by`).

1.1. Data representation in Stata

The data manipulated in Stata are mainly of two types: numbers and character strings. The numbers can be integers or real numbers. The first type is also used to encode the levels of a categorical variable to which text labels can be associated, called "variable labels" in Stata.

1.1.1. *The Stata language*

There are controls that allow users to easily generate a series of random numbers. The following example helps to familiarize with the basic elements of the Stata language. The following series of instructions allows storing in a variable called `x` 10 observations obtained from a normal distribution of average 12 and standard deviation 2:

```
. set obs 10
. generate x = rnormal(12, 2)
. format x %6.3f
. summarize x, format
obs was 0, now 10
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           x |        10      11.112       2.246       7.956     14.224
```

Several remarkable features of the language should be noted: it is necessary to indicate the size of the sample used. In the following sections, we will see how these data can be obtained during manual input or when importing an external data file. The command `generate` can be used to associate with a variable, here `x`, a sequence of numeric values (assimilated here to our 10 observations) provided by the function (or subcommand) `rnormal()`. This latter has options available which enables the user to specify parameters of the distribution (mean and standard deviation, respectively) be specified. The command `format x` makes it possible to limit the display to 3 decimal places: this is a property of representation of the values of `x` directly associated with the variable that the command `summarize` can use.

Individual data can be examined by means of the command `list`. For example, the command `list x` will display all of the values of `x`. Since there is only a single variable present in the Stata workspace, it is nonetheless equivalent to typing `list` for short. The option `in` can be included to restrict the display of the values of `x` to the fifth observation or to the first five observations. In the latter case, the ranks of the observations are indicated in the first value/last value form: the expression `1/5` therefore designates the observations numbered 1–5:

```
. list x in 5
     +-------+
     |     x |
     |-------|
  5. | 7.956 |
     +-------+
. list x in 1/5
     +--------+
     |      x |
     |--------|
  1. | 11.118 |
  2. | 13.889 |
  3. | 14.224 |
  4. |  8.726 |
  5. |  7.956 |
     +--------+
```

1.1.2. *Creating and manipulating variables*

In the case of small datasets, it is possible for users to enter themselves the observations, although most of the time it will be preferable to work from an external file. For this purpose, the command `input` is available which is employed in the following manner: after the name of the command, the name of the variable(s) is indicated, separated by a space, and then the user ought to press the Enter key before entering the data, always separated by spaces. To indicate to Stata that the entry is complete, the word `end` must be inserted. This manual entry can also be performed from the data editor (Data ▷ Data Editor ▷ Data Editor (Edit)). Here is an example of the usage with a series of 10 weight measurements collected in newborns (x, in grams) and their mother (y, in kilograms).

x	2523	2551	2557	2594	2600	2622	2637	2637	2663	2665
y	82.7	70.5	47.7	49.1	48.6	56.4	53.6	46.8	55.9	51.4

Table 1.1. *Artificial data on the weight at birth*

Data input in Stata would thus be achieved as follows: enter `input x y` in the Stata console, and then for the line numbered `1.` indicate `2523 82.7` and type Enter, followed by the line numbered `2.` indicate `2551 82.7` and Enter and so on until the 10th line. For the 11th line, we will simply write `end` and press Enter. The result should look like the following:

```
. input x y

        x    y
  1. 2523 82.7
  2. 2551 70.5
  3. 2557 47.7
  4. 2594 49.1
  5. 2600 48.6
  6. 2622 56.4
  7. 2637 53.6
  8. 2637 53.6
  9. 2637 46.8
 10. 2663 55.9
 11. 2665 51.4
 12. end
```

It is possible to transform the values taken by a variable or to create new variables based on the values taken by a variable by using the commands `generate` and `replace`. This last command works exclusively on an existing variable. Here is an example of using `generate` where the weight of infants (x) is converted into kilograms:

```
. generate x2 = x / 1000
. list x x2 in 1/3

     +--------------+
     |    x      x2 |
     |--------------|
  1. | 2523   2.523 |
  2. | 2551   2.551 |
  3. | 2557   2.557 |
     +--------------+
```

It is also possible to replace the set of values from any transformation, for example the logarithm:

```
 . replace x 2 = log (x
2) (10 real changes made)
```

or specifically override certain values indicating an observation number, as illustrated hereafter:

```
. replace x2 = 2600 in 3
(1 real change made)
. list x x2 in 1/3

     +-----------------+
     |    x         x2 |
     |-----------------|
  1. | 2523   .9254487 |
  2. | 2551   .9364855 |
  3. | 2557       2600 |
     +-----------------+
```

Finally, the command `drop` can be inserted to delete any variable of the workspace. In this case, the observations are then permanently lost:

```
. drop x2
```

1.1.3. *Indexed or criteria-based selection of observations*

The option for the selection of observations based on indices (or ranks) has already been presented with the option `in`:

```
. list x in 1/3
     +------+
     |    x |
     |------|
  1. | 2523 |
  2. | 2551 |
  3. | 2557 |
     +------+
```

It is quite possible to select observations on the basis of an external criterion, for example the values taken by a second numerical variable. In the following example, only the weight of babies for which the mothers' weight ≤ 50 is retained:

```
. list x if y <= 50
     +------+
     |    x |
     |------|
  3. | 2557 |
  4. | 2594 |
  5. | 2600 |
  8. | 2637 |
     +------+
```

Suppose that there is also some information available concerning the fact that the mother was smoking during the first trimester of pregnancy; we will call this variable z. When the mother did not smoke during this period, the variable is equal to 1; when the mother was smoking, the variable is 2. In the following, the previous data table is presented amended to account for this information.

x	2523	2551	2557	2594	2600	2622	2637	2637	2663	2665
y	82.7	70.5	47.7	49.1	48.6	56.4	53.6	46.8	55.9	51.4
z	1	1	2	2	2	1	1	1	2	2

Table 1.2. *Augmented artificial data about the weights at birth*

The new data entry does not raise any difficulty and once again `input` will be used, indicating z as a new variable. The end of the input should be signaled by means of

the instruction `end` (followed by Enter). Here follows an overview of the first five observations for the three variables:

```
. list in 1/5
     +-----------------+
     |    x      y   z |
     |-----------------|
  1. | 2523   82.7   1 |
  2. | 2551   70.5   1 |
  3. | 2557   47.7   2 |
  4. | 2594   49.1   2 |
  5. | 2600   48.6   2 |
     +-----------------+
```

It is possible to refine our search criteria by restricting the selection of observations x depending on the values taken by y and z. The following instruction displays the weight of babies whose mother weighs less (strictly) than 55 kg and who was not smoking during her pregnancy:

```
. list x if y < 55 & z == 1
     +------+
     |    x |
     |------|
  7. | 2637 |
  8. | 2637 |
     +------+
```

It can be seen that there are two statistical units that verify the previous conditions ($y < 55$ and $z = 1$). The logical "and" (conjunction) is denoted `&`, while the logical "or" (disjunction) is written `|`. In order to count the observations verifying the preceding condition, it is possible to employ the command `count`:

```
. count if y < 55 & z == 1
    2
```

1.1.4. *Processing the missing values*

Missing values are represented by a dot in Stata. For example, it is possible to replace the third observation of `x` with a missing value, by using the command `replace` presented above:

```
. replace x = . in 3
```

```
(1 real change made, 1 to missing)
. summarize x

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           x |         9    2610.222    48.53035       2523       2665
```

It shall be verified that the number of observations reported by `summarize` is correctly accounting for the missing datum ($n = 9$ instead of 10). We can also verify the number of missing values identified for a variable by using the command `misstable`:

```
. misstable summarize x
                                                              Obs<.
                                                  +------------------------------
                |                                 | Unique
     Variable   |     Obs=.     Obs>.     Obs<.   | values        Min         Max
  -------------+-----------------------------------+------------------------------
              x |         1                     9   |      8       2523        2665
  ------------------------------------------------------------------------------
```

In practice, the missing data are represented in the form of a very large number; care should be taken when performing numerical comparison tests, and when in doubt always use a test of the type `if !missing(x)` (more elegant than `if x< .`) in order to be certain that you are working with the observed data:

```
. summarize y if !missing(x)
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           y |         9    57.22222    11.85219      46.8       82.7
```

1.1.5. *Data management*

1.1.6. *Importing external data*

There are several commands that enable importing the data contained in a text file. For files in which the fields are separated by one or more spaces, we will make use of the command `infile`. In the case where there is a field separator such as a comma (typical of CSV files exported from an Excel spreadsheet) or a tabulation, the command `insheet` will be employed, eventually specifying the type of the field delimiter with the option `delimiter()`. The command `insheet` works well in the case where the first line of the file contains the name of the variables. However, in both cases, it is possible to provide a list for the names of the variables, and the name

of the file will always be indicated after the instruction `using`. The representation format of the data that was read can also be customized by specifying its type before each variable, for example, with the command:

```
. infile str5 name age byte rep using "fichier.txt", clear
```

Stata is instructed to build from the file called `fichier.txt` a table containing three variables: `name`, `age` and `rep`. The variable `name` must be explicitly treated as a string of characters (maximum five) and the storage format of the variable `rep` must be limited to the minimum (1 byte = values varying from -127 to -100), for instance so as not to unnecessarily occupy memory. Finally, in some cases, we may avoid inserting options at the command line (variables name and storage format) and store all of this information in what is called a dictionary, see `help infile2`. Note that Stata provides dialog boxes that make it possible to specify different options for each of these commands, such as the field delimiter or the presence of a header row, and that a data previewer allows verifying whether the data structure has been properly defined.

Consider data on birth weight, available in a file called `birthwt.dat` in which fields are separated by a space as in the following overview:

```
0 19 182 2 0 0 0 1 0 2523
0 33 155 3 0 0 0 0 3 2551
0 20 105 1 1 0 0 0 1 2557
0 21 108 1 1 0 0 1 2 2594
0 18 107 1 1 0 0 1 0 2600
```

It should be noted that the name of the variables does not appear on the first line of the file. Each column corresponds, respectively, to the following variables: weight status of the baby at birth `low` (= 1 if weight < 2.5 kg, 0 otherwise), `age` of the mother (years), `lwt` weight of the mother (in pounds), `race` ethnicity of the mother (encoded in three classes, 1 = white, 2 = black, 3 = other), `smoke` ($= 1$ if tobacco consumption during pregnancy, 0 otherwise), `ptl` (number of previous preterm births), `ht` ($= 1$ if hypertension history, 0 otherwise), `ui` ($= 1$ if intrauterine irritability, 0 otherwise), `ftv` (number of consultations with the gynecologist during the first pregnancy trimester), `bwt` for the weight of babies at birth.

The command `infile` can be used to import these data, indicating the list of variables that Stata will associate to each column of the external data file. The option `clear` instructs Stata to delete the existing data in the workspace before importing:

```
. infile low age lwt race smoke ptl ht ui ftv bwt using "birthwt.dat", clear
(189 observations read)
. list in 1/5
```

```
     +------------------------------------------------------------+
     | low   age   lwt   race   smoke   ptl   ht   ui   ftv   bwt |
     |------------------------------------------------------------|
  1. |   0    19   182      2       0     0    0    1     0  2523 |
  2. |   0    33   155      3       0     0    0    0     3  2551 |
  3. |   0    20   105      1       1     0    0    0     1  2557 |
  4. |   0    21   108      1       1     0    0    1     2  2594 |
  5. |   0    18   107      1       1     0    0    1     0  2600 |
     +------------------------------------------------------------+
```

In order to display the observations that were read and that are now contained in the workspace, we will use the command `list`. As previously seen, it is possible to limit the number of displayed observations ("rows") by including a filter on the numbers of observations: the option `in 1/5` instructs Stata to select only the observations ranging from 1 to 5. It is also possible to open the data viewer in order to obtain a view similar to an Excel table for these data.

1.1.7. *Variable management*

The data imported into the workspace can also be displayed by inserting the command `describe`. With the option `simple`, Stata returns the name of the variables only, whereas with the option `short` we get a summary indicating the number of observations and variables:

```
. describe, short
Contains data
  obs:           189
 vars:            10
 size:         7,560
Sorted by:
```

It is also possible to provide a list of variables, for example the variables `low`, `age` and `lwt`. These are the first three variables and the expression `describe low age lwt` can then be simplified into `describe low-lwt`:

```
. describe low-lwt
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
low             float  %9.0g
age             float  %9.0g
lwt             float  %9.0g
```

It can be seen that these three variables (weight indicator $< 2,500$ g, mother's age and mother's weight, in pounds) are manipulated as numbers. Due to economy concerns regarding memory space, the following command might be preferred:

```
. infile byte low age lwt race smoke ptl ht ui ftv bwt using "birthwt.dat", clear
(189 observations read)
```

to inform Stata to reserve less memory space for the variable `low`, which is a binary variable (the name of the variable has been prefixed by the statement `byte`).

To associate labels with variables, we will use the command `label variable`:

```
. label variable low "Weight smaller than 2.5 kg"
. describe low-lwt race
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
low             byte   %8.0g                  Weigh less than 2.5 kg
age             float  %9.0g
lwt             float  %9.0g
race            float  %9.0g
```

The variable `race`, although categorical, is represented as a number by Stata (`float`), and it is possible to verify its values with the command `tabulate` that provides a frequency table:

```
. tabulate race
       race |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         96       50.79       50.79
          2 |         26       13.76       64.55
          3 |         67       35.45      100.00
------------+-----------------------------------
      Total |        189      100.00
```

The modalities or levels of the categorical variables assumed as numbers can be associated with labels, which facilitates the reading of the descriptive graphs and summary tables. To this end, it is necessary to define, as a first step, the correspondence between the values of the variable and the labels (`label define`), then, in a second step, to associate these labels to the variables (`label values`). Here follows how to proceed with the variables `race`, `ht` and `ui`:

```
. label define yesno 0 "No" 1 "Yes"
```

```
. label define ethn 1 "White" 2 "Black" 3 "Other"
. label values ht ui yesno
. label values race ethn
```

The usage of `label define` is rather straightforward: the name of the label that will serve as a reference is given and each value is associated with a description in the form of characters (0 is here associated with ‘‘No’’). Similarly, for `label values`, the variables are indicated followed by the reference created with `label define`:

```
. tabulate race
       race |      Freq.     Percent        Cum.
------------+-----------------------------------
      White |         96       50.79       50.79
      Black |         26       13.76       64.55
      Other |         67       35.45      100.00
------------+-----------------------------------
      Total |        189      100.00
```

1.1.8. *Converting a numerical variable into a categorical variable*

When the bounds of the class intervals that must be considered are known, the command `egen cut` can be utilized, indicating the lower bounds of the intervals. Here follows a usage example:

```
. egen lwt3 = cut(lwt), at(70,120,170,220,270)
. tabulate lwt3
       lwt3 |      Freq.     Percent        Cum.
------------+-----------------------------------
         70 |         75       39.68       39.68
        120 |         93       49.21       88.89
        170 |         17        8.99       97.88
        220 |          4        2.12      100.00
------------+-----------------------------------
      Total |        189      100.00
```

The command `egen` constitutes an extension of the command `generate` allowing the creation of new variables, but accepting a certain number of options (acting most of the time as functions that enable performing calculations on a given variable). See online help, `help egen`, for more information and in particular the list of functions available (`count`, `iqr`, `max`, etc.).

Although all of the bounds of the intervals have been explicitly specified, it would also be possible to write `at(70(50)270)` to inform Stata to build a sequence of values

ranging from 70 to 270 with increments of 50. Stata can also automatically build more or less balanced groups with the option `group(4)`.

In order to build classes based on quartiles or on deciles, the command `xtile` should be employed instead, specifying in the option `nq()` the number of desired groups:

```
. drop lwt3
. xtile lwt3 = lwt, nq(4)
. tabulate lwt3
4 quantiles |
     of lwt |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         53       28.04       28.04
          2 |         43       22.75       50.79
          3 |         46       24.34       75.13
          4 |         47       24.87      100.00
------------+-----------------------------------
      Total |        189      100.00
```

1.2. Descriptive univariate statistics and estimation

1.2.1. *Summarizing a numerical variable*

The command `summarize` provides a four-point numerical summary (average, standard deviation, minimum and maximum) for one or more numerical variables. The option `detail` provides a more comprehensive summary, notably including the five most extreme values, different quantiles and the indicators of skewness and kurtosis indicators of the distribution of the variable:

```
. summarize bwt
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         bwt |       189    2944.587    729.2143        709       4990
```

To obtain a 95% confidence interval based on the approximation by the normal distribution, we will insert the command `ci` followed by the name of the variable of interest:

```
. ci bwt
    Variable |        Obs        Mean    Std. Err.       [95% Conf. Interval]
-------------+---------------------------------------------------------------
         bwt |        189    2944.587    53.04254        2839.952    3049.222
```

The command `mean bwt` will provide the same result.

To build a histogram of frequencies or frequency counts, the command to employ is `histogram`. The option `frequency` should be included when willing to work with frequency counts, or `percent` for proportions (Figure 1.1):

```
. histogram bwt, frequency
(bin=13, start=709, width=329.30769)
```

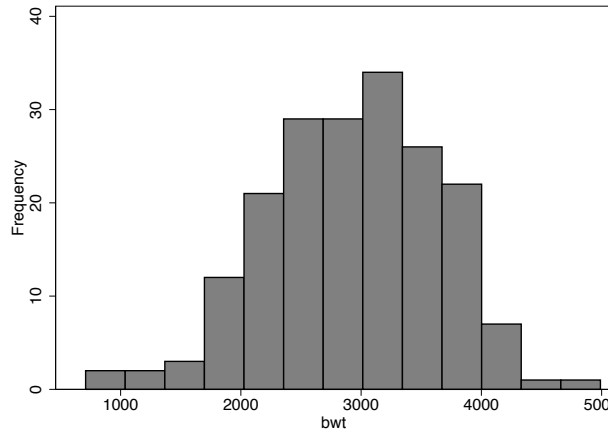

Figure 1.1. *Frequency counts histogram for the weight at birth*

The option `bin()` make it possible to change the number of class intervals used by Stata. When willing to display a nonparametric density curve, we will add the option `kdensity`. There is also an option `discrete` that assimilates the histogram to a representation in the form of a bar plot in which are represented on the x-axis, the unique values of the numerical value, without consideration of class intervals.

1.2.2. *Summarizing a categorical variable*

The command `tabulate` (which can be simplified in `tab`) provides a frequency analysis table for a variable:

```
. tabulate race, plot
       race |      Freq.
------------+------------+----------------------------------------------------
      White |         96 |**************************************************
      Black |         26 |**************
```

```
      Other |         67 |**********************************
------------+------------+----------------------------------------------------
      Total |        189
```

`tab1` is preferable when we want to build (univariate) frequency counts tables for several variables, with the following syntax:

```
. tab1 race ht ui
```

As in the case of numeric variables, the command `ci` (or `prop`) provides confidence intervals for a proportion. The option `binomial` will be added if it is desirable to use the binomial distribution to build 95% confidence intervals:

```
. ci low, binomial
                                                        -- Binomial Exact --
    Variable |        Obs        Mean    Std. Err.       [95% Conf. Interval]
-------------+---------------------------------------------------------------
         low |        189    .3121693    .0337058        .2468886    .3834546
```

The option `level()`, as in most of the commands related to statistical tests of null hypothesis or to interval-based estimation, allows changing the degree of significance associated with the confidence intervals. For example, `level(90)` instructs Stata to return 90% confidence intervals, instead of 95%, which is the value by default.

There exist several ways to graphically represent frequency counts or frequency distributions. Hereafter follows a solution for the variable `race` that is based on the use of the command `graph bar` (Figure 1.2). The idea consists of creating an auxiliary variable in which the counts associated with each category of `race` are accumulated:

```
. gen freq = 1
. graph bar (sum) freq, over(race) ytitle("Ethnicity")
```

The instruction `bar` will be replaced by `hbar` to represent the bars horizontally instead of vertically. Another solution is to use the command `histogram` with the option `discrete`.

1.3. Bivariate descriptive statistics

1.3.1. *Describing a numeric variable by the levels of a categorical variable*

The command `summarize` operates in a univariate manner only, that is for each listed variable. When a numeric variable has to be summarized for each level of a

categorical variable, a selection option by can be used, which has to be placed at the beginning of the command:

```
. by low, sort: summarize lwt
-------------------------------------------------------------------------------
-> low = 0

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         lwt |       130       133.3    31.72402         85        250

-------------------------------------------------------------------------------
-> low = 1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         lwt |        59    122.1356    26.55928         80        200
```

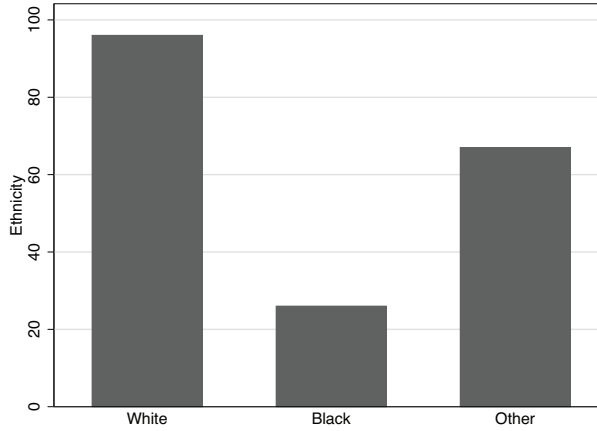

Figure 1.2. *Bar diagram for the distribution of counts of the variable* `race`

It should be noted that the data must be sorted first, hence the addition of the option `sort`. An alternative consists of directly using `bysort`:

```
. bysort low: summarize
```

Care should be taken not to confuse : with , when by is specified before a command.

It may happen that only certain statistics have to be calculated, for example the mean and the standard deviation. In this case, the command `tabstat` is simpler to use. Here is an example of its usage:

```
. tabstat lwt, by(low) stats(mean sd) format(%6.2f)
Summary for variables: lwt
     by categories of: low (Weight less than 2.5 kg)

     low |      mean        sd
---------+--------------------
       0 |    133.30     31.72
       1 |    122.14     26.56
---------+--------------------
   Total |    129.81     30.58
------------------------------
```

The option `format(%6.2f)` makes it possible to limit the display to two decimal places.

An alternative formulation which can be generalized to several classification factors (potentially using different statistics, for example the average for a variable and the median for another), consists of using the command `table`. The equivalent to the option `stats()` of `tabstat` is here `contents()` and what is to be calculated is specified therein: `freq` corresponds to the count by the level of the classification variable `low`, `mean lwt` corresponds to the average of the variable `lwt` for each level of the classification factor, etc.:

```
. table low, contents(freq mean lwt sd lwt) format(%6.2f)
----------------------------------------------
Weight     |
less  |
than 2.5   |
kg         |      Freq.   mean(lwt)     sd(lwt)
-----------+----------------------------------
         0 |      130.00      133.30       31.72
         1 |       59.00      122.14       26.56
----------------------------------------------
```

In order to represent the average weight of mothers for underweight children or within standards, a dot plot (or even a bar plot, as in the previous case with the variable `race`) is appropriate: the classification factor is indicated in an option `over()`, which allows overlaying graphical elements within the same graph as illustrated in Figure 1.3:

```
. graph dot lwt, over(low)
```

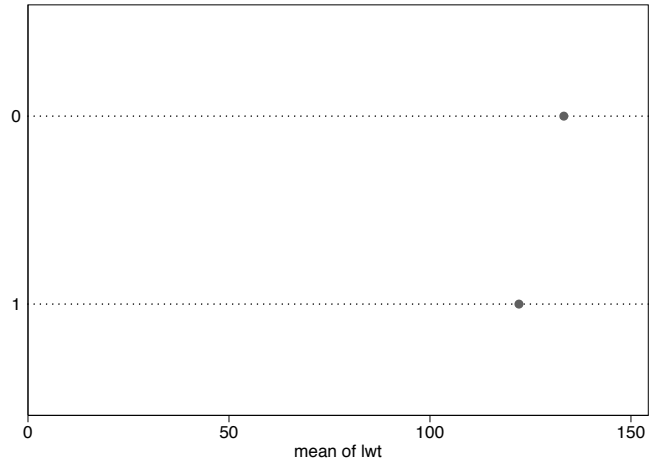

Figure 1.3. *Dot chart for the average values of* `lwt` *conditionally to the variable* `low`

By default, it is the average of the numeric variable (`lwt`) that is considered, but another statistic can be used by inserting the option `(stats)`. For example, when the median weight must be displayed, the previous command will be replaced by:

```
. graph dot (median) lwt, over(low)
```

In some cases, it may happen that only a particular summary statistic is of interest. For example, suppose that we want to calculate the maximal weight of the babies (in grams) in the two groups of individuals defined by the variable `ui` (0, no intrauterine pain during pregnancy; 1, intrauterine pain). In this case, it is possible to store these two values in variables known as "local", by making use of the instruction `scalar`. In order to calculate and display the maximum weight observed in babies in the group of mothers presenting no intrauterine pain, the following commands should be used:

```
. drop maxbwt
. quietly: summarize bwt if ui == 0
. scalar maxbwt0 = r(max)
. display maxbwt0
4990
```

In fact, we use the results generated by the command `summarize` but recorded temporarily in an invisible manner by Stata (they can be accessed by typing `return list`, just after typing `summarize bwt if ui == 0`). The content of interest is retrieved by inserting the command `r()` and it is then possible to display the results

with `display`. The addition of the prefix `quietly:` to the command `summarize` ensures that the results are not displayed (but that they still remain available).

1.3.2. *Describing two qualitative variables*

The command `tabulate` makes it possible to build contingency tables when it is invoked with a list of two variables. For example, the following command enables crossing the levels of the variables `low` (row-wise) and `smoke` (column-wise) and presents the relative frequencies calculated row-wise:

```
. tabulate low smoke, row
+----------------+
| Key            |
|----------------|
|   frequency    |
| row percentage |
+----------------+

  Weight   |
 less than |         smoke
   2.5 kg  |         0          1 |     Total
-----------+----------------------+----------
         0 |        86         44 |       130
           |     66.15      33.85 |    100.00
-----------+----------------------+----------
         1 |        29         30 |        59
           |     49.15      50.85 |    100.00
-----------+----------------------+----------
     Total |       115         74 |       189
           |     60.85      39.15 |    100.00
```

The other options, `col` and `cell`, allow the calculation of the relative frequencies in a columnwise manner or relatively to the total number of the observations (conditional frequencies).

1.4. Key points

– Stata represents data as a list of variables, similar to a data table in which the variables are arranged in columns, all having the same number of observations, and the values of the variables are generally numbers that can be associated with labels when they refer to the modalities of a categorical variable.

– The commands `summarize` and `tabulate` provide a univariate descriptive summary in the case of numeric and categorical variables, whereas `tabstat` and `tabulate` allow working in a bivariate manner.

– The main graphics commands to represent the distribution of a numeric or categorical variable are `histogram` and `graph bar` (bar plot) or `graph dot` (dot plot).

1.5. Further reading

Stata is a well-known software program to facilitate database management. For more information, the manual "[D] data management", accessible in PDF format or by using the command `help data management`, provides an accurate description of all the controls related to the management and manipulation of variables.

With regard to the production of graphs in Stata, it is strongly recommended to consult the work of Mitchell [MIT 12], which provides a detailed description of each type of graph with the most common options. The Stata website also offers an overview of the different charts available depending on the type of variables manipulated: http://www.stata.com/support/faqs/graphics/gph/stata-graphs/.

Finally, for all aspects related to automation or programming with Stata, the reader is invited to refer to the book by Baum [BAU 16].

1.6. Applications

1) The plasma viral load is used to describe the amount of virus (for example HIV) in a blood sample. This viral marker that allows following the progression of the infection and measuring the effectiveness of treatments represents the number of copies per milliliter, and most measurement instruments have a detectability threshold of 50 copies/mL. Here follows a series of measurements, X, expressed in logarithms (base 10) collected on 20 patients:

```
3.64 2.27 1.43 1.77 4.62 3.04 1.01 2.14 3.02 5.62 5.51 5.51 1.01
1.05 4.19 2.63 4.34 4.85 4.02 5.92
```

As a reminder, a viral load of 100,000 copies/mL is equivalent to 5 log.

– indicate how many patients have a viral load considered as non-detectable;

– the researcher realizes that the value 3.04 corresponds to a data entry error and must be changed to 3.64. Similarly, she has a doubt about the seventh measurement and decides to consider it as a missing value: perform the corresponding transformations;

– what is the median viral load level in copies per milliliter, for the data considered as valid?

Stata provides a data editor that comes in the form of a spreadsheet program (similar to Excel), but data can be directly recorded by utilizing the command `input`. After having given the name of the variables (when there are several variables, their name has to be separated with a space), the user presses the Enter button and the observations are input (similarly, when there are several variables the observations or observed values are typed in separated by a space). When there are no more data to be entered, the user can type the word `end` and press Enter to inform Stata that data input is completed. It is also possible to use the data editor, but in this case the user will have to rename the variable (by default `var1`).

The detection threshold in logarithm is:

```
. display log10(50)
```

It is therefore possible to count the number of observations that do not verify the condition $X > \log(50)$ by employing the command `count`:

```
. count if X <= log10(50)
```

Hence, the calculation of the median viral load considering only the data above the detection threshold:

```
. egen Xm = median(10^X) if X > log10(50)
. display round(Xm)
```

To calculate the median of X verifying the condition $X > \log(50)$, the command `egen` has been directly used that provides a certain number of transformations and basic computing functions (see the online help, `help egen`).

2) The file `anorexia.dat` contains data from a clinical study in anorexic patients who have undergone one of the three following therapies: behavioral therapy, family therapy and control therapy [HAN 93].

– how many patients are there in total? How many patients are there per treatment group?

– the weight measures are in pounds. Convert them into kilograms;

– create a new variable containing differences scores (`After - Before`);

– indicate the mean and the range (min/max) of the difference scores per treatment group.

The data contained in the file `anorexia.data` is read in the form of the overview provided below:

```
Group Before After
g1 80.5  82.2
g1 84.9  85.6
g1 81.5  81.4
g1 82.6  81.9
g1 79.9  76.4
```

we will provide the command `infile` with a "dictionary" file that describes the structure of the data. This file usually has the same name as the data source file, and has the extension `dct`. Here is the complete listing (`anorexia.dct`):

```
infile dictionary using anorexia.dat {
  _first(2)
  str2 Group "Therapy type"
  double Before "Before"
  double After "After"
}
```

Therein, we indicate that the observations begin on the second row (ignoring the header row), and that there are three variables, `Group` (qualitative variable), `Before` and `After` (numerical variables). Note that description labels have been associated for these three variables. It then suffices to just make use of the command `infile` by providing the name of the dictionary file (there is no need to specify the file extension):

```
. infile using anorexia
```

We can then verify that the data have been correctly imported by entering `describe`. This command also provides the number of observations available in the data table ($N = 72$):

```
. describe
```

To obtain the frequency counts by therapy type, a simple table of counts is produced by inserting the command `tabulate`:

```
. tabulate Group
```

The conversion of units for the weights does not raise any specific problem, but a decision must be made whether new variables have to be created (`generate`) or if the existing values are to be replaced (`replace`). Here, the existing values will be replaced:

```
. replace Before = Before/2.2
. replace After = After/2.2
```

For differences scores, this time a new variable is created by making use of the command `generate`:

```
. generate diff = After - Before
. summarize diff
```

The command `summarize` can be utilized to provide a numeric summary for each of the treatment groups, for example `by Group, sort: summarize diff`. However, to specifically calculate some descriptive indicators, it is more convenient to employ the command `tabstat` to which the response variable and the classification factor are provided, as well as the desired statistics through `stats()`:

```
. tabstat diff, by(Group) stats(mean min max)
```

3) A quantitative variable X takes the following values with a sample of 26 individuals:

```
24.9,25.0,25.0,25.1,25.2,25.2,25.3,25.3,25.3,25.4,25.4,25.4,25.4,
25.5,25.5,25.5,25.5,25.6,25.6,25.6,25.7,25.7,25.8,25.8,25.9,26.0
```

– calculate the mean, the median and the mode of X;

– what is the value of the variance estimated from these data?

– assuming that data are grouped into four classes whose bounds are: 24.9–25.1, 25.2–25.4, 25.5–25.7, 25.8–26.0, display the distribution of the figures by class in the form of a frequency counts table;

– represent the distribution of X as a histogram, without consideration of *a priori* class intervals.

Concerning data entry, we can simply enter the data in a text format file. Suppose that the data have been entered in a file called `saisie_x.txt` of which an overview follows here:

```
24.9 25.0 25.0 25.1 25.2 ...
```

The values of X are simply separated by a space. In this case, the data can be read and imported into Stata with the command `infile`:

```
. infile x using "saisie_x.txt"
```

The option `clear` has to be added if data are already stored in the Stata workspace.

The command `tabstat` can be used to calculate the average and the median of the observations. However, the command `egen` must be included with the function `mode` to calculate the modal values of X. Since there are several modes, Stata will be instructed to return the lowest value of the two modes. Note that the command `egen`

could also be employed to calculate the average and the median of X:

```
. tabstat x, stats(mean median)
. egen xmode = mode(x), minmode
. display xmode
```

Concerning the variance, `summarize x` provides direct access to the standard deviation, but it can also be calculated with `egen` as the square of the standard deviation estimated from the sample, and then displayed in the console of the results:

```
. egen varx = sd(x)
. di varx^2
```

A more elegant alternative consists of exploiting the results returned by a statement such as `summarize x`: in this case, `display r(Var)` provides the expected result.

Regarding the discretization of `x` into four predefined class intervals, the function `cut` can always be used along with `egen`. It can then be verified that the class intervals are properly respected by employing `table,` which is more flexible than the command `summarize` and makes it possible to specify the list of statistics values to be displayed:

```
. egen xc = cut(x), at(24.8,25.2,25.5,25.8,26.1) label
. table xc, contents(min x max x)
```

On the other hand, the frequency counts table can be directly obtained with `tabulate`, for example:

```
. tabulate xc, plot
```

Finally, Stata relies on a specific algorithm to determine the optimal number of classes to use in an histogram, just like R. Everything is managed from the command `histogram`:

```
. histogram x, frequency
```

The option `frequency` should not be forgotten when intending to display the counts instead of the density (which is the default choice).

4) The file `elderly.dat` contains the size, measured in centimeters, of 351 elderly females, randomly selected from the population during a study on osteoporosis. A few observations are however missing.

– how many missing observations are there in total?

– give a 95% confidence interval for the average size in this sample, using a normal approximation;

– represent the distribution of the sizes observed in the form of a density curve.

To import the data stored in a simple text file, the command `infile` is employed:

```
. infile tailles using "elderly.dat", clear
```

Here, it should be noted that the manner in which the missing values are coded is not specified because the "." is the default format by Stata (for numeric variables only).

There are several commands that facilitate the detection and the identification of the missing values. Among those available by default in Stata, `codebook` can be distinguished, that provides a summary of the variable, as well as the basic functions that allow the tabulation or counting the observations that meet a certain criteria:

```
. count if sizes == .
```

To obtain the average size and its associated 95% confidence interval, the following command has to be entered:

```
. ci sizes
```

Finally, to display the distribution of the sizes in the form of a density curve, the command `histogram` is inserted with the option `kdensity`. The degree of smoothing can be controlled with the option `kdenopts`; for example, adding `kdenopts(gauss width(1))` to the following command would produce a "less smooth" curve (that is to say, fitting better to the data):

```
. histogram sizes, kdensity
```