

SPSS Tutorial 3: Basic Tools

Goals

- To learn some of the basic tools you will need to complete the assignments. For example:
 - how to do addition, subtraction, multiplication & division
 - how to compute squares and square roots
 - how to get the sum of a column of numbers
 - how to get the number of scores in a column
- To compute means using these basic tools

Preamble

When I was first learning statistics in the mid 1980s, we used hand calculators for all (or most) of our calculations. In those days, it was common for textbook authors and lecturers to proceed as follows. They would:

- Present the **conceptual** formula for some statistic (e.g., the conceptual formula for the variance)
- Point out that if you used the conceptual formula to calculate the statistic in question, you would often end up with an inaccurate answer due to rounding error
- Give you a **computational** formula for the same statistic that allowed you to avoid the rounding error, but provided little or no conceptual insight into the statistic.

The problem with this approach was that most of us students became very familiar with the computational formulae, but were not so well acquainted with the conceptual formulae. Putting it another way, we damn well knew how to *calculate* all of the various statistics, but we did not always have a great deal of insight into what they meant.

As computers and statistical software packages have become more available to students in introductory and intermediate stats classes, there has been a definite shift away from the presentation of computational formulae in textbooks and lectures. The common refrain nowadays is that we should *let the computer do the computations*.

But notice that if this is our approach (i.e., letting the computer do the computations), all we have done to the old sequence of events is replace use a computational formula on a hand calculator with use of a computer. The point is that *neither of these activities provides any great conceptual insight into the statistic you are calculating*.

What lecturers and textbook authors *ought* to be doing instead, and what we will be doing in this course, is this:

- We will first ask you perform the calculations using the **conceptual formula**, not with a hand calculator, but on the computer, where you can carry enough decimal places to avoid rounding error.

- Then, after you have had some practice with the conceptual formula (which we hope will help you to gain conceptual understanding), you will learn how to obtain the same results using the standard functions and procedures provided in SPSS.

Reading in the Data

For this tutorial, we will use the data from Problem 2 on page 67 of *Bare Essentials* (2nd Edition). You can read the data into and assign variable labels using the following syntax. (NOTE: To save keystrokes, you can copy the syntax from this document and paste it into your syntax file.)

* Data from Bare Essentials, Page 156, Table 17-1.

```
DATA LIST LIST /subj grp pretest posttest (4f3.0).
BEGIN DATA.
1      1      22      16
2      1      24      17
3      1      32      25
4      1      24      21
5      1      35      32
6      1      27      22
7      1      34      27
8      1      15      13
9      1      29      25
10     1      25      21
11     2      32      31
12     2      33      34
13     2      42      34
14     2      27      24
15     2      22      24
16     2      18      15
17     2      16      13
18     2      32      29
19     2      25      19
20     2      24      22
END DATA.
```

```
var lab
  subj      'Subject Number'
  grp       'Group'
  pretest   'Pre-test score'
  posttest  'Post-test score'.
```

Basic Arithmetic Operations

The basic arithmetic operations can all be done using the **COMPUTE** statement.

Addition. There are a couple of ways to obtain the sum of 2 or more numbers (across columns within a row). Type the following commands into your syntax window, run them, then look at your data file.

```
compute sum1 = pretest + posttest.  
compute sum2 = sum(pretest,posttest) .  
exe.
```

Your data file should now have a new variables SUM1 and SUM2, both of which show the sum of the pre- and post-test scores.

Subtraction. Table 17-1 in Bare Essentials has a Difference column which shows the pre-test minus post-test difference. You may compute that same difference score using the following syntax.

```
compute diff = posttest - pretest.  
exe.
```

Multiplication. The symbol for multiplication in SPSS is an asterisk (*), as the following compute statement illustrates.

```
compute product = pretest * posttest.  
exe.
```

Division. The division operator in SPSS is a slash (/). For example:

```
compute halfpre = pretest/2.  
compute halfpost = posttest/2.  
exe.
```

Integer division. There may be times when you need to perform integer division—i.e., when you want the quotient (a whole number) with a remainder. You can obtain the quotient using the TRUNC function (i.e., truncate), and the remainder using the MOD function. For example, the following lines compute the quotient and remainder for PRETEST divided by 3:

```
compute quotient = trunc(pretest/3) .  
compute remaind = mod(pretest,3) .  
exe.
```

The following variables are from the first 3 lines of my data file:

SUBJ	PRETEST	QUOTIENT	REMAIND
1	22	7.00	1.00
2	24	8.00	.00
3	32	10.00	2.00

These lines show that:

- 22 divided by 3 = 7 with 1 left over
- 24 divided by 3 = 8 with 0 left over
- 32 divided by 3 = 10 with 2 left over

Computing squares, square roots, etc

As you may have noticed, squaring is a very common operation in statistics (e.g., you need the sum of the squared deviations about the mean when calculating the variance). Earlier, you learned to use '*' as the multiplication operator in SPSS. To raise a number to some power, you simply use two asterisks. So for squaring, you raise a number to the power of 2, as follows:

```
compute sumsq = sum1**2.  
exe.
```

To compute the cube of variable SUM1:

```
compute sumcubed = sum1**3.  
exe.
```

There are two ways to compute the square root of X in SPSS. The first way is to raise X to the power of 0.5, like this:

```
compute sqrt1 = sumsq**0.5.  
exe.
```

The nice thing about this method is that it generalizes to other roots. For example if you wanted the cube root of X, you would use something like this: `compute cuberoot = X**(1/3)`.

The second way to obtain square roots is with the SQRT function:

```
compute sqrt2 = SQRT(sumsq).  
exe.
```

Your data file should have the same values for variables SQRT1 and SQRT2.

Putting the pieces together: Computing means within rows

You now have the tools you need to compute a mean of the pre- and post-test scores for each subject. Give it a try, bearing in mind the order of operations.¹

Creating COMPUTE Statements via the Pull-down Menus

When the SPSS Data Window is active, you can find the pull-down menu for **COMPUTE** by clicking on **Transform→Compute**. This brings up a dialog box that has variable list on the left, and a list of all of the available functions on the right. For common transformations, you will eventually find it is quicker and easier to type the command directly. However, you may wish to

¹ Multiplication and division take precedence over addition and subtraction. For example, $a+b/c$ indicates that a should be added to the result of b/c ; but $(a+b)/c$ indicates that the sum of a and b is divided by c.

use the pull-down menu when using some of the more complicated or less familiar functions. Of course, you should always exit the **COMPUTE** dialog box by clicking **Paste** rather than **Okay**.

Getting the Sum and N for a Column of Numbers

To this point, we have been using COMPUTE statements, which allow us to compute sums, differences, squares, and so on across columns, *within rows* (e.g., you computed the mean of the pre- and post-test scores for each subject). We have not yet seen how to obtain the sum and N for a column in the SPSS data editor—and we will have to know how to do that if we are going to calculate means, variances, standard deviations etc, using the conceptual formulae.

In Tutorial 1, you learned how to use the MEANS procedure (**Analyze**→**Compare means**→**Means**). Use the MEANS procedure now to obtain the SUM and COUNT for variables PRETEST and POSTTEST. (Hint: You will need to add SUM to the default list of statistics generated by MEANS.) Use variable GRP as the independent variable. The output should look something like this:

Report

Group		Pre-test score	Post-test score
1	Mean	26.70	21.90
	N	10	10
	Std. Deviation	6.075	5.646
	Sum	267	219
2	Mean	27.10	24.50
	N	10	10
	Std. Deviation	7.824	7.472
	Sum	271	245
Total	Mean	26.90	23.20
	N	20	20
	Std. Deviation	6.820	6.582
	Sum	538	464

Using the SUMS and Ns to obtain Means

You now have the pieces you need to compute pre-test and post-test means for each group separately, and for the entire sample. You also have the tool you need to carry out these computations: the COMPUTE statement.

The pre- and post-test means for the entire sample can be calculated as follows:

```
compute mean1 = 538/20.          /* Pre-test mean (overall) .
compute mean2 = 464/20.          /* Post-test mean (overall) .
exe.
```

We should also add labels to the new variables MEAN1 and MEAN2 so that we know what they are if we ever come back to this file.

```
var lab
  mean1      'Pre-test mean (overall)'
  mean2      'Post-test mean (overall)' .
```

Computing Group Means

Computation of the group means is slightly more complicated, because the Sums and Ns may be different for the various groups. Therefore, we need to use IF statements. There are a couple of ways to do this. The easiest way is as follows:

```
if (grp = 1) mean1.gr = 267/10.      /* Pre-test mean for Group 1 .
if (grp = 1) mean2.gr = 219/10.      /* Post-test mean for Group 1 .
if (grp = 2) mean1.gr = 271/10.      /* Pre-test mean for Group 2 .
if (grp = 2) mean2.gr = 245/10.      /* Post-test mean for Group 2 .
exe.
```

Again, we should add variable labels.

```
var lab
  mean1.gr    'Pre-test mean (group)'
  mean2.gr    'Post-test mean (group)' .
```

The series of 4 IF statements shown above will certainly get the job done. But it is not very efficient, because each statement entails a pass through the entire data file. A better, more efficient approach is to use a DO IF structure, such as this:

```
do if (grp=1).
-   compute mean1.gr = 267/10.      /* Pre-test mean for Group 1 .
-   compute mean2.gr = 219/10.      /* Post-test mean for Group 1 .
else if (grp=2).
-   compute mean1.gr = 271/10.      /* Pre-test mean for Group 2 .
-   compute mean2.gr = 245/10.      /* Post-test mean for Group 2 .
end if.
exe.
```

This method requires only one pass through the data file, and so is 4 times more efficient. In a small file such as the one we are using, this is not a big deal. But suppose you were working with a large file of several thousand records. In that case, you would certainly want to use the more efficient approach.

Be sure to save your syntax file, because you may want to refer to it when working on other tutorials or assignments.