SPSS syntax for data set definition

Johan A. Elkink

January 31, 2013

1 Introduction

For replication purposes, it is crucial to keep proper documentation of all the steps taken in a particular analysis. The recommended approach is always to keep the data sets exactly as downloaded from your data source, or as coded by the original version of the data entry, and keep a record of all changes made since. When you need to redo the analysis, you can simply use the same commands on the same original files and you will get the same results. This way any change subsequently made at any point in the analysis will automatically be properly recorded and automatically affect all subsequent steps accordingly.

While SPSS has an excellent trick to keep track of those steps - using the "Paste" instead of the "Ok" button on every dialogue screen - this trick is not always available. One area where it is not, is where settings are changed directly in the "Variable screen". These steps can be properly taken through SPSS syntax,[1] however, by typing the commands directly into the syntax screen.

Note that under the "Help" menu in SPSS, the syntax reference guide (*PASW Statistics 18 command syntax reference* 2007) can be accessed, on which most of this note is also based. Furthermore, note the use of the period at the end of each command. Every command in SPSS ends on a period. Some commands do not run until the command EXECUTE is given, so you might want to end the syntax file standard with:

```
EXECUTE.
```

Comments can be added to the syntax file by using the asterisk symbol at the start of the line and a period at the end.

2 Commands for creating variables

Generating a new dataset

If you have a very small data set you can type in directly, the best command to use is DATA LIST. This command can be used to read in raw data in a text file or to type in data right into the syntax file. For example, to read data from the text file `states.txt` we can use:

```
DATA LIST FILE="states.txt" FIXED
/id 1-3 age 4-6 sex 7.
```

[1] Throughout, the terms "syntax" and "command" will be used interchangeably.

This assumed that there is this data file, with fixed locations on each line for the data. The first three characters on each line contains a respondent identification number, the next three characters the age and the last character the sex. For example, the `states.txt` file could look like:

```
0010201
0020312
0030911
0040451
...
```

Alternatively, one can enter the data directly into the syntax file, using a DATA block like this:

```
DATA LIST /id 1 age 3-4 sex 6.
BEGIN DATA
1 20 1
2 31 2
3 91 1
4 45 1
END DATA.
```

Note the locations of the punctuation in the syntax. If you clearly have columns using whitespace such as in this example, it is not actually necessary to list the exact column locations. An alternative is:

```
DATA LIST FREE /id age sex.
BEGIN DATA
1 20 1
2 31 2
3 91 1
4 45 1
END DATA.
```

The addition of FREE implies that you will not define the exact locations.

The DATA LIST command is an extensive one and I will refer to *PASW Statistics 18 command syntax reference* (2007: 545-562) for further details and examples.

Adding a new variable

The syntax for creating a new variable depends on the type of variable. For numerical variables, the command is COMPUTE, while for string variables it is STRING. A string variable is a variable to store verbatim text, which is generally not very useful for statistical analysis. Variables that are categorical are stored as numerical variables, with labels attached to each value. Therefore, you will generally need the COMPUTE and not the STRING command. You could for example create a new variable by first setting all values to 0:

```
COMPUTE interest = 0.
```

And then replace with more appropriate values. COMPUTE can also be used to use information from other variables, e.g.:

```
COMPUTE total_score = score1 + score2 + score3 + score4.
```

Naming a data set

You can give the current data set a name, for example to call is "INES" (Irish National Election Study):

```
DATASET NAME INES
```

This will allow to activate this data set later prior to running commands affecting this data set:

```
DATASET ACTIVATE INES
```

Adding variable labels

Variable names are often short and it is easy to subsequently forget what each variable meant. Often, unfortunately, variable names are very cryptic, for example named after the question number in a survey or, even worse, sequentially throughout the file (V1, V2, V3, etc.). Both are not recommended - it is better to use easy to remember names. Nevertheless, in all cases one should add more descriptive variable labels to avoid any future confusion. This can be done with the VARIABLE LABELS command. For example:

```
VARIABLE LABELS
sex 'Sex of respondent'
vote 'Vote choice in 2004 Parliamentary Elections'
turnout 'Turned out to vote in 2004 Parliamentary Elections'
relig 'Level of religiosity of respondent'.
```

Again, not the period at the end (only). The location of the whitespace (spaces, tabs, end-of-line) is irrelevant for the interpretation of the command. It is advisable to create some format that is "easy on the eye", so that it is easy to get an overview of the variables that are being named.

To check the success of this command, you can use the DISPLAY LABELS command to get a list of variable labels.

Adding value labels

For most categorical and some scale variables, it is useful to properly label each category. You do not want to have to figure out retrospectively whether on a variable called "sex", "1" was male or "2". So we can do, for example:

```
VALUE LABELS sex
1 'male'
2 'female'.
```

You can add identical value labels to a set of variables, like this:

```
VALUE LABELS hunting fishing rowing
1 'strongly agree'
2 'agree'
3 'neither disagree nor agree'
4 'disagree'
5 'strongly disagree'.
```

Missing values

For most real life data, we have specific observations where, on specific variables, we have no data. For example, some respondents might have refused to give their age or voting preference. It is useful to have separate categories for missing data, which are then subsequently ignored by any analysis in SPSS. To make clear to SPSS what the missing value categories are, one can do, for example:

```
MISSING VALUES sex (9).
```

This will define the value "9" as representing missing information on the respondent's sex.

Printing a codebook

To get an overview of all variables, their values, their missing values, etc., you can print a codebook for the entire data set with the simple command:

```
CODEBOOK
```

See *PASW Statistics 18 command syntax reference* (2007: 322-329) for the various options to this command.

3 Example

On the lecture slides we had the following exercise:

	System	D.Magnitude	Seats	Prop	EU.Member
1	PR	10	80	0.8	Yes
2	PR	150	150	0.9	No
3	STV	9	100	0.8	Yes
4	FPTP	1	300	0.4	Yes
5	FPTP	1	600	0.5	No
6	PR	3	200	0.7	Yes

- Variable names and labels
- Definition of categories and missing values
- Definition of measurement level
- Production of codebook

The answer to this then becomes:

```
* Create a data set and call it ElectSyst.

DATA LIST FREE / id system magnitude seats proportionality EUmember.
BEGIN DATA
1 1  10  80 .8 1
2 1 150 150 .9 0
3 2   9 100 .8 1
4 3   1 300 .4 1
5 3   1 600 .5 0
6 1   3 200 .7 1
END DATA.

DATASET NAME ElectSyst.

* Create variable labels.

VARIABLE LABELS
id              'Identification'
system          'Electoral system'
magnitude       'Average district magnitude'
seats           'Total number of seats in parliament'
proportionality 'Disproportionality index'
EUmember        'Is EU member?'.

* Create value labels where necessary.

VALUE LABELS system
1 'PR'
2 'STV'
3 'FPTP'.
VALUE LABELS EUmember
1 'Member'
0 'Not a member'.

* Define missing values (not really necessary in this, complete data set,
* but by way of example).

MISSING VALUES id system proportionality EUmember (9).
MISSING VALUES magnitude seats (999).

* Define levels of measurement.

VARIABLE LEVEL
 system EUmember (NOMINAL)
```

```
/magnitude seats proportionality (SCALE).

* Compute a variable based on existing ones.

COMPUTE districts = seats / magnitude.
VARIABLE LABELS
districts       'Number of electoral districts'.

* Some summary information about the file.

DISPLAY LABELS.

CODEBOOK.

EXECUTE.
```

References

PASW Statistics 18 command syntax reference. 2007. Technical report SPSS Inc. Chicago, IL: .