The basics of using SPSS syntax for students

Erik van Ingen

(Version February 2011)

(If you have comments or suggestions for improvement regarding this tutorial, please send them to e.j.vaningen@uvt.nl)

Introduction

In this tutorial you will learn the basics of using SPSS syntax. It will help you to organize your analyses in a step-by-step fashion, which enables replication of your research and which will help you remember the things you did when using the syntax on a next occasion. Other advantages of using syntax are:

- Extra functions are available
- More efficient (once trained)
- Allows you to adjust a few parameters and rerun all of the analyses
- Allows you to re-use data transformations from previous projects
- Organization of your own thoughts.

The downside of using SPSS syntax is that it takes a considerable time investment to master it, and as a consequence students sometimes find it frustrating at the start. The idea of this tutorial is to make those first steps easier. By focusing on the basics of the commands, by providing a general procedure of creating syntax, and by giving some warnings about SPSS' peculiarities, most of the frustrations are avoided.

The section "The minimal syntax for the most important commands" is probably the most important part of this tutorial, which can also be used as a reference for users who have some experience in working with syntax. The "Getting additional help" is probably also useful for most users. If you have no experience in using syntax, make sure you go through the "My first SPSS session" carefully. To all users, it is highly recommended to organize your syntax files in accordance with "The basic setup of a syntax file".

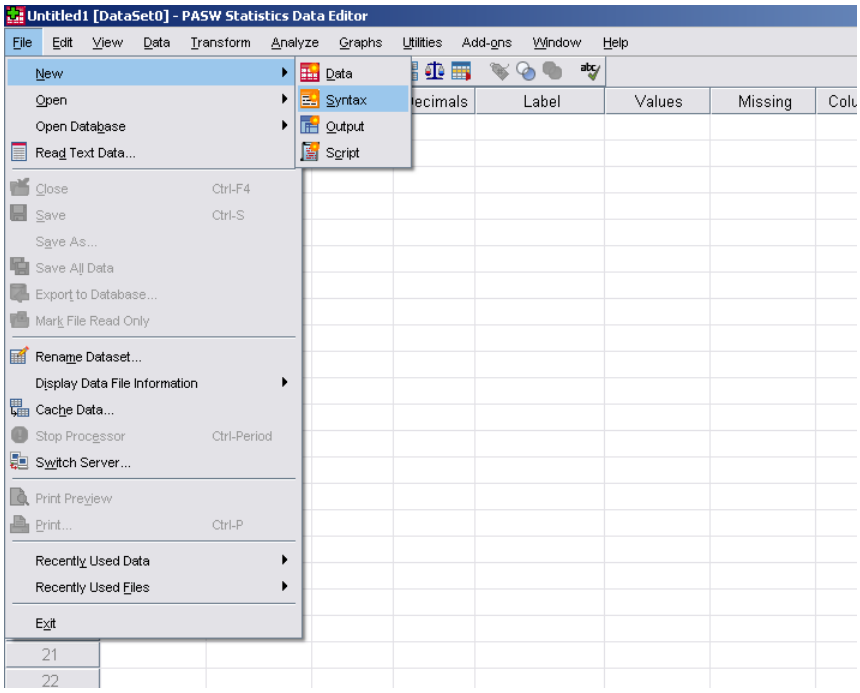
Good luck!

Note: The `Courier New font` is used to indicated SPSS syntax throughout this tutorial.

Tutorial: My first SPSS session

This tutorial uses data from the European Values Study (from the Netherlands in the year 2000). In order to complete the tutorial, you need to download these data from http://spitswww.uvt.nl/~ejvingen/spss%20tutorial.sav. Please make sure you save the file to a location where you can find it back! (If you are doing this tutorial in a computer class, a USB stick would be the most convenient storage)

When you want to start using SPSS, you run the program, and then as a first step, you open a new syntax file (click file – new – syntax).

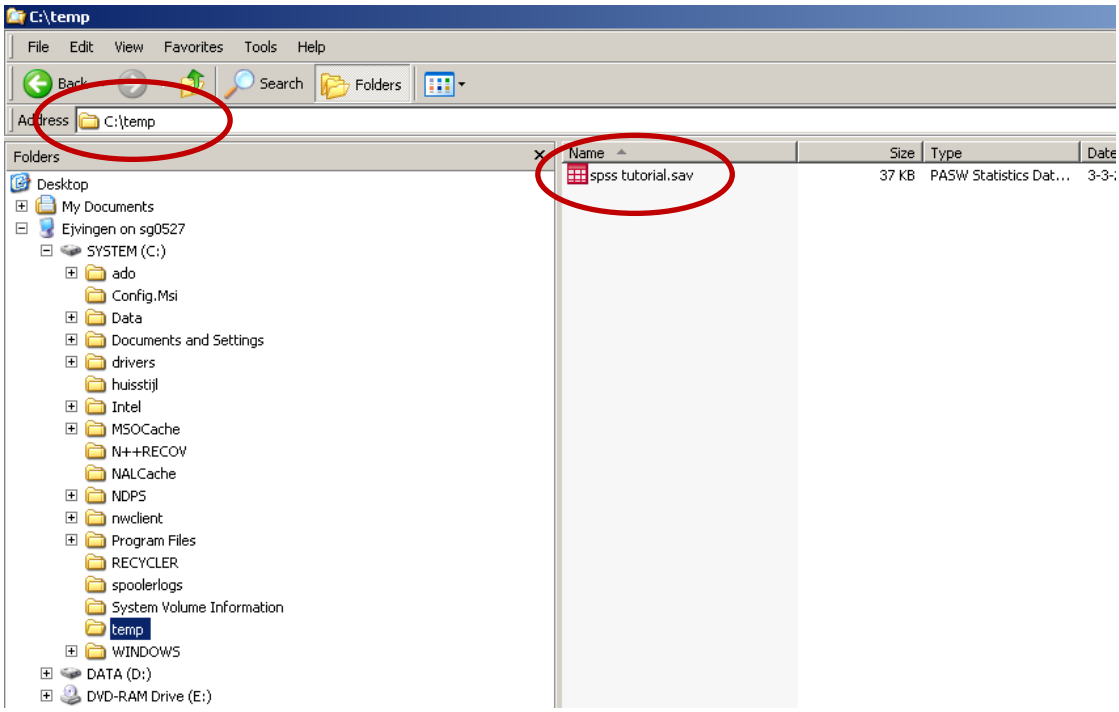


When you use SPSS syntax, comments are very important; they help you remember what you have done and why (and if you like you can briefly summarize the results of your analyses with comments). All text after the comment command will be ignored by SPSS. The easiest way to include comments is to start a line with an asterisk (*).

You can type a comment in the first line of your syntax file now. On the first line of your syntax file it is helpful to indicate what the syntax file is for. For example:

```
* This is the syntax for my first SPSS session.
```

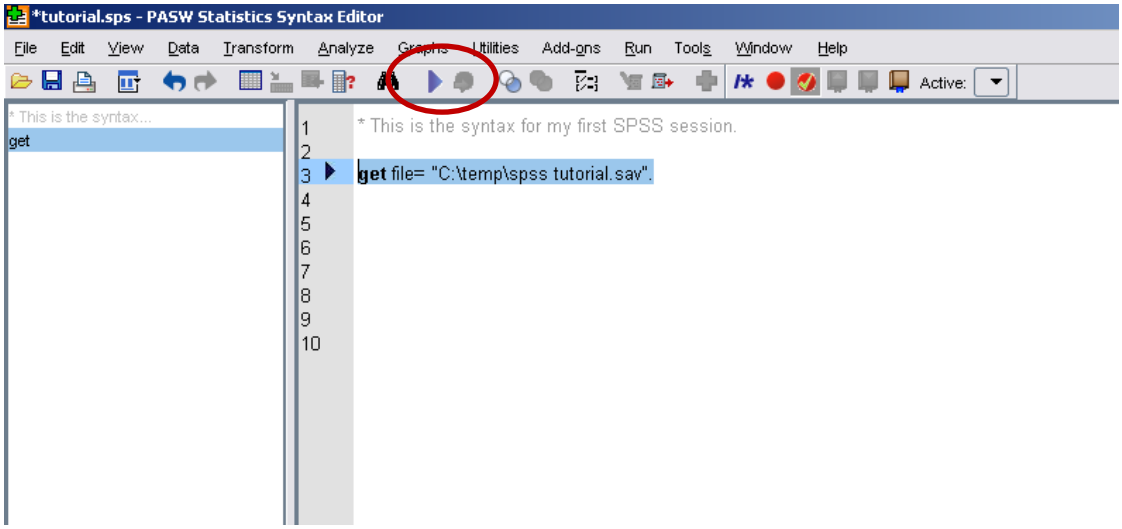
In order to use data for your analysis, you have to load them into your computer's memory. The command "get file" takes care of this. In this command you have to specify the full path of the data file. If you don't know it by heart, you can look it up in Windows Explorer (right click Windows' start button and click "explore").



The full path of the data file consists of its address + file name (see illustration). Now you're ready to use the *get file* command. You apply it by typing "get file=", the full path of your dataset within quotation marks, and a dot at the end of the line. The latter is important: you have to end every line with a dot!! (also at the end of comments, such as the one you just typed) If you don't, SPSS assumes the command continues on the next line. So, in order to load the data we will use, you type in the following:

```
get file= "C:\temp\spss tutorial.sav".
```

Obviously, you have to replace the path with your own path. The next step is to "run" the command; every line you type will only become effective after executing it. If you want to run commands from the syntax file, you select them with your mouse and press <CTRL> <R>. Or, alternatively click the "play" button (see illustration). If no syntax is selected, hitting <CTRL> <R> will make SPSS run the current line only.



After you run *get file*, a data window pops up with your data. That means you now have three windows opened: the syntax editor, the output window, and the data window. Please check the output window and make sure there are no error messages displayed. If there are, it is most likely you have misspecified the location of your data, and no data will be visible in the data window. If this is the case, you should correct the path and run the get file command again. (if it is not, please the lines you typed for missing dots)

We are now ready to start working on the data and perform analyses. But before we start, SAVE the syntax file!! And make sure you save it to a location where you can find it back (e.g., a USB stick when you are working in a computer lab). Saving your syntax regularly avoids frustration when SPSS (or your entire computer) crashes and everything is lost.

An alternative way of generating syntax – besides typing it directly – is to use the "Paste" button that is included in every command you click using the menu system. However, this typically generates a lot of syntax you don't need, making the syntax file much harder to read. Instead of pasting syntax from the menu system, using the "minimal syntax" provided in this tutorial is much more efficient! This becomes clear already after the simplest examples. For instance, we will tell SPSS to provide us with a frequency table of gender in the data. In the menu, go to "Analyze – Descriptive Statistics – Frequencies", double click the variable "Sex" or "x001"[1] and click "OK". The distribution of this variable is now displayed in the output window, with 491 (or 49%) men and 510 (or 51%) women.

We will now have a look at the equivalent of asking for these frequencies using syntax. Go to frequencies again through the menu, but now hit the "Paste" button instead of "OK". The following syntax is then displayed in your syntax window:

```
FREQUENCIES VARIABLES=x001
  /ORDER=ANALYSIS.
```

[1] This depends on whether "display names" or "display labels" is specified in the options (Edit menu). For this tutorial it is most convenient to set it to "display names".

The frequencies command is relatively simple, and that is why this syntax is still understandable. However, we can make it even simpler! As you can see in the section "The minimal syntax for the most important commands" of this tutorial, the most basic way of asking for frequencies is "freq varA", in which you replace "varA" with the name of the variable you are interested in. In this case, that would be "x001". So please type in the following syntax and run it:

```
freq x001.
```

Remember always to end a command line with a dot! In the output window you will see a table that is identical to the one you got using the menu system.

Once you have become familiar with SPSS syntax, you can abbreviate the commands you use. For example, in the last example, "freq" was used instead of the full "frequencies variables". This saves some time when you are generating a lot of syntax. However, make sure that the remaining text stays understandable, to yourself and the program (abbreviating "frequencies" to "f" would be a problem, for example, as there are more commands that start with an "f").

You now know how to ask for frequencies in SPSS syntax! Let's also try a data transformation. We will use the variable "Marital Status" or "x007". Please first explore the variable, by running frequencies:

```
freq x007.
```

As you can see, marital status has 5 categories, and three different user missing values (we will not discuss these here). If you don't see numbers in front of the categories (e.g., married should have a 1, divorced should have a 3) you need to set the output options right. In the menu, go to "Edit – Options – Output Labels" and make sure "Names and Labels" and "Values and Labels" are selected under Outline Labeling.

Suppose we are not interested in these five categories, but only in the difference between married and non-married people. That means we have to create a variable that only includes these two categories. The command "recode" can make this happen. In this command, you always specify the name of the variable, together with the old values and new values. In this case, we could assign the value of 1 to married respondents (as it is now) and 0 to all non-married respondents. The syntax to do this is the following:

```
recode x007 (1=1)(3,4,5,6=0).
```

Please inspect carefully what this command does. You first type the name of the command "recode", then the name of the variable ("x007"), and then between brackets the old and new values. In this case, the value 1 should remain 1 (category married), and values 3, 4, 5, and 6 should become 0 (category unmarried). However, it is useful to add something in front of the value changes, and that is "(missing=sysmis)". That way, you tell SPSS to keep the missing values of the original variable (and to convert them into system missing values). If you don't provide this statement it can cause errors in your data, therefore it is advisable to always specify the recode command like that. Our new syntax is now:

```
recode x007 (missing=sysmis)(1=1)(3,4,5,6=0).
```

Now run this command. It is very important to always check the results of your transformations! You can do this by running frequencies again:

```
freq x007.
```

Please inspect the results in your output window, and compare them to the frequencies table before the recode. As you can see, there are still 550 respondents in the "married" category, and there are now 447 respondents who have a value 0 (all other categories). However, there is no value label attached to the value 0. In SPSS, both variables and the values of those variables have labels to make remembering what they represent easier. You can easily see this when you go to the data window and click "variable view" (bottom of the window). Here you see the names of the variables as they are stored in the data (you always use these names in the syntax), and the label they have (what they mean). E.g. the variable "x007" has the variable label "Marital Status". Just like variables have labels to help you understand them, values can have labels too. We will now give the marital status variable new value labels, by using the "value labels" command. Please type in the following:

```
val lab x007
0 "non-married"
1 "married".
```

If you want to, you can also type everything into one line, but using a separate line for every label is a bit easier to read. Remember that SPSS considers a next line to be part of the same command if no dot is provided at the end of the line! Again the syntax starts with the name of the command ("val lab"), then the name of the variable ("x007"), and then the combinations of the values and the labels they should get (within quotation marks). Now run the syntax, and then run "freq x007" again. In the output you'll see that the value label is added in the table. It is important to always label variables and values, to make sure you still know what they represent when you use SPSS on more than one occasion.

Unfortunately, there are two problems with using the "recode" command like this: (1) when you discover that you made a mistake you cannot undo the changes, and (2) you cannot use the original variable and the recoded variable together since the former was overwritten. To solve those problems, we make sure SPSS stores the recoded values in a new variable, and the way to do that is by adding the keyword "into" and the name of a new variable:

```
recode x007 (missing=sysmis)(1=1)(3,4,5,6=0) into x007b.
```

Instead of "x007b" you can give the variable any name you like. Unfortunately, the abovementioned syntax doesn't work currently, since the original values of x007 are no longer in the computer's memory (you recoded them). However, there is an easy way to reset this! First, save your syntax file. Make sure you DON'T save the data file!! The reason why you don't save the data (something you never do when you are working with syntax) is that when you save data which has mistakes in it (e.g., due to a wrong recode) it is irreversible! Second, rerun the *get file* command at the start of

you syntax file (after selecting it). The original dataset is now restored in the computer's memory! After that you can run the last "recode…into" command.

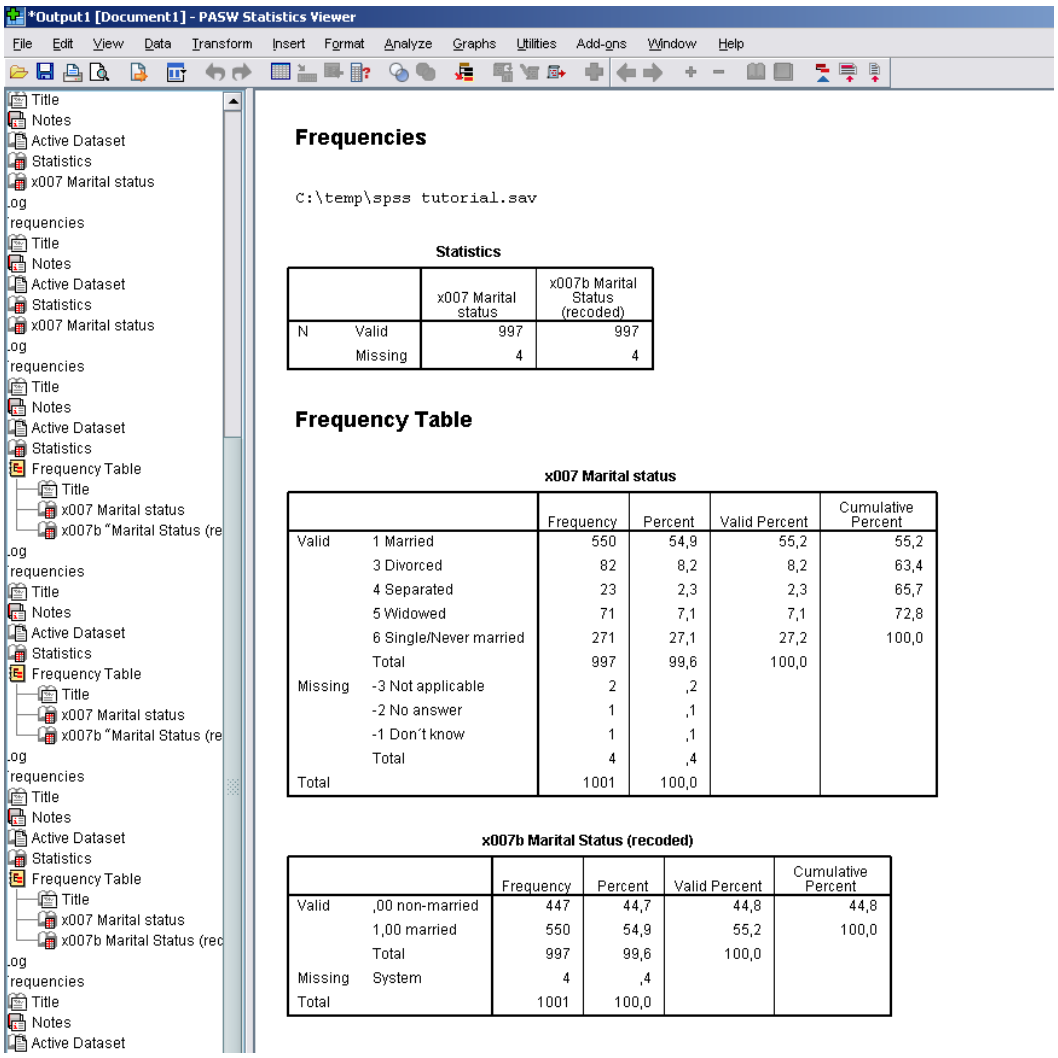
This creates a new variable, named "x007b". You can see it at the bottom of the "variable view" of your data window. It is now important to check the results of our transformations again, but before we do so, we first assign labels to the variable and its values. Similar to "value labels", the command "variable labels" gives labels to variables. The syntax is simply "var lab", followed by the name of the variable and the label (within quotation marks). So type in the following:

```
var lab x007b "Marital Status (recoded)".
val lab x007b
0 "non-married"
1 "married".
```

You can use any label you find useful. We now check the results by asking for the frequencies of both variables:

```
freq x007 x007b.
```

If everything went well, the output should look like this:



Compare the two variables:

- they should have the same number of missing values,
- the number of cases in the married category should be similar,
- the number of cases of the other categories in x007 should add up to the number of cases in the "non-married" category of x007b, and
- the new variable should have variable and value labels.

By now, you already have much of the required knowledge to start using your own syntax! We will look at two other examples of analyses to conclude.

The first one is the use of the command "means". When you are dealing with continuous variables, you often want to know what their mean values are. The syntax for that is very simple, it is just "mean" and then the name of the variable. Let's look at the variable "x003" (age):

```
mean x003.
```

As you can see in the output, the mean age is 46 years in our data (from 1001 respondents, with a standard deviation of 16 years). However – as you can see in the section about "minimal syntax"— you can do more than this with the "means" command. For example, you can specify groups by using the keyword "by".

```
mean x003 by x007.
mean x003 by x007b.
```

As you can see in the output, SPSS gives you the average ages for the different groups that are included in the two different marital status variables. Another elaboration would be to request test statistics:

```
mean x003 by x007 /stat.
```

Almost all commands in SPSS have optional subcommands, which are specified by a slash ("/") and the name of the command (in this case "/stat"). In the output you can see the ANOVA (analysis of variance) statistics; it is tested whether the differences between the means are simultaneously zero, or equivalently, whether the groups have the same average age. In our case, this is clearly not the case (judging from the p value of .000, the hypothesis is clearly rejected).

You see that by using syntax you have a very flexible way of asking for all kinds of output. Finally, we will look at the syntax that runs regression analysis. Suppose we want to analyze the effects of "marital status" and "age" on "feeling of happiness". We already created a variable for marital status ("x007b") and we can use age as it is, but we have to recode the happiness variable. As a first step, we always explore the variable:

```
freq a008.
```

As you can see the value 1 refers to "very happy" and 4 refers to "not at all happy". It would be more intuitive to let higher values correspond to increased happiness, so we

reverse the scores by recoding the variable. Additionally, we provide the new variable with labels and check the results of our transformations.

```
recode a008 (missing=sysmis)(1=4)(2=3)(3=2)(4=1) into happy.
var lab happy "happiness (recoded)".
val lab happy
4 "Very happy"
3 "Quite happy"
2 "Not very happy"
1 "Not at all happy".
freq a008 happy.
```

As you can see in the output, the values are reversed, and the number of cases in the categories and missing values are still correct. Next we do a regression:

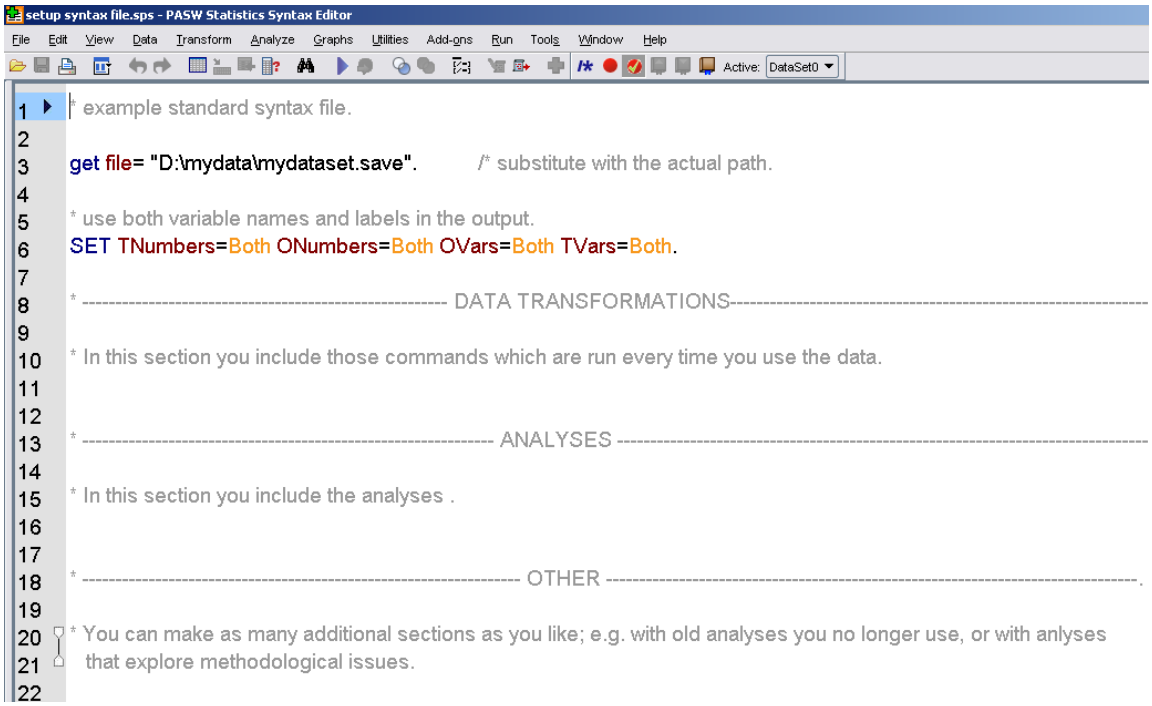
```
regres /dep=happy /enter=x007b x003.
```

After the "/dep" you always specify one dependent variable, after "/enter=" one or more independent variables. As you can see, people who are married report higher levels of happiness, and there is a negative effect of age on happiness (older people are less happy).

That's it! You completed you first SPSS session and you're ready to start working on your own syntax file. However, make sure you check the next sections of this tutorial!!

The basic setup of a syntax file

When writing syntax files, it is helpful to organize them in the same fashion. The illustration provides an example. You can use it as starting point for your own syntax, and expand it with your own text. It starts with a comment that states the purpose of the current syntax file. Subsequently, the dataset is opened using the "get file" command (see section about minimal syntax).



It is also useful to ask SPSS to provide you with both "names and labels" for variables and "values and labels" for the values of those variables in the output. You can set this in the menu, by going to "Edit – Options – Output Labels" and make sure "Names and Labels" and "Values and Labels" are selected under Outline Labeling. But easier is to specify those options at the start of your syntax file:

```
Set tnumbers=both onumbers=both ovars=both tvars=both.
```

After that, you can start with the syntax for data transformations. These are the lines of syntax you typically run at the start of every new SPSS session in an ongoing project. After that you could create a section that contains the analyses that end up in your paper (so that you can find them back easily), and optionally, additional sections can be added (e.g. for methodological issues or just exploring different model specifications). Remember that you ALWAYS SAVE YOUR SYNTAX BUT NOT YOUR DATA!!

With a separate section for data transformations, it is also very easy to reset the data in your computer's memory (e.g., when you did something wrong): you just select everything from the "get file" line until the end of the "data transformations" section and run it. Obviously, the main advantage of this strategy is that you never save

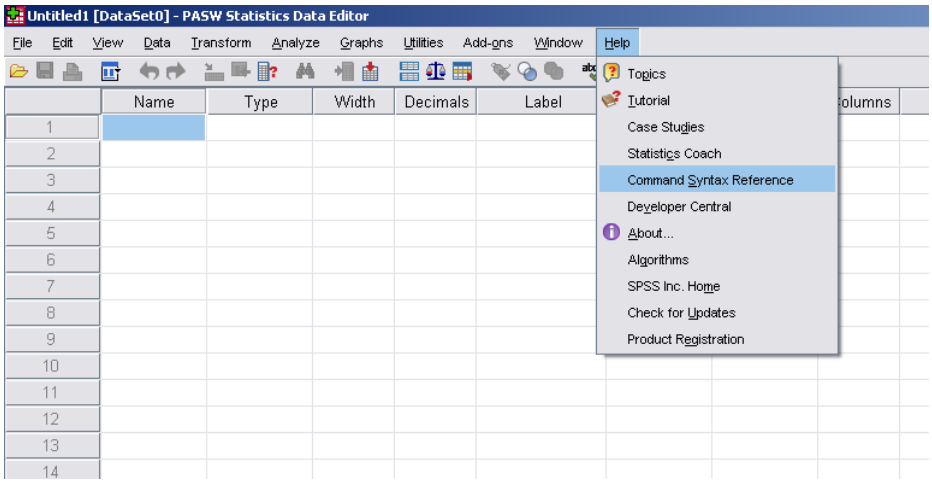
changes to the data that turn out to be wrong on later occasions! That means that whatever you did, you can correct it in your syntax and apply it to the original data. Trust me: you will use this option frequently!

Getting additional help

The preceding tutorial only covered a few basics of using SPSS syntax. In addition to that, the section on "minimal syntax" provides an overview of some very common commands in SPSS and clean ways to use them. However, at some point, you will probably need information on more advanced issues. Apart from the many handbooks that are around, you may consult:

- Google; there are many good online sources that deal with advanced issues, such as the website of UCLA (http://www.ats.ucla.edu/stat/spss/default.htm).
- Your fellow students and teacher(s)
- The command syntax reference (CSR)

The latter deserves some explanation, since most people's first impression of the CSR is that it obscures more than it clarifies. The CSR is basically a PDF file that is provided by SPSS through the help menu.



Except for examples and a brief explanation, the CSR always gives the format of a certain command, with all possible options and subcommands. At first this looks very messy, but with a few reading tips it turns out to be quite easy.

Let's look at the example of the means command (below). At first glance, this looks complicated. However you should note the following:

- Everything between square brackets […] is optional
- Everything in **bold** is used as the default option (which means you don't have to specify it)
- Everything with ** is used when an entire subcommand is omitted

Furthermore:

- If words are capitalized this means that you have to copy them literally; words in lower case letters should be replaced by your own values (typically variable names)
- The syntax itself can be either in capitals or lower case letters (SPSS syntax is not case sensitive).

```
MEANS [TABLES=]{varlist} [BY varlist] [BY...] [/varlist...]
              {ALL    }

 [/MISSING={TABLE     }]
           {INCLUDE   }
           {DEPENDENT}

 [/CELLS= [MEAN**   ] [COUNT**   ]  [STDDEV**]
          [MEDIAN] [GMEDIAN] [SEMEAN]  [SUM ]
          [MIN] [MAX] [RANGE] [VARIANCE]
          [KURT] [SEKURT] [SKEW] [SESKEW]
          [FIRST] [LAST]
          [NPCT] [SPCT] [NPCT(var)] [SPCT(var)]
          [HARMONIC] [GEOMETRIC]
          [DEFAULT]
          [ALL] [NONE]  ]

 [/STATISTICS=[ANOVA] [{LINEARITY}]  [NONE**]]
                      {ALL      }
```

Let's look at the format of the means command. First, you are allowed to leave out everything that is optional (between []). Let's do that and focus on everything not between square brackets, starting with the word "means". Since it is written in capitals, you should copy it literally in your syntax. The second thing we encounter (leaving out words within []) is {varlist}. Since this is written in lower case letters, you should specify your own variable(s) here. Alternatively, you can type the keyword "all", which is displayed underneath {varlist} to indicate that it is an alternative to entering your own variable list. By using "all", you tell SPSS to compute the means of all variables in the dataset. Usually, we do not want that; we just want a few variables instead.
A simple example of the means command would therefore be:

```
means education income.
```

Remember to end the line with a dot. This will show the means, count, and standard deviations of the variables education and income. We can also add a few options:

```
means education by gender /stat.
```

This will show the means, count, and standard deviations of education for each gender category (i.e. men and women) and tests whether they have the same mean. Or maybe we are only interested in means and we want the output to look clean. To achieve that we can tell SPSS to show means only:

```
means education /cells=mean.
```

If you follow the instructions in the command syntax reference like this, you can create fully customized output, which provides you with the information you need and which leaves out information you don't need.

The minimal syntax for the most important commands

Many SPSS commands can be used by typing very brief syntax, which is helpful for inexperienced users, as it avoids making difficult, advanced choices. The program then uses the default options for most subcommands. The list below provides an overview of common commands, in their shortest format, and with a brief explanation. The commands are organized by section and listed in alphabetical order.

Instead of some formal declaration, I have used syntax examples of the commands to explain their purpose. This should make this section easier to read for novices. Please note that the examples use the imaginary variables varA, varB, etc. Obviously, you have to replace these by your own variable names!
Note that the keyword "var" or "variables" can be part of a command. For example, in "factor var= varA varB varC", the first "var" is part of the command, and the other ones should be replaced by your own variable names.

Part I: General commands

Comments

There are three ways to include comments in your syntax.
1. By using the word 'comment' at the start of a line:

```
comment bla bla bla.
```

2. By starting a line with an asterisk (*):

```
* bla bla bla.
```

This is obviously more efficient than typing "comment".

3. By surrounding the comment with slash asterisk (/*) and a closing (*/):

```
compute a= b /* bla bla bla */ + 1.
compute a= b + 1.                  /* bla bla bla.
```

As you can see in the examples you can use this everywhere you want, even inside a command. If you use it at the end of the line you don't have to provide a closing */.

Do if … End if statements

```
do if (age < 18).
     comp varA= $sysmis.
else if (age ge 18).
     comp varA= varB.
end if.
```

Sometimes you want to perform a transformation on a subset of cases and another transformation on a second subset of cases. To make that happen you need to use a *conditional* statement, by using "do if". Every time a statement is true, the accompanying data transformations are executed and SPSS continues with the next case (other "else if…" statements are not evaluated); on the other hand, when a statement is false, SPSS continues with the next "else if…" (until it reaches "end if"). See "Selecting cases" for more conditions.

In the example, for respondents below 18 yrs, variable varA is assigned a missing value (the use of the "$" is mandatory). For respondents aged 18 or older varA is assigned the value of varB (whatever that may be).
You can also use "else" instead of "else if" to run certain syntax on all cases for which the preceding "if"-s were untrue.

Tip: If there is only one condition that needs to be evaluated, you can use the if statement in short format:

```
if (age>85) age=85.
```

This assigns age 85 to all respondents older than 85 ("top coding"). The short format of the if statement is very peculiar: you are only allowed to use it for "compute" commands, but you are NOT allowed to type "compute" after the condition.

Opening and saving data

```
get file= "name of the file".
save file= "name of the file".
```

The name of the file should contain its full path (see section "my first SPSS section").
Normally, you hardly use the save file command (since you rerun your data transformations every time you run SPSS).

Tip: if you only want to use a few variables from a large dataset, you can use the subcommand "/keep" to specify which variables you want to keep (this gives you a better overview of the data and less memory is used):

```
get file= "name of the file" /keep varA varB varC.
```

Selecting cases

```
select if (varA ge 18).
```

Selects cases (permanently) whose values on "varA" are greater / equal than 18 ("ge" is an abbreviation of greater/ equal). This can be helpful when you are doing your analyses on some group only, e.g., men, the elderly, ethnic minorities, etc. Similarly:
"eq" = equal to,
"ne" = not equal to,
"lt" = less than,
"le"= less / equal, and
"gt"= greater than.
Alternatively, you can use the <, >, and = signs to build these expressions.

Tip: If you use the command "temporary" in front of "select if" it only selects cases for one analysis or data transformation. For example:

```
temp.
select if (varA ge 18).
mean varB.
```

The means are shown for the selected cases; after that, the selection is cancelled.

Part II: Data transformations

Compute (create new variables)

```
comp varA= means (varB, varC, varD).
```

Calculates the average of variables varB, varC and varD and stores this value in the variable varA.
The command is always: `comp var = [transformation].`
The transformation can take on many forms, including any arithmetic transformation using other, continuous variables. Other examples:

```
comp varA= 10.
comp varA= 10 + varB.
comp varA= (varA/varB) + 2*varC -12.
comp varA= sqrt(varB).
```

The section "Universals - Transformation expressions" in the Command Syntax Reference (see the "Getting additional help" section) gives many other possibilities of functions and expressions you can use.

Count

```
count varA= varB, varC (2).
```

Counts the number of times that the value 2 occurs in variables varB and varC (thus, in this case that would be a number between 0 en 2) and stores this number in variable varA. Use "`freq varA.`" to see the results.

Tip: you can count any value, also "missing":

```
count tmis= varA, varB, varC (missing).
```

The number of missing values on "varA", "varB", and "varC" for each respondent are now stored in the new variable "tmis".

Recode

```
recode varA (missing=sysmis)(0=0)(1,2=1)(3,4=2) into varB.
```

Recodes the original values of varA into other values and stores them in variable varB. The use of "into varB" is recommended; that way the original variable will not be overwritten (which you may want to use / check later). The use of "(missing=sysmis)" is also recommended, especially when using the keyword "else" (see "My first SPSS session" section).

Tip: There are short formats for specifying a range of values. You can use the key word "thru" in combination with "lo" and "hi", and the keyword "else".

```
recode varA (missing=sysmis)(lo thru 8=1)(9 thru 12=2)(else=3)
into varB.
```

Variable labels

```
var lab varA "a name you make up yourself".
```

Changes the label of varA into a self-chosen label.

Value labels

```
val lab varA
0 "category A"
1 "category B".
```

Gives the values of varA (self-chosen) labels.

Part III: Analyses

Crosstabs

```
cross varA by varB.
```

Creates a crosstabulation of varA and varB. If you want percentages instead of absolute numbers, you need to specify the subcommand "/cells=". Depending on what you need you can ask for column percentages, row percentages, or both.

```
cross varA by varB /cells=count row col.
```

This gives you the absolute number of cases and row + column percentages in every cell.

Correlations

```
corr varA varB varC.
```

Displays correlations, significance, and number of cases for all pairs of variables.

Descriptives

```
descr varA varB.
```

Gives number of cases, means, standard deviations, and ranges of varA and varB.

Factoranalyse (PCA)

```
factor var= varA varB varC.
```

Without specifying options, SPSS will run a Principal Components Analysis (PCA), with varimax rotation, using the criterion of eigenvalue > 1 to determine the number of factors.

The command syntax reference (see section "Getting additional help") provides you with many other options for extraction or rotation.

Frequencies

```
freq varA varB.
```

Gives you frequencies (absolute numbers + percentages) and missing values of varA and varB.

Means

```
mean varA  varB.
```

Provides the means, standard deviations, and number of cases for varA and varB.
When you want to assign the mean value to a variable, you have to use the compute command.

Tip: you can get an overview of the means for different groups by using the "by" keyword.

```
mean varA  varB by varC.
```

Regression (linear)

```
regres /dep varA /enter= varB varC.
```

"varA" is your (continuous) dependent variable. After "/enter=" you can add as many independent variables as you like.

Tip 1: typing "/enter=" more than once makes SPSS run separate models:

```
regres /dep varA /enter= varB /enter= varC
```

This gives you two models: one with "varB" as independent only, and one with both "varB" and "varC". This is very useful when you are interested in how entering an additional variable affects the estimates of the other variables in the model (e.g. in intermediation analysis).

Tip 2: by adding the subcommand "select", the analysis is only performed on the group that meets that criterion.

```
regres select varD eq 1 /dep varA /enter= varB varC.
```

In this example, only cases that have a score of "1" on "varD" (e.g., women) are included in the analysis. Note: SPSS is very picky with regard to this option. E.g., you are not allowed to use brackets () and you should use one of the following expressions: `EQ`, `NE`, `LT`, `LE`, `GT`, of `GE`. (see the "selecting cases" command). Furthermore, you have to specify select before "/dep".

Reliability (Cronbach's alpha)

```
rel var= varA varB varC.
```

Provides the reliability of varA, varB, and varC as a scale.

Tip: add "/sum=all" and "/stat=all" to acquire all available information, such as the average inter-item correlation and the change in Cronbach's alpha when one variable is omitted.

T-test

```
ttest groups= varA(1,4) /var=varB
```

There are several ways to test whether differences in group means are significant (or formally: whether they are the same), and this is one of them (alternative commands are "means", "oneway", or "regression" (with a dummy variable for the contrast)).
In the example, the groups that need to be compared are specified first; in this case, respondents with value 1 on varA are compared with respondents with value 4 on varA. They are compared with regard to their average on "varB".

Part V: Other helpful syntax tips

Controlling missing values

When you compute sumscores, averages etc you can control for which cases the computation is still valid (and which will be assigned a missing value). This can be done by adding ".x" to "mean", "sum", etc in computations. The x then indicates the minimal number of variables you want to have a valid score.

```
comp varA= means.2 (varB, varC, varD).
```

This calculates the mean of variables "varB", "varC", and "varD", and assigns it to variable "varA" only if 2 (out of the 3) scores are valid (non-missing). Respondents with 1 or 0 valid scores will be assigned a missing value.

Dummy variables

The most efficient way to compute dummy variables is by using the compute command combined with a conditional statement.

```
comp varAd1 = (varA = 1).
comp varAd2 = (varA = 2).
```

Variable "varAd1" will be assigned the value 1 if the condition between brackets is true. In fact, this is an "if" statement in disguise. SPSS will automatically assign zeros and missing values to the other cases. The brackets are optional.
A more complicated example:

```
comp aged1= ((age ge 20) and (age le 23)).
```

Creates a dummy variable (aged1) that is true (1) for all respondents aged 20, 21, 22, and 23. Respondents with other ages are assigned a "0" on aged1, and those with missing values on age are assigned a missing value on aged1 as well.

Variable lists

Instead of typing all the names of the variables you want to use in a command, you can use the keyword “to”: e.g. `descr varA to varZ`.
Note that you SPSS now gives you all variables between varA and varZ in the data.

Excercises (using the tutorial data)

1. What is the syntax to compute the year of birth of respondents, using the variable x003 (age) and knowing that the survey was conducted in 2000?

2. Run the syntax of exercise 1. Next, create and run the syntax that displays the correlation between your newly created variable and x002. What is it?

3. The correlations command has a subcommand that you can use to get one-tailed instead of the more common two-tailed test of significance. Find that subcommand in the Command Syntax Reference and create the syntax to calculate the correlation between age and "feeling of happiness", with a one-tailed test of significance.

4. Write the syntax that computes a new variable that consists of the squared age of respondents.

5. Some respondents have not answered all of the questions. Write the syntax that counts the number of missing values across all variables. You can use the keyword "to" to indicate a variable list (e.g. "var1 to var99").

6. View the scores on the variable you just created. How many people have a score of 3? What does that mean?

Useful literature

Aiken, L. S., S. G. West, et al. (1991). Multiple regression: testing and interpreting interactions. Newbury Park, Calif. etc., Sage Publications.

Allison, P. D. (1999). Multiple regression: a primer. Thousand Oaks, CA [etc.], Pine Forge Press.

Pallant, J. (2001). SPSS survival manual: a step by step guide to data analysis using SPSS for Windows. Buckingham [etc.], Open University Press.

Tabachnick, B. G. and L. S. Fidell (2000). Using multivariate statistics. Boston [etc.], Allyn and Bacon.

Te Grotenhuis, M., & Visscher, C. (2007). SPSS met syntax. Assen: Van Gorcum.