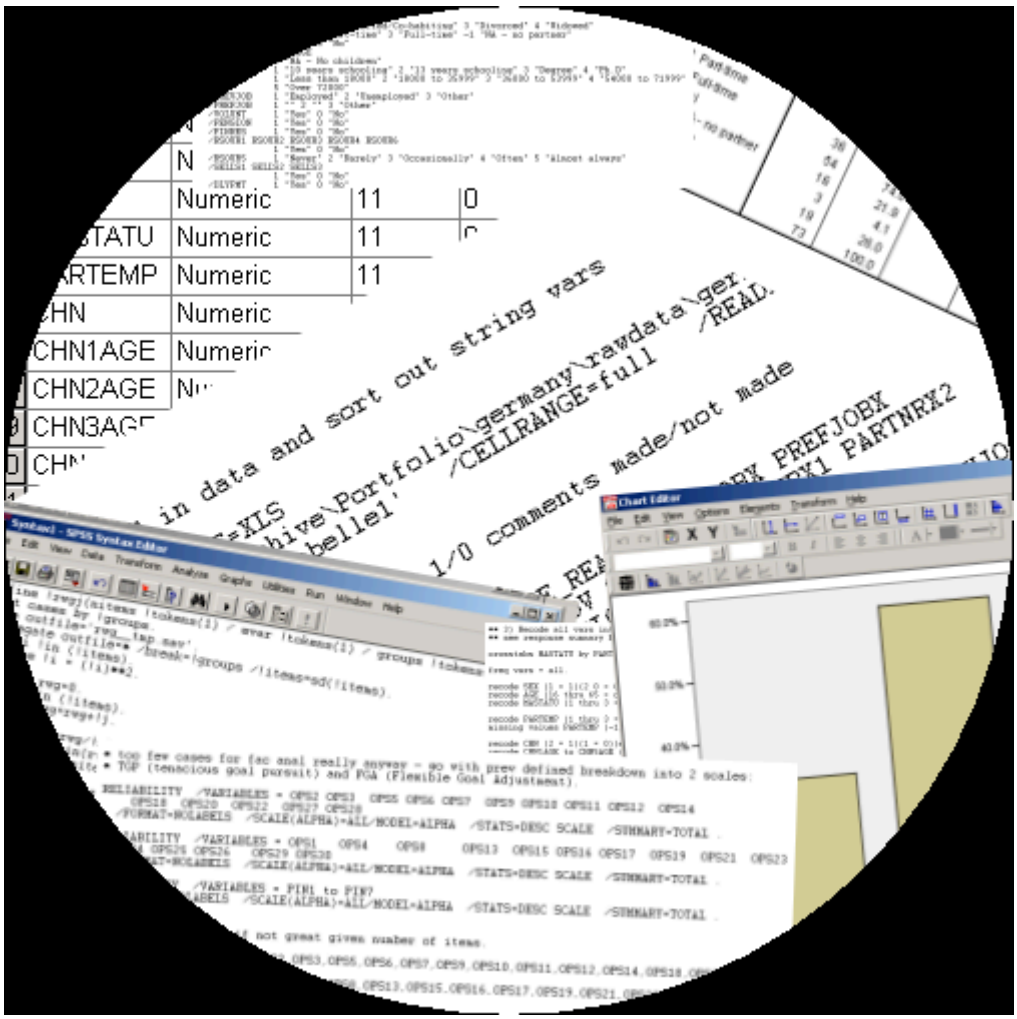
ASSESS: SPSS USERS GROUP

Half-Day SPSS Workshop

York, October 2015



Writing and using Macros

http://www.spssusers.co.uk/

Contents

Timetable (Macros)

1.45pm REGISTRATION

2.00pm Macro sessions 1, 2, 3 & 4

3.30pm COFFEE

3.50am Macro sessions 5 & 6

4:45pm (approx.) FINISH

Note all files for today's workshop are contained in the macroscourse folder on the memory stick which should be on drive E:. To access the workshop folder, macroscourse, insert the memory stick into the PC. The macroscourse folder should appear on the screen. (If not, click My Computer in the 'My Computer' window and then click removeable disk E). The macroscourse folder contains two folders for the macros and OMS halves of this workshop. We will begin in the macros folder.

Writing and Using SPSS Macros

Session 1: Introduction to Macros

1.1) What is a Macro?

A macro is a set of commands that generates SPSS command syntax. It can be used to reduce the effort required for complex tasks and is useful shorthand for automatically carrying out repeatable operations. It builds blocks of SPSS syntax and controls execution of these blocks.

It enjoys all the flexibility of syntax presenting an audit trail of analyses, repeatability as well as succinctness over the usual syntax.

- Issue a series of commands repeatedly
- Group sets of variables
- Create complex input programs which can be read into SPSS and executed.

Session 2: Creating a macro

2.1) Writing a basic macro

The structure of a macro is as follows:

```
DEFINE macroname (optional arguments)
    macro body commands
!ENDDEFINE.
```

Each macro must start with a `DEFINE` command and end with a `!ENDDEFINE` command which indicates the beginning and end of the macro.

The macroname following the `DEFINE` command can be up to eight characters long and usually starts with a letter or '!'. The macro body contains SPSS syntax commands or special macro commands. The optional macro input elements (arguments) are contained in brackets following the macroname. These refer to words or numbers which are fed into the SPSS commands by the macro e.g. variable names, sample size.

The subsequent examples in this session use demographic data in file *demog.sav* in the macros folder.

2.2) Using a macro

A macro is written and run in a SPSS syntax window. It is run (ie implemented) as for all other SPSS syntax using either the run button or run menu. There are two parts to run: The macro, itself, which defines a new command into SPSS and the newly macro defined command itself, identified by the macro's name. The macro, identified by its name, can be thought of as an extra SPSS syntax command, created by the user.

Session 3: Positional arguments

3.1) A macro with no input elements (arguments)

(see *eg 1.sps* in macros folder in the macroscourse folder which uses data in *demog.sav* located in the macros folder)

```
DEFINE !NEWVARS ( )
 SEX MASTAT
!ENDDEFINE.

MEANS AGE BY !NEWVARS.
```

The macro `!NEWVARS` is shorthand for variables 'sex' and 'mastat'. This is equivalent to

```
MEANS AGE BY SEX MASTAT.
```

The parentheses after the macro name are compulsory even if there are no input elements between them.

3.2) A macro with two positional input elements

We can send variable names and other details needed by the syntax commands inside the macro using input statements. The input statement is the bracketed list of input elements following the macroname on the `DEFINE` command. Input elements are usually entered into the macro in groups occupying the same position(s) in the macro. For example, the variables 'sex' and 'edqual' may both be used to define subsamples of age means and consequently both follow the 'by' in the `MEANS` command.

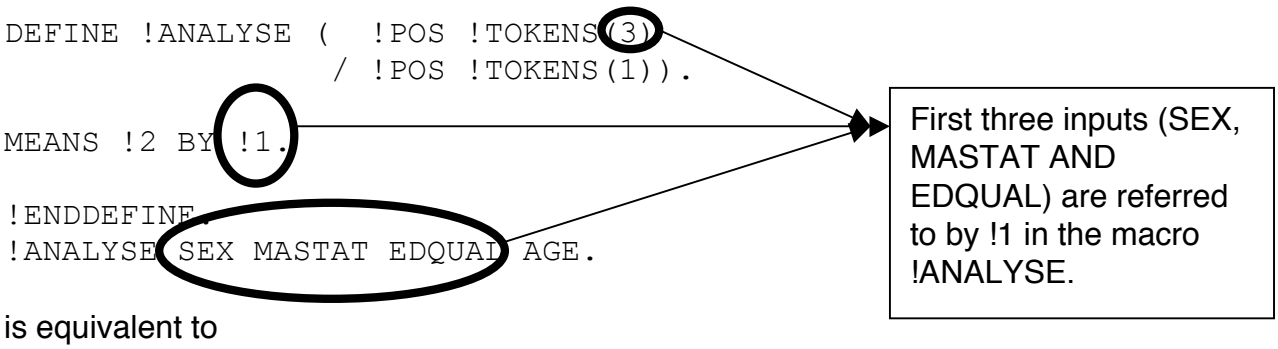
```
MEANS AGE BY SEX.
MEANS AGE BY EDQUAL.
```

could be expressed more succinctly as

```
MEANS AGE BY B. /* B={SEX, EDQUAL}
```

Each set of input elements is referred to in the macro input statement using `!POS !TOKENS(n)` which stands for positional tokens where n is a whole number greater than 0. n denotes the number of input elements (or tokens as SPSS calls them) which occur in the same position(s) in the macro. For example `!TOKENS(3)` means three input elements occur in the same position.

The !POS command states each set of input elements will be referred to in the macro by a '!' followed by a number representing their position, or order in which they follow the macro name when the macro is run . For example (see *eg 2.sps*):



is equivalent to

```
MEANS AGE BY SEX MASTAT EDQUAL.
```

The input statement, in brackets, which forms part of the DEFINE command:

```
(   !POS !TOKENS(3)     /*  first set of 3 input elements = SEX MASTAT EDQUAL
 / !POS !TOKENS(1)).  /*   second set of 1 input element = AGE
```

tells the macro there are two sets of input elements consisting of one group of three elements and another group of one element. The input elements are referred to in the macro by their position in the sequence following the newly defined macro command !ANALYSE with a '!' in front ie as '!1' and '!2'. Each set of input elements in the input statement is placed on a separate line. Except for the first, each set of input elements is separated by a '/' in the input statement. Notice the full stop at the end of the input statement.

When we call the macro there will be four input elements (which in this case are variable names) following the macroname, divided into two groups of sizes three and one respectively. The first three elements name the three variables which define subsamples of the data: 'sex', 'mastat' and 'edqual'. These are called '!1' in the macro because the first row of the input statement assigns the first three variables following the !ANALYSE command in position 1. The last variable name ('age') represents the variable which will be averaged in each subsample. This is referred to as '!2' in the macro because the second row in the input statement assigns the next (fourth) variable, following the three variables in position 1, into position 2. Notice the input element sets can appear in any order in the macro e.g. '!2' can be referred to in the macro before '!1'.

3.3) Exercises

1. Create a macro called !ctabs to produce a simple table using the Crosstabs procedure, with counts and row percentages. Use one row variable and one column variable as positional arguments.

Run the macro with the row variable= mastat and the column variable=sex. Your macro should generate the following SPSS commands:

```
CROSSTABS TABLES=MASTAT BY SEX
/ CELLS = COUNT ROW.
```

2. Modify the !ctabs macro to let you specify 2 row variables but only one column variable. Run the macro with
   row variables = mastat, edqual and the
   column variable = sex.

Your macro should generate the following commands:

```
CROSSTABS TABLES= MASTAT, EDQUAL BY SEX
/ CELLS = COUNT ROW.
```

3. Check the effect of `SET MEXPAND=OFF` and `SET MEXPAND=ON` by running the !ctabs macro after running each of these commands. Repeat using `SET PRINTBACK=ON SET MPRINT=ON` & `SET PRINTBACK=OFF SET MPRINT=OFF.` Now re-run the !ctabs macro definition and run it using firstly `SET ERRORS=OFF` and then `SET ERRORS=ON`. Finally run the `SHOW` command.

4. What happens if you re-run your macro (!ctabs) with
   row variables = mastat, edqual, bmicat
   and column variable=sex.

Session 4: Keyword arguments, `!CMDEND` and `!ENCLOSE`

4.1) Keyword input

Instead of the order of input in the input statement we can refer to our groups of input elements by giving them a name (referred to in SPSS as a keyword).

We can, for example, rewrite the example in 3.2 as

```
DEFINE !ANALYSE (   GROUP !TOKENS(3)
                  / Y     !TOKENS(1)).

MEANS !Y BY !GROUP.

!ENDDEFINE.

!ANALYSE Y=AGE GROUP=SEX MASTAT EDQUAL.
```

This macro is in *eg 3.sps.*

There are two groups of input elements called ‘group’ and ‘y’ referred to as ‘!GROUP’ and ‘!Y’ respectively inside the macro. Names have replaced the positional input item identifier, `!POS`, in the input statement. ‘!Y’ replaces ‘!2’ inside the macro referring to the input element: age and ‘!group’ replaces ‘!1’ in referring to the set of three input elements: ‘sex’,’mastat’ and ‘edqual’. ‘Y’ and ‘!Y’ represent the variable which is averaged and ‘GROUP’ and ‘!GROUP’ the set of three categorical variables which define the subsamples over which age is averaged.

4.2) The `!CMDEND` command

Instead of specifying the number of input elements, n, in an input element group using the `!TOKENS(N)` command we can use the more flexible `!CMDEND` command. `!CMDEND` defines the input elements as all elements in the macro call (eg `!ANALYSE SEX MASTAT EDQUAL`

AGE) not previously allocated to any input element groups in the input statement. In the first example below !CMDEND refers to the variable name 'age' whereas in the second example it refers to the variable names defined in the argument list by the label 'group', namely, 'sex', 'mastat' and 'edqual'.

With positional inputs (see *eg 4.sps*):

```
DEFINE !ANALYSE (   !POS !TOKENS(3) /* three input elements called !1
                  / !POS !CMDEND). /* what's left (called !2)

MEANS !2 BY !1.

!ENDDEFINE.

!ANALYSE SEX MASTAT EDQUAL AGE.
```

With keyword inputs:

```
 DEFINE !ANALYSE (   Y !TOKENS(1) /* one input element called 'Y'
                   / GROUP !CMDEND). /* what's left called 'GROUP'

MEANS !Y BY !GROUP.

!ENDDEFINE.

!ANALYSE Y=AGE GROUP=SEX MASTAT EDQUAL.
```

4.3) The !ENCLOSE command

A more flexible alternative to !CMDEND for defining an unspecified number of input elements is !ENCLOSE. Unlike !CMDEND more than one !ENCLOSE may be used in defining input elements into a macro. The elements are defined by sandwiching them between a pair of stated characters e.g. ' " ' or '/'. In the example below (see *eg 4.sps*) !ENCLOSE specifies 'age' as the second input element referred to as '!2' in the macro.

```
DEFINE !ANALYSE (   !POS !TOKENS(3) /* three input elements called !1
                  / !POS !ENCLOSE('/','/')). /* 1 input element
between '/'s (called !2)

MEANS !2 BY !1.

!ENDDEFINE.

!ANALYSE SEX MASTAT EDQUAL /AGE/.
```

And in the following example !ENCLOSE specifies the first set of three input elements referred to as '!1' in the macro.

```
DEFINE !ANALYSE (!POS !ENCLOSE('/','/') /* 3 input elements in '/'s
named !1
```

```
                  / !POS !TOKENS(1)). /*  1 input element called !2

MEANS !2 BY !1.

!ENDDEFINE.

!ANALYSE / SEX MASTAT EDQUAL / AGE.
```

We can specify two `!ENCLOSE` together: note the extra '/'s required in the macro call.

```
GET FILE='E:\MACROSCOURSE\MACROS\DEMOG.SAV'.
EXE.

DEFINE !ANALYSE (   !POS !ENCLOSE('/','/') /* 3 input elements called
!1
                  / !POS !ENCLOSE('/','/')). /* 1 input element
between '/'s (called !2)

MEANS !2 BY !1.

!ENDDEFINE.

!ANALYSE / SEX MASTAT EDQUAL / / AGE/.
```

4.4) Exercises

1. Adapt the !ctabs macro you wrote in question 2 of exercises 3.3 so that the command to run the macro is `!CTABS ROW=MASTAT EDQUAL COL=SEX` and run it.

2. Rewrite the crosstabs macro you made in question 2 of exercises 3.3 but using the !cmdend token for the rows input.

3. Rewrite the crosstabs macro you made in question 2 of exercises 3.3 but using the !enclose token to sandwich the column input between two forward slashes so the macro is run by typing `!CTABS ROW=MASTAT EDQUAL COL= /SEX/.`

Session 5: Incorporating other macros

5.1) The `INCLUDE` command

The `INCLUDE` command is a convenient way of reading in SPSS syntax contained in a syntax file. It is the macro equivalent of the `GET` command for SPSS data files. The SPSS syntax must be contained in a macro.

```
INCLUDE FILE='SPSS syntax filename'.
```

The file *analyse.sps* in the macros folder contains a macro called `!ANALYSE` which contains syntax to obtain means for males and females:

```
MEANS AGE HEIGHT WEIGHT BY SEX.
```

We can incorporate this syntax into a SPSS session without writing or running the SPSS command syntax contained in the body of the macro called `!ANALYSE`. Once the syntax file containing macro `!ANALYSE` is read into the SPSS session using the `INCLUDE` command `!ANALYSE` becomes a recognized SPSS syntax command. Running '`!ANALYSE.`' produces age, height and weight means for males and females. So the user supplied SPSS macro `!ANALYSE` can be run in a SPSS session as if it were any other SPSS (in-built) command. The below syntax allows the macro `!ANALYSE,` contained in the file *analyse.sps* in the syntax files folder, to be used in a SPSS session (see *eg 5.sps*):

```
INCLUDE FILE='E:\macroscourse\macros\analyse.sps'.
!ANALYSE.
```

More than one macro may be contained in a single syntax file. All macros in the syntax file are read into SPSS if the syntax file they are contained in is used in an `INSERT` command.

5.2) String macros

These are available for use in macros and manipulate string variables (ie characters).

`!CONCAT (`*`string1, string2`*`)`
Concatenate (ie join) the strings called *`string1 & string2.`*

`!LENGTH (`*`string`*`)`
Return the length of the specified string (the number of characters it contains). Note that the result is itself a string.

`!SUBSTRING (`*`string, from,[length]`*`)`
Return a substring of the specified string, starting with the *from*th character and consisting of *length* characters.

`!INDEX (`*`string1, string2`*`)`
Return the character position of the first occurrence of string2 in string1. The result is a string representation of a number.

`!HEAD (`*`string`*`)`
Return the first token within *string.*

`!TAIL (`*`string`*`)`
Return the last token within *string.*

`!QUOTE (`*`string`*`)`
Put apostrophes around *string.*

`!UNQUOTE (`*`string`*`)`
Remove any quotes and apostrophes from around the argument (i.e. at the beginning and end of *string).*

`!UPCASE` (*string*)

Convert *string* to uppercase.

`!BLANKS (n)`

Return a string containing just *n* blanks.

`!NULL`

Return an empty string (i.e. one of length 0).

`!EVAL (`*`string`*`)`

Scan *string* for macro calls. If it contains any macro calls, these are expanded in returning a result. Otherwise the result is *string* unchanged.

For example `!CONCAT(AQ,1)` in the below creates a variable called AQ1 and can be used with SPSS commands such as `FREQUENCIES.` Try running the below on file *aqdat.sav*.

```
DEFINE !FREQ ( ).
FREQUENCIES !CONCAT(AQ,1).
!ENDDEFINE.

!FREQ.
EXE.
```

5.3) Nested macros

Another useful feature of macros is that they can be nested within each other. This means that one macro can be created and called from within another macro. For example (*eg 6.sps*) the macro *!analy1* is referenced by macro *!analy2* to additionally produce frequencies.

```
* NESTED MACROS

DEFINE !ANALY1 ( !POS !CMDEND ).
FREQUENCIES VARIABLES= !1.
!ENDDEFINE.

DEFINE !ANALY2 (!POS !CMDEND ).

!ANALY1 !1. /*  macro !ANALY! is run within macro !ANALY2.

DESCRIPTIVES !1.
!ENDDEFINE.

GET FILE='E:\MACROSCOURSE\MACROS\DEMOG.SAV'.

!ANALY2 MASTAT.
!ANALY1 SEX.
```

5.4) Exercises

1. Save the crosstabs macro you made in question 2 of exercises 3.3 (without the call) into a file called *mymac.sps* in the *macroscourse/macros* folder. Then close it *(File>Close).*
Now use `INCLUDE` and call the macro defined in *mymac.sps***.**

2. Write a macro called !freqs which produces frequencies of variables ‘sex’ and ‘mastat’. The variables names being defined in a macro called !newvars. The !newvars macro should be referred to (ie nested) within the macro called !freqs.

Session 6: More advanced statements in macros

6.1) !DO - !DOEND commands

Looping constructs are available for use in macros and use the !DO and !DOEND commands.

```
!DO !varname=start !TO finish [BY step]

   statements  [!BREAK]

!DOEND
```

In the above start, finish and step are all numbers and the !DO iterates through from start to finish in steps of step.

or, alternatively to repeat an action on a list of variable names.

```
!DO !varname !IN (list)

   statements  [!BREAK]

!DOEND
```

To illustrate !DO and !DOEND open file *aqdat.sav.* Suppose we wish to obtain frequencies on 50 variables, called aq1 to aq50. These are contained in file *aqdat.sav* in the macroscourse/macros folder. They represent scores on 50 items assessing different personality traits in 20 patients.

The frequencies could be obtained with one line of syntax using a short macro provided the variables are in consecutive columns in the spreadsheet.

Using conventional (in-built) SPSS syntax with the FREQUENCIES command we could run the below:

```
FREQUENCIES
  VARIABLES=AQ1 AQ3 AQ5 AQ7 AQ9 AQ11 AQ13 AQ15 AQ17 AQ19 AQ21 AQ23
AQ25 AQ27 AQ29 AQ31 AQ33 AQ35 AQ37 AQ39 AQ41 AQ43 AQ45 AQ47 AQ49.
```

The above can be achieved more succinctly using the macro below which only involves typing out the string ‘aq’ once (see eg *7.sps*)!

```
DEFINE !FREQ ( START !TOKENS(1)
             / END !TOKENS(1)
             / STEP !TOKENS(1)) .
!DO !I=!START !TO !END !BY !STEP.
FREQUENCIES !CONCAT(AQ,!I).
!DOEND.
!ENDDEFINE.

!FREQ START=1 END=50 STEP=2.
EXE.
```

!FREQ START=1 END=50 STEP=1. (note the full stop) is equivalent to FREQUENCIES VARIABLES=AQ1 to AQ50.

6.2) Dummy variable example

Dummy variables are used for fitting group variables in regression analyses. They take the value '1' for a specific group and '0' otherwise. You can create four dummy variables, one for each category, for the 'bmicat' variable in *demog.sav* using the logical variable command (in yesterdays syntax workshop) below (see *eg 8.sps*):

```
COMPUTE BMICAT_1 = (BMICAT EQ 1).
COMPUTE BMICAT_2 = (BMICAT EQ 2).
COMPUTE BMICAT_3 = (BMICAT EQ 3).
COMPUTE BMICAT_4 = (BMICAT EQ 4).
EXE.
```

This creates four dummy variables which take the value '1' for groups 1,2,3 and 4 respectively and '0' otherwise.

Exercises

Write a macro called `!DUMMY` which uses the looping `!DO !DOEND` construction to act as a variable counter (from 1 to 4) to run the above logical variable syntax to create 4 dummy variables from the group variable, 'bmicat' in file *demog.sav*. The macro should be called using `!DUMMY BMICAT 4.` The input elements here are the variable name and the number of groups. The macro makes use of the string macro `!CONCAT`. `!CONCAT` adds the string 'bmicat' to the character string '_I' to form a dummy variable for the i-th group. So you end up with 4 new dummy variables: bmicat_1 to bmicat_4. You can adapt the example in 6.1.

Create a macro called `!INMISS` which uses the `!DO !DOEND` construction with the `SMEAN` function in the data file *aqdat.sav* to impute the missing values in all the questions with the variable mean.

6.3) The `!IF` command

```
!IF (expression) !THEN statements

    [!ELSE statements]

!IFEND
```

The macro (*eg 9.sps*) below performs a two-way frequency table if variable 'type' is categorical otherwise it performs an unpaired t-test. It uses the string macro, `!UPCASE,` which ensures a categorical variable can have type 'cat' or 'CAT'. The data is in file *demog.sav*.

```
DEFINE !GPCOMP(TYPE !TOKENS(1)
               /VAR !TOKENS(1)
              /GROUP !TOKENS(1)).
!IF (!UPCASE(!type) !EQ 'CAT') !THEN
CROSSTABS
  /TABLES=!VAR  BY !GROUP
  /FORMAT= AVALUE TABLES
```

```
  /STATISTICS=CHISQUARE
  /CELLS= COUNT .
!ELSE
T-TEST
 GROUPS=!GROUP(0 1)
 /MISSING=ANALYSIS
 /VARIABLES=!VAR
 /CRITERIA=CIN(.95) .
!IFEND
!ENDDEFINE.

!GPCOMP TYPE=CAT VAR=BMICAT GROUP=SEX.
!GPCOMP TYPE=NOTCAT VAR=AGE GROUP=SEX.
```

6.4) Macros on the web

http://www.spssusers.co.uk/Tips/macrolibs.html
http://www.spsstools.net/Macros.htm
http://www.ats.ucla.edu/stat/spss/code/macro_prgramming.htm
http://imaging.mrc-cbu.cam.ac.uk/statswiki/FAQ/ (Can copy and paste into syntax window!)
http://www.afhayes.com/spss-sas-and-mplus-macros-and-code.html
http://www2.jura.uni-hamburg.de/instkrim/kriminologie/Mitarbeiter/Enzmann/Software/Enzmann_Software.html
http://mason.gmu.edu/~dwilsonb/ma.html (meta-analyses)
http://www.statisticshell.com/meta_analysis (meta-analyses)
http://www.columbia.edu/~ld208/ (multivariate normality)
ftp://ftp.boulder.ibm.com/software/analytics/spss/support/Stats/Docs/Statistics/Macros/RMPost.htm (Posthoc tests for ANOVA contrast)
http://www.socialsciences.leiden.edu/educationandchildstudies/childandfamilystudies/organisation/staffcfs/van-ginkel.html (multiple imputation for ANOVAs)
http://tqmp.org/Content/vol10-1/p029/p029.pdf (Bruce Weaver macro for 95% CIs for a Pearson correlation)
http://socialsciences.leiden.edu/educationandchildstudies/childandfamilystudies/organisation/staffcfs/van-ginkel.html (multiple imputation)

http://www.stat.tamu.edu/spss.php (Michael Speed's macros but this page no longer exists! If you see a macro you like on the web save it to your hard drive!) Included a box-cox macro which I have a copy of for sharing!

Some of the examples at the above websites are complex and use the MATRIX syntax in SPSS. There are also two macros (Canonical Correlations.sps and Ridge regression.sps) in C:\Program Files\SPSS.

Some useful references:

Aberson, CL (2010) Applied power analysis for the behavioural sciences. Routledge Academic:London.

Boslaugh, S. (2005) An Intermediate Guide to SPSS Programming: Using Syntax for Data Management. Sage:Thousand Oaks, CA. SPSS syntax (including macros) with some description of the statistical methods that they implement.