SPSS Tutorial 7: Various Topics

Goals

- To compute AGE from date of birth and another date variable
- To merge files (add variables, and add cases)
- To flag the First and Last records in a group of records
- To combine data across records using the AGGREGATE command
- To restructure a data file (from WIDE to LONG and vice versa)

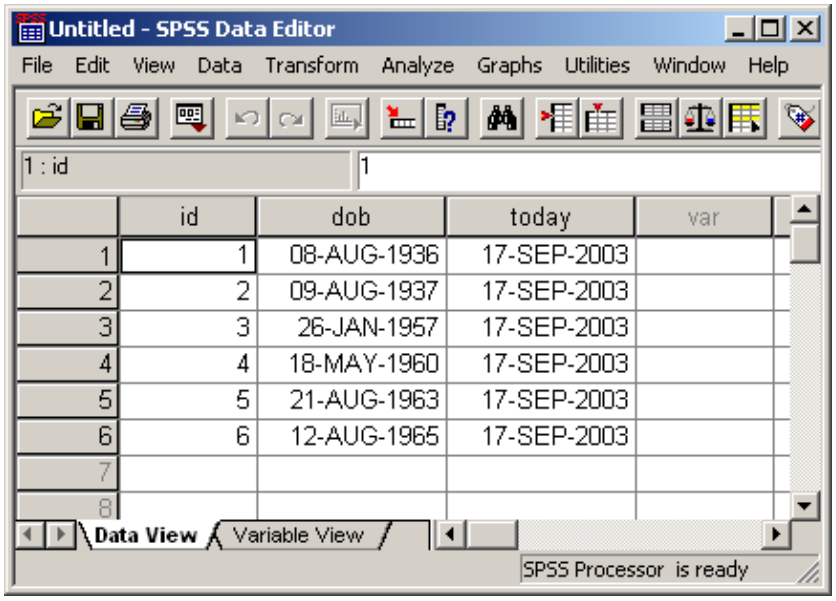
Computing AGE

After SPSS is running, open a new syntax file (File→New→Syntax), paste the following syntax into it, and run the syntax. (If you wish, add another line of data that has your own birthday.)

```
DATA LIST LIST /id (f5.0) dob(date11).
BEGIN DATA.
1 8-Aug-1936
2 9-Aug-1937
3 26-Jan-1957
4 18-May-1960
5 21-Aug-1963
6 12-Aug-1965
END DATA.

numeric today (date11). /* This creates a numeric variable TODAY.
compute today = $time.  /* $TIME is a system variable.
exe.
var width dob today(11).  /* Widen column for display purposes .
```

Your data file should now look like this (except TODAY will have a different date):



Here are the steps needed to compute a new variable AGE:

- Click on Transform→Compute (when the data editor is active)
- Type **age** into the Target Variable box
- Find the CTIME.DAYS(timevalue) function in the list of functions, and move it into the Numeric Expression box
- Replace the question mark in `CTIME.DAYS(?)` with `today - dob`
- Divide the result of the CTIME.DAYS function by 365.25 to convert the scale to YEARS; i.e., add `/365.25` to the end of the numeric expression
- PASTE and run the syntax, then look at your data file

The syntax you generated should look something like this:

```
compute age = ctime.days(today-dob)/365.25.
exe.
```

And the data file should look like this:

ID	DOB	TODAY	AGE
1	08-AUG-1936	17-SEP-2003	67.11
2	09-AUG-1937	17-SEP-2003	66.11
3	26-JAN-1957	17-SEP-2003	46.64
4	18-MAY-1960	17-SEP-2003	43.33
5	21-AUG-1963	17-SEP-2003	40.08
6	12-AUG-1965	17-SEP-2003	38.10

As you can see, it is fairly easy to compute AGE in SPSS, once you know how. Unfortunately, one of the SPSS manuals gives a very poor example of how to compute AGE. It uses the YRMODA and XDATE functions, as follows:

```
COMPUTE age2 = ( YRMODA(XDATE.YEAR(today),XDATE.MONTH(today)
 ,XDATE.MDAY(today))-YRMODA(XDATE.YEAR(dob),XDATE.MONTH(dob)
 ,XDATE.MDAY(dob))) / 365.25 .
EXECUTE .
```

If you copy and run this syntax, you will see that it gives essentially the same results as the CTIME.DAYS method (the values of AGE and AGE2 are identical to 2 decimals in most cases). But it is obviously a far more cumbersome method.

How to Merge Files: Adding New Variables

Databases are often divided into several sections. For example, imagine a study involving ICU patients. You might have one file that has some basic demographic data (age, sex, SES, employment status, marital status, etc), another file with baseline health status upon admission to the ICU, and yet another file with daily data. For a particular analysis, you may need data from more than one of these files. SPSS can only have one working data file open at a time. So the way you cope with a situation like this is by merging files. In this section of the tutorial, we will see how to use the MATCH FILES command, which allows you to add new variables (from other data files) to the current working data file.

The three files we will use are taken from a study that examined the use of nitric oxide (NO to prevent ischemia-reperfusion injury after lung transplantation. Here's a brief description of the files:

File name	**Description**
ptinfo.sav	Demographic and baseline health information about the patients
donor.sav	Lung donor's age and last PaO2
dailymods.sav	Daily MODS for patients (MODS = multiple organ dysfunction score)

Notice that all 3 of these files contain variable MRN (medical record number). MRN is a key variable that we will use when merging files.

IMPORTANT: The records in all files you wish to merge must be sorted in ascending order of the key variables. If they are not all sorted this way, the merge will not work.

To ensure that your 3 files are all sorted in ascending order of MRN, open each one in turn, then run the following syntax, and save the file. (The `(a)` in the `sort` command is for *ascending order*. You would use `(d)` for *descending order*.)

```
sort cases by mrn(a).
```

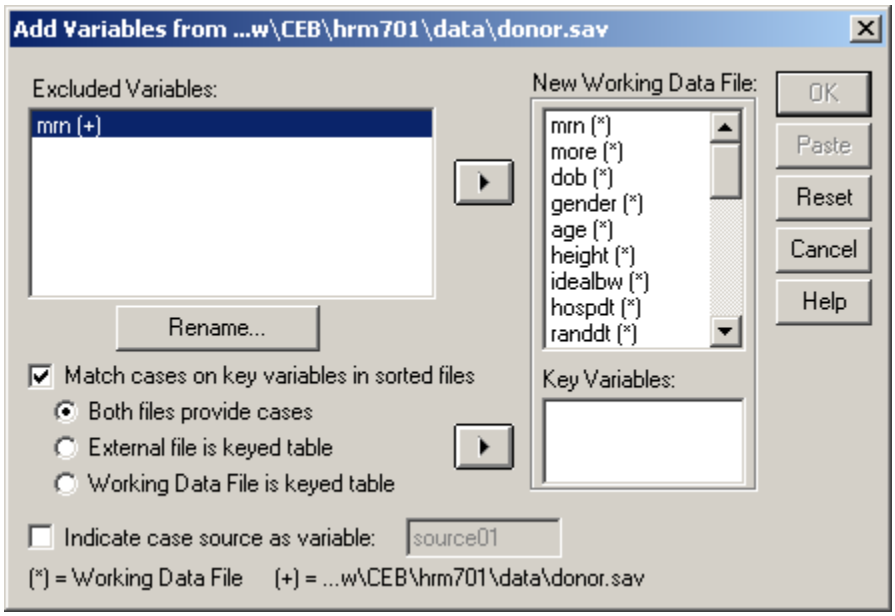
Merging two files that each have one record per patient

1. Open file ptinfo.sav (using PASTE rather than OKAY)
2. With the data editor active, click on Data→Merge Files→Add Variables
3. Select file donor.sav, then click Open
4. In the dialog box that appears (see below):
    a. click on MRN in the Excluded variables box
    b. put a check beside "Match cases on key variables…"
    c. move MRN into the Key Variables box
    d. click on PASTE
    e. note the warning about sort order of Key Variables, and click Okay

f. Click NO when prompted about saving the working data file

5. Run the `MATCH FILES` syntax that has been generated

Your data file should now contain 4 new variables that were not there previously: `morerec`, `moredon`, `agedonor`, and `lastpao2`.



Adding variables from a file with multiple records per patient

Now let's try adding the variables found in the Daily MODS file.

1. With the data editor active, click on Data→Merge Files→Add Variables
2. Select file dailymods.sav, and click Open
3. In the dialog box that appears, highlight MRN in the Excluded Variables box, put a check beside "Match cases on key variables…", and move MRN into the Key Variables box
4. Click on PASTE, then click Okay when you see the warning about sort order for key variables, and NO when prompted about saving the working data file
5. Run the syntax
6. Note the warning in the output window, and look at your data file

Your output window will now contain a warning message like this:

```
>Warning # 5132
>Duplicate key in a file.  The BY variables do not uniquely identify
each
>case on the indicated file.  Please check the results carefully.
```

Why did this warning occur? Because we tried to merge a file with multiple records per patient with a working file that had only one record per patient. Notice in your data file that the variables from *ptinfo.sav* and *donor.sav* appear on only the first record for each patient. On the other records, these variables contain the system missing indicator.

Now let's try the merge in another way such that the variables from *ptinfo.sav* and *donor.sav* appear on all records rather than just the first record.

Run the following syntax to return the working data file to the state it was in before we attempted merging with dailymods.sav:

```
match files file = *
 /drop = date to mod.
select if not missing(more).
exe.
```

Your data file should once again have one record per patient (n=20) with all of the variables from files *ptinfo.sav* and *donor.sav*.

Now attempt another merge with dailymods.sav as follows:

1. With the data editor active, click on Data→Merge Files→Add variables
2. Select file dailymods.sav, and click Open
3. Highlight variable MRN in the Excluded Variables box, put a check by "Match cases on key variables…", and move MRN into the Key Variables box
4. Click on the radio button beside "Working data file is keyed table"
5. Click on PASTE, and deal with the various warnings as before
6. Notice now the `MATCH FILES` syntax differs from the previous example (watch for FILE versus TABLE in the syntax)
7. Run the syntax
8. Note the absence of a warning message, and look at the data file

Your file should now show complete data for all variables on all records (except for any cases where there was missing data to start with, of course).

Let's try one more merge, but this time with the Daily MODS file open first.

1. Paste the syntax to open file dailymods.sav, then run it
2. Paste the syntax to add variables from file ptinfo.sav, making sure to check the radio button beside "External file is keyed table" this time
3. Notice how this MATCH FILES syntax differs from the previous example (FILE versus TABLE)
4. Highlight and run the MATCH FILES syntax

How to Merge Files: Adding New Cases

Suppose you had two or more data files that were all identical in structure, and you wanted to stack them one on top of the other. This might arise, for example, if you had data files from several centres in a multi-centred study. You can stack the files very easily in SPSS using Data→Merge Files→Add Cases.

The ptinfo.sav file we have been using has a variable CENTRE. I split ptinfo.sav into 3 files, one for each centre. We will use files ptinfo1.sav, ptinfo2.sav, and ptinfo3.sav to see how file stacking works.

1. Paste and run syntax to open file ptinfo1.sav
2. With the data editor active, click on Data→Merge Files→Add Cases
3. Select file ptinfo2.sav, and click Paste
4. With the data editor active, click on Data→Merge Files→Add Cases
5. Select file ptinfo3.sav, and click Paste
6. In your syntax file, highlight the EXECUTE line from the first ADD FILES command, and the ADD FILES line of the 2nd ADD FILES command, and delete those rows;
7. This should leave you with one ADD FILES command with two /FILE lines; **IMPORTANT**: You must delete the period from the end of the first /FILE line
8. Highlight and run the ADD FILES line and the EXECUTE that follows it

You should now have a file with records from all 3 centres (i.e., the same as file ptinfo.sav).

How to flag the First and Last Records in a Group of Records

It is often necessary to flag the first or last record in a group of records (e.g., the first or last record for a patient in a daily data file). This is easy to do using either MATCH FILES or ADD FILES. Here's how it works:

Paste and run syntax to open file dailymods.sav
Paste the following syntax into your syntax window and run it

```
match files file = *
 /by mrn
 /first = firstrec
 /last = lastrec.
exe.
```

Your working data file should now have two new variables FIRSTREC and LASTREC. FIRSTREC = 1 on the first record for each unique medical record number, and 0 otherwise. LASTTREC = 1 on the last record for each unique medical record number, and 0 otherwise. Also, you could have used ADD FILES instead of MATCH FILES to produce exactly the same result.

How to use the AGGREGATE command

The following is from the SPSS help files:

> Aggregate Data combines groups of cases into single summary cases and creates a new aggregated data file. Cases are aggregated based on the value of one or more grouping variables. The new data file contains one case for each group. For example, you could aggregate county data by state and create a new data file in which state is the unit of analysis.
>
> **Break Variable(s).** Cases are grouped together based on the values of the break variables. Each unique combination of break variable values defines a group and generates one case in the new aggregated file. All break variables are saved in the new file with their existing names and dictionary information. The break variable can be either numeric or string format.
>
> **Aggregate Variable(s).** Variables are used with aggregate functions to create the new variables for the aggregated file. By default, Aggregate Data creates new aggregate variable names using the first several characters of the source variable name followed by an underscore and a sequential two-digit number. The aggregate variable name is followed by an optional variable label in quotes, the name of the aggregate function, and the source variable name in parentheses. Source variables for aggregate functions must be numeric.
>
> You can override the default aggregate variable names with new variable names, provide descriptive variable labels, and change the functions used to compute the aggregated data values. You can also create a variable that contains the number of cases in each break group.

Let's use the Daily MODS file to see how it works.

1. Paste and run syntax to open file dailymods.sav
2. With the data editor active, click on Data→Aggregate
3. In the dialog box that appears:
4. Move variable MRN into the Break Variable(s) box
5. Move variable DAILYMOD into the Aggregate Variable(s) box, then:
    a. Click on Name & Label
    b. Change the variable name to **dm.mean**
    c. Add the variable label **Mean daily MOD**
6. Move variable DAILYMOD into the Aggregate Variable(s) box again, then:
    a. Click on Name & Label
    b. Change the variable name to **dm.sd**
    c. Add the variable label **Daily MOD SD**, then click Continue
    d. Click on Function, and select Standard Deviation
7. Move variable DAILYMOD into the Aggregate Variable(s) box again, then:
    a. Click on Name & Label
    b. Change the variable name to **dm.min**
    c. Add the variable label **Minimum daily MOD**, then click Continue
    d. Click on Function, and select Minimum
8. Move variable DAILYMOD into the Aggregate Variable(s) box again, then:
    a. Click on Name & Label
    b. Change the variable name to **dm.max**
    c. Add the variable label **Maximum daily MOD**, then click Continue

   d. Click on Function, and select Maximum
9. Put a check by "Save number of cases…"
10. Select the "Replace working data file" radio button
11. Paste and run the AGGREGATE syntax

Your AGGREGATE syntax should look something like this (I cleaned it up a bit to make it more readable):

```
AGGREGATE /OUTFILE=*
  /BREAK=mrn
  /dm.mean 'Mean daily MOD' = MEAN(dailymod)
  /dm.sd 'Daily MOD SD' = SD(dailymod)
  /dm.min 'Minimum daily MOD' = MIN(dailymod)
  /dm.max 'Maximum daily MOD' = MAX(dailymod)
  /N_BREAK=N.
```

The working data file should now have one record per patient (n=19 patients) with only the 6 variables mentioned in the AGGREGATE command. Note that this example uses only a fraction of the functions that are available under AGGREGATE.

How to Restructure a Data File

It may be necessary on occasion to restructure your data file from so-called WIDE format (i.e., one row per person with multiple columns) to LONG format (i.e., several rows per person), or vice versa (e.g., the usual approach to repeated measures ANOVA requires WIDE format, but Mixed Linear Models, which could be used to analyze the same data requires LONG format). Recent versions of SPSS have a Data Restructure wizard that makes the job of restructuring your data a lot easier. In this section, we will look at some simple examples of how it words.

First, run the following syntax to create a WIDE format data file:

```
* Read in some data in WIDE format .

DATA LIST LIST /id (f2.0) a.1 b.1 c.1 a.2 b.2 c.2 a.3 b.3 c.3 (9f2.0).
BEGIN DATA.
1 1 2 3 4 5 6 7 8 9
2 3 2 1 5 3 4 . . .
3 1 1 1 . . . 3 3 3
4 1 2 3 . . . . . .
5 9 8 7 4 5 6 3 2 1
END DATA.

list all.

* There will be some warnings in the output window due
* to the missing values for some variables, but these
* are not important.
```

Now do the following:

1. With the data editor active, click on Data→Restructure
2. When prompted to save the working file, click NO
3. In the dialog box that appears, check the top radio button (Restructure selected variables into cases), then click Next
4. In the next dialog box, click the bottom radio button (More than one), and change the number from 2 to 3 (beside "How many?"); then click Next
5. In the next dialog box:
    a. Under **Case group identification**, select *Use selected variable* from the pull-down list, then move variable ID into Variable box
    b. Hold down the CTRL key and click on variables A.1, A.2, and A.3 (while holding down CTRL key)
    c. Release the CTRL key and move variables A.1 to A.3 into the **Variables to be transposed** box
    d. Click on trans1 to highlight it, and type the letter 'a' (to name the target variable a)
    e. Select trans2 from the pull-down list, and rename it to **b**
    f. Click on B.1, B.2, and B.3 while holding down the CTRL key, then move them into the **Variables to be transposed** box
    g. Select trans3 from the pull-down list, and rename it to **c**
    h. Click on C.1, C.2, and C.3 while holding down the CTRL key, then move them into the **Variables to be transposed** box
    i. Click Next
6. Click the top radio button (beside **One**), then click Next, Next and Next
7. In the Finish dialog box, click the bottom radio to Paste the syntax, then click Finish

The syntax you generated should look something like this:

```
VARSTOCASES
 /MAKE a FROM a.1 a.2 a.3
 /MAKE b FROM b.1 b.2 b.3
 /MAKE c FROM c.1 c.2 c.3
 /INDEX = Index1(3)
 /KEEP =  id
 /NULL = KEEP.
```

Run the syntax, and observe what happens to your data file. There are now 3 rows per person instead of 1 row per person.

Now let's see if we can use the Restructure Wizard to go in the opposite direction—i.e., to change the data format from LONG to WIDE.

1. With the data editor active, click on Data→Restructure
2. Click NO when prompted to save working data file
3. Click the middle radio button (Restructure selected cases into variables)
4. Move variable ID into the Identifier variable(s) box

5. Move variable index1 into the Index variable(s) box
6. Click Next, and Next again
7. Select the **Group by index** radio button, then click Next
8. Click the **Paste syntax** radio button, then Finish

Your syntax should look like this:

```
CASESTOVARS
 /ID = id
 /INDEX = index1
 /GROUPBY = INDEX .
```

Run the syntax, and observe that the data file has been restored to it's original (WIDE) format.

```
ID A.1 B.1 C.1 A.2 B.2 C.2 A.3 B.3 C.3

 1   1   2   3   4   5   6   7   8   9
 2   3   2   1   5   3   4   .   .   .
 3   1   1   1   .   .   .   3   3   3
 4   1   2   3   .   .   .   .   .   .
 5   9   8   7   4   5   6   3   2   1
```

These are just two simple examples of how to use VarsToCases and CasesToVars.

Save your syntax

That's the end of the tutorial. Save your syntax for future reference.
