Session 3 - SPSS[1]

Saving a SPSS data file. Reading a SPSS data file. Another way to read a hierarchical file. Matching (merging) SPSS data files. More on errors and error detection.

Saving and Accessing SPSS data files (AKA SPSS systems file, SPSS file)

So far we have used raw data – numbers that you can look at. An SPSS data file is different. You cannot look at it on the disk and read it. It is stored in binary format, and has special information that only SPSS (and other programs trained to understand SPSS). It has all of the information that was given to SPSS about variables before the file was saved. For example, if variable names were provided, or new variables were created, this information is stored in the file.

To save an SPSS data file, the following code is used.

```
save outfile = 'd:/fullpath/filename.sav'.
```

The file can be accessed with the command:

```
get file = 'd:/fullpath/filename.sav'.
```

More about Hierarchical Files, and Other Ways to Read Them.

Reading a hierarchical file is sometimes called "rectangularizing" it. In the small example we have been using, we have three household records, and seven person records. We end up with seven records, but more variables. For a tiny data set like this, storage and processing is not an issue. But, with real census data sets, storage can become an important issue. For example, with one of the 1990 1% IPUMS files (Integrated Public Use Microdata Series, see http://www.ipums.umn.edu/), there are 1,105,583 household records with 105 variables, and 2,500,052 person records with 149 variables. It takes much less space to store the 105 household variables 1.1 million times than it does to store them 2.5 million times. If these files were rectangularized, they would take far more space to store. Another options is to store the person and household records separately, and merge them when you need both household and person records. The following program reads the data, and saves two systems files.

```
* prog4.sps.
title 'prog4.sps' .
* Read the data and save separate files for Household and Person Records.
** The data (as a comment) follow:
 h011360010
 p0101125
 p0102222
 p0103302
 p0104301
 h023130010
 p0201145
 p0202325
 h034530010
 p0301166     .

** READ and save HOUSEHOLD cases and VARIABLES .
input program .
DATA LIST file = 'c:\all\spssclass\progs\prog3.data'   /RECTYPE 1 (a).
```

[1]This document has been prepared by Patty Glynn, University of Washington.

```
do if (RECTYPE eq 'h').
  reread.
     DATA LIST file = 'c:\all\spssclass\progs\prog3.data'
        / serial 2-3 region 4 statefip 5-6 county 7-10 stateco 5-10 .
end case.
end if.

** keep and save only household records and variables .
select if (RECTYPE eq 'h').
end input program.
variable label
 rectype = 'Record Type'
 serial  = 'Number unique to HH'
 region  = 'Geographic Region'
 statefip= 'State Fips Code'
 county  = 'County ICPSR Code'
 stateco = 'Fips State + ICPSR County' .

save outfile = 'c:\all\spssclass\progs\prog4hh.sav' .

** READ and save PERSON cases and VARIABLES .
input program .
DATA LIST file = 'c:\all\spssclass\progs\prog3.data'   /RECTYPE 1 (a).
do if  (RECTYPE eq 'p').
  reread .
     DATA LIST file = 'c:\all\spssclass\progs\prog3.data'
         / serial 2-3 pernum 4-5 relate 6 age 7-8.
end case.

end if.
end input program.
select if (RECTYPE eq 'p').
variable label
 rectype = 'Record Type'
 pernum  = 'Person number in HH'
 relate  = 'Relationship to head'
 age     = 'Age in years' .

save outfile = 'c:\all\spssclass\progs\prog4pers.sav' .
** program ends here .
```

Note that two “save outfile” commands are issued, saving two different data sets. The first saves the household cases and variables, and the second saves the person records and variables.

Variations on Merging - Wrong and Right.

It is common in social science to have to put to data sets together so that each case has more variables. For example, putting household and person variables for PUMS files, if they are stored separately would require merging.

In SPSS, the command to accomplish this is “MATCH FILES”. The syntax varies somewhat depending on the situation, as I will show later.

If both files have one case for each “variable in common” then the syntax for matching files is :

```
match files   file = ‘c:\directory\fileone.sav’ /
              file = ‘c:\directory\fileone.sav’ /
              by {variable(s) in common} .
```

While a BY statement is not always required, it is best to include one if possible. The likelihood of putting data together incorrectly is greater if you don’t use a “by” statement.

Examples of successful and unsuccessful merges, and explanations follow.

This a simple merge where there are the same observations in both data sets, and the observations are in the same order.

```
*** PROGRAM BEGINS HERE *** ;
* testmrg1.SPS.
title 'testmrg1.SPS - no by statement' .

** Read and save first data set.
DATA LIST FREE / Name (a9) g1 (A2) g2 (A2) g3 (A2) .
BEGIN DATA
Alice   a1 a2 a3
Brian   b1 b2 b3
Charles c1 c2 c3
Dianne  d1 d2 d3
END DATA.
save outfile = 'c:\all\spssclass\trash\f1.sav' .

** Read and save second data set.
DATA LIST FREE / Name (a9) g4(A2) g5 (A2) g6 (A2) .
BEGIN DATA
Alice   a4 a5 a6
Brian   b4 b5 b6
Charles c4 c5 c6
Dianne  d4 d5 d6
END DATA.
save outfile = 'c:\all\spssclass\trash\f2.sav' .

match files
   file = 'c:\all\spssclass\trash\f1.sav' /
   file = 'c:\all\spssclass\trash\f2.sav' .

list var = all .

NAME      G1 G2 G3 G4 G5 G6

Alice     a1 a2 a3 a4 a5 a6
Brian     b1 b2 b3 b4 b5 b6
Charles   c1 c2 c3 c4 c5 c6
Dianne    d1 d2 d3 d4 d5 d6


Number of cases read:  4    Number of cases listed:  4
```

Everything went well in this case because the cases matched in both data sets. But, suppose the data were not in the same order in both data sets.

```
*** PROGRAM BEGINS HERE *** ;
* testmrg2.SPS.
title 'testmrg2.SPS, no by statement, wrong - cases not match' .

** Read and save first data set.
DATA LIST FREE / Name (a9) g1 (A2) g2 (A2) g3 (A2) .
BEGIN DATA
Alice   a1 a2 a3
Brian   b1 b2 b3
Charles c1 c2 c3
Dianne  d1 d2 d3
END DATA.
save outfile = 'c:\all\spssclass\trash\f1.sav' .

** Read and save second data set.
DATA LIST FREE / Name (a9) g4(A2) g5 (A2) g6 (A2) .
BEGIN DATA
Brian   b4 b5 b6
```

```
Charles c4 c5 c6
Dianne  d4 d5 d6
Alice   a4 a5 a6
END DATA.
save outfile = 'c:\all\spssclass\trash\f2.sav' .

match files
   file = 'c:\all\spssclass\trash\f1.sav' /
   file = 'c:\all\spssclass\trash\f2.sav' .

list var = all .

NAME      G1 G2 G3 G4 G5 G6

Alice     a1 a2 a3 b4 b5 b6
Brian     b1 b2 b3 c4 c5 c6
Charles   c1 c2 c3 d4 d5 d6
Dianne    d1 d2 d3 a4 a5 a6


Number of cases read:  4    Number of cases listed:  4
```

Notice that the data for Brian and Alice are mixed up. The match did not go well, and our data are very wrong.

Now consider the following example. The data are not in the right order, and, data for Brian is missing from the second data set. Examine the names. Each case can have only one value for each variable, and the values for the variable "name" were different in the two data sets. One data set will overwrite the values of another. Note that the values from data set two overwrote the values for name in data set one.

```
*** PROGRAM BEGINS HERE *** ;
* testmrg3.SPS.
title 'testmrg3.SPS, no by statement, wrong - cases not match' .

** Read and save first data set.
DATA LIST FREE / Name (a9) g1 (A2) g2 (A2) g3 (A2) .
BEGIN DATA
Alice   a1 a2 a3
Brian   b1 b2 b3
Charles c1 c2 c3
Dianne  d1 d2 d3
END DATA.
save outfile = 'c:\all\spssclass\trash\f1.sav' .

** Read and save second data set.
DATA LIST FREE / Name (a9) g4(A2) g5 (A2) g6 (A2) .
BEGIN DATA
Charles c4 c5 c6
Dianne  d4 d5 d6
Alice   a4 a5 a6
END DATA.
save outfile = 'c:\all\spssclass\trash\f2.sav' .

match files
   file = 'c:\all\spssclass\trash\f1.sav' /
   file = 'c:\all\spssclass\trash\f2.sav' .

list var = all .

NAME      G1 G2 G3 G4 G5 G6

Alice     a1 a2 a3 c4 c5 c6
Brian     b1 b2 b3 d4 d5 d6
Charles   c1 c2 c3 a4 a5 a6
Dianne    d1 d2 d3
```

```
Number of cases read:  4    Number of cases listed:  4
```

Now we will add a by statement, and examine the results. And, this time we must look at the log to understand what happens. (Of course, we should always look at the log!)

```
*** PROGRAM BEGINS HERE *** ;
* testmrg4.SPS.
title 'testmrg4.SPS, wrong - cases not sorted' .

** Read and save first data set.
DATA LIST FREE / Name (a9) g1 (A2) g2 (A2) g3 (A2) .
BEGIN DATA
Alice   a1 a2 a3
Brian   b1 b2 b3
Charles c1 c2 c3
Dianne  d1 d2 d3
END DATA.
save outfile = 'c:\all\spssclass\trash\f1.sav' .

** Read and save second data set.
DATA LIST FREE / Name (a9) g4(A2) g5 (A2) g6 (A2) .
BEGIN DATA
Charles c4 c5 c6
Dianne  d4 d5 d6
Alice   a4 a5 a6
END DATA.
save outfile = 'c:\all\spssclass\trash\f2.sav' .

match files
   file = 'c:\all\spssclass\trash\f1.sav' /
   file = 'c:\all\spssclass\trash\f2.sav' / by name / map .

list var = all .
```

RESULTS:

```
NAME      G1 G2 G3 G4 G5 G6

Alice     a1 a2 a3
Brian     b1 b2 b3
Charles   c1 c2 c3 c4 c5 c6
File #2
     KEY: Alice

>Error # 5130
>File out of order.  All the files in MATCH FILES must be in non-descending
>order on the BY variables.  Use SORT CASES to sort the file.
>This command not executed.

>Note # 35
>There is no working file to be restored.  The state of the SPSS Processor
>is as if a NEW FILE command had been executed.
```

The matched data are not correct, and we got an error message. Data to be matched must be sorted in the order of the variable(s) in the by statement. If the files are already in that order, then it is not necessary to sort them. Sorting is a resource intensive task.

Efficiency note: Use keep statements and SELECT IF statements to eliminate variables and cases not needed before sorting.

```
*** PROGRAM BEGINS HERE *** ;
* testmrg5.SPS.
title 'testmrg5.SPS, cases sorted, by statement - ok' .

** Read and save first data set.
DATA LIST FREE / Name (a9) g1 (A2) g2 (A2) g3 (A2) .
```

```
BEGIN DATA
Alice   a1 a2 a3
Brian   b1 b2 b3
Charles c1 c2 c3
Dianne  d1 d2 d3
END DATA.
save outfile = 'c:\all\spssclass\trash\f1.sav' .

** Read and save second data set.
DATA LIST FREE / Name (a9) g4(A2) g5 (A2) g6 (A2) .
BEGIN DATA
Charles c4 c5 c6
Dianne  d4 d5 d6
Alice   a4 a5 a6
END DATA.
** The cases in this file are not sorted in the correct order .
** The following command asks SPSS to sort them .
sort cases by name .
save outfile = 'c:\all\spssclass\trash\f2.sav' .

match files
   file = 'c:\all\spssclass\trash\f1.sav' /
   file = 'c:\all\spssclass\trash\f2.sav' / by name / map .

list var = all .

match files
   file = 'c:\all\spssclass\trash\f1.sav' /
   file = 'c:\all\spssclass\trash\f2.sav' / by name / map .
```

Note the “/ map” in the above command line. This asks SPSS to tell you which file each variable came from in the resulting matched file. This is a very useful tool. The information this request provides is given below. Sometimes, variables might exist in two data set. Of course, each variable can only hold one value for each case. So, the value from one of the files will be lost in the resulting matched data set. The “MAP” option allows you to figure out which file “won”, and to determine if any variables were over-witten.

```
Map of BY variables

Result    Input1   Input2
------    ------   ------
NAME      NAME     NAME

Map of the result file
Result    Input1   Input2
------    ------   ------
NAME      NAME     NAME
G1        G1
G2        G2
G3        G3
G4                 G4
G5                 G5
G6                 G6

list var = all .
NAME      G1 G2 G3 G4 G5 G6

Alice     a1 a2 a3 a4 a5 a6
Brian     b1 b2 b3
Charles   c1 c2 c3 c4 c5 c6
Dianne    d1 d2 d3 d4 d5 d6
Number of cases read:  4
Number of cases listed:  4
```

Homework for next week:

Write a SPSS program to read the following hierarchical data set. Read the data as an external file. Create TWO systems files - one with household data and one with person data, then merge them. Save the final merged data set as a permanent SPSS data set.

The variables for the Household data set are:

- Record type
- household ID
- number of persons in the household
- region
- state
- county
- An indicator of whether it is in an MSA or not.

The variables for the Person data set are:

- Record type
- household ID
- person number
- relationship to head
- age
- race
- gender
- highest grade completed

```
H 1 3 1 AK 001 0
P 1 1 1 23 1 1 12
P 1 2 2 24 1 2 14
P 1 3 3  2 1 2  0
H 3 2 1 WA 001 1
P 3 1 1 67 2 1  9
P 3 2 2 65 1 2  9
H 2 4 1 AZ 001 0
P 2 1 1 34 1 1 15
P 2 2 7 36 1 1 16
P 2 3 3  7 1 2  2
P 2 4 3  5 1 2  0
```