

AN INTRODUCTION TO SPSS ON THE UNIX¹

SPSS is a packaged statistical program, like SAS, that can be extremely useful to social scientists. This document is designed as an introduction to SPSS, and will not substitute for the SPSS manuals. We assume that you are somewhat familiar with the UNIX system, and editing files. Please see the document <http://www.albany.edu/csda/classa.pdf> if you need information about using the UNIX system.

I. Using SPSS on the UNIX

SPSS code is very similar on different platforms. You may use any editor that can create an ASCII file to create your programs. Remember that the UNIX is case-sensitive. When you define data, be sure to use upper and lower case characters accurately.

```
get file='/full/path/filename.spssfile'
```

It is possible to save an SPSS systems file that will be deleted at the end of the SPSS job with the following command.

```
save outfile= "$SPSSTMPDIR/fileone.spssfile"
```

To access the file (within the same SPSS program) use the command:

```
get = "$SPSSTMPDIR/fileone.spssfile"
```

Type the following to run an spss program with the name of "test.spss": `spss -m -s 10m test.spss > test.lis &` where:

- `-m` runs SPSS in batch mode
- `-s 10m` reserves 10 megabytes for RAM memory
- `test.spss` name of batch program
- `> test.lis` redirects output into the named listing file
- `&` runs program in background so that you can work on something else while it runs

In order to simplify this command, we have put a shell program called "sps" in the /csda/help directory.

The shell program consists of the following line:

```
spss -m -s 10m "$1".spss > "$1".lis &
```

Make sure that your home directory is in your "path" in your .cshrc file, then change to your home directory and issue the following command.

```
cp /home2/generic/csda/sps sps
```

Once this shell program is in a directory that is in your path, you can type the following command to run the program "test.spss".

```
sps test
```

For this to work, your programs must have a file extension of ".spss", and it will automatically produce a listing file with the same filename as the program and a file extension of ".lis" (in this case, "test.lis").

Increasing "Scratch Files" Space

If your jobs will not run because there is not enough space on for scratch files, you may redirect where SPSS will write its scratch files. You may do this by issuing a command each time you login before you run an SPSS job, or you can incorporate the command in your .cshrc file. (This is not necessary on CSDA machines.) The command is:

¹Prepared by Kyle Crowder and Patty Glynn, CSDA, University at Albany. February 11, 1998. Updated December 21, 1999.

setenv SPSSTMPDIR '/full/path/name'

where '/full/path/name' is substituted with an existing directory to which have write access, and that has sufficient free space.

III. CREATING AN SPSS SYSTEMS FILE, READING AND DEFINING DATA

The first step toward data analysis is to read data. Data can come in different forms. In most cases, you will start out by reading a raw data file. For example:

1	35	33	44	ARIZONA	45
3	44	33	55	MAINE	-9
4	32	33	44	OREGON	23
5	34	44	33	GEORGIA	34
2	32	77	33	WASHINGTON	-9

In the above example, each line, or row, represents a case. There are six variables for each case. SPSS reads raw data with DATA LIST statement. The DATA LIST statement provides SPSS with information about the variables it is going to read. Variable names can be up to eight characters long. The above data could be read as free-field or fixed-field. Data can be read as free-field if there is at least one space (sometimes two) between each variable, and missing data are not represented with blanks.

The above dataset could be read with the following free-field data list:

```
DATA LIST FREE FILE='/full/path/TEST.DATA' RECORDS = 1 /  
ID V1 TO V3 * NAME (A10) V4
```

When SPSS encounters a space, it realizes that it has come to the end of a variable, and goes on to read the next in the variable list. Unless SPSS is told otherwise, SPSS expects numbers. NAME is a character variable. You tell SPSS this with an "A" in parentheses. The "10" specifies the longest string for the variable called "NAME". The (A10) tells SPSS to allow up to 10 spaces for the character variable.

The above data could also be read as fixed-field with the following data list statement.

```
DATA LIST FILE='/full/path/TEST.DATA' FIXED RECORDS = 1 /  
ID 1-2 V1 9-10 V2 18-19 V3 27-28 NAME 33-45 (A) V4 46-47
```

This tells SPSS that it will find the information about ID in columns 1 and 2, about V1 in columns 9-10, and so on. With fixed-field input, missing data can be represented as blanks, and spaces can be embedded in character values.

Most datasets are fixed-field without spaces between values, requiring less space to store the data.

Before current technology, data and programs were stored on cards with holes punched in them. Cards could only hold 80 columns of data. As a hold-over from that time, each row of data is sometimes referred to as a card. Another hold-over from that time is that data are often stored with a width of just 80 columns. And, SPSS programs cannot have instructions beyond the 80th column.

Large datasets often have more than one record of data for each case or observation. SPSS needs more information when reading these datasets. The data list statement must specify the card number for each variable being read, and must tell SPSS how many cards there are for each case. For example, the ICPSR dataset, Demographic Characteristics of the Population of the U.S., 1930-1950, which has information about population and migration for counties and states, is a fixed format file (each variable is in the same location for each of the cases) and has 55 records for each case. Following is part of the data list statement that reads that dataset.

```

DATA LIST FILE='/full/path/DEM1940.DATA' FIXED RECORDS = 55
/1 M001 3-4 M002 5-8 M003 9-12
/2 M011 13-19 M012 20-26 M013 27-33 M014 34-40 M015 41-47
M016 48-54 M017 55-61 M018 62-68 M019 69-75
/4 M029 13-19 M030 20-26 M031 27-33 M032 34-40 M033 41-47
M034 48-54 M035 55-61 M036 62-68 M037 69-75
/54 M479 13-19 M480 20-26 M481 27-33 M482 34-39 (2)
M483 40-45 (2) M484 46-51 (2) M485 52-57 (2)
M486 58-63 (2) M487 64-69 (2) M488 70-75 (2)

```

In the above example, there are 55 records for each case, but variables are being read from some of the cards. The "RECORDS=55" tells SPSS that there are 55 cards for each case. You must tell SPSS how many cards there are per case, or the data will not be read properly.

Notice the "(2)" in the part of the data list statement that defines card 54. If there is not an embedded decimal in the data, and with many datasets there is not, you must tell SPSS how many digits should be to the right of the decimal. "M486 58-63 (2)" tells SPSS to read the variable M486 in columns 58 to 63 (of card 54) and that 2 digits should be to the right of the decimal. So, the digits "123456" would be read as "1234.56".

Spacing is important in SPSS programs. SPSS does not use punctuation, as SAS does, to determine when one statement ends, and another begins. Instead, it uses spacing. If a statement begins in COLUMN 1, it is assumed to be a new statement. Continuations of statements must be indented at least one column.

SPSS uses asterisks for comments. A comment is a non-executable statement that you can use to document your programs. Comments make your programs more readable, and will help you remember what your programs are doing, and why.

As mentioned above, variable names cannot be more than 8 characters long. It is hard to get a meaningful variable name in 8 letters. However, SPSS allows you to assign labels of up to 40 characters. You do this with a VARIABLE LABEL statement. For example, the following VARIABLE LABEL statement goes with the demographic dataset mentioned above. Notice that the labels are enclosed in single quotes. Regular quotes can also be used.

VARIABLE LABEL

```

M001 'M001=ICPR STATE CODE'
M002 'M002=UNIQUE ID'
M003 'M003=SEQUENTIAL ID'
M004 'M004=STATE NAME'
M005 'M005=COUNTY NAME'
M006 'M006=930-940 CNTY BOUNDARIES'
M007 'M007=940-950 CNTY BOUNDARIES'
M008 'M008=SPECIAL COUNTY INFORMATN'
M009 'M009=NEGATIVE ESTIMATE INDCTR'
M010 'M010=930 TTL MALES, ALL AGES'
M011 'M011=930 TTL MALES, <5'      M012 'M012= 930 TTL MALES, 5-9'
M013 'M013=930 TTL MALES, 10-14'  M014 'M014=930 TTL MALES, 15-19'
M015 'M015=930 TTL MALES, 20-24'  M016 'M016=930 TTL MALES, 25-29'
M017 'M017=930 TTL MALES, 30-34'  M018 'M018=930 TTL MALES, 35-44'

```

IV. TRANSFORMING DATA

Now that you know how to read raw data, you need to know what you can do with it. We will use the demographic dataset as an example.

In many datasets there are codes for missing data. For example, in the demographic dataset described above, 999999 is the missing value for many variables. It is very important that you tell SPSS that these values mean missing data. Missing data are not used in computing means, sums, regressions, etc. Failing to define values as missing can result in wrong answers to the

questions you ask.

```
MISSING VALUES M001 M002 M003 M011 M012 M013 M014 M015 M016 (999999)
```

Suppose you want to know the percent of males that are 20 to 29 in each county. You could create that variable with the following statements.

```
COMPUTE MALE2029=((M015+M016)/M010)*100
VARIABLE LABEL MALE2029 '% OF MALES THAT ARE AGED 20-29'
```

SPSS also has a SUM function. The statement $MALE2029=(SUM(M015,M016)/M010)*100$; could also be used. But, if there is missing data for M015 or M016, the results you get using a plus sign and the SUM function may not be the same. With the SUM function, SPSS will give you a non-missing answer if any of the values in the parentheses have non-missing values. However, using the plus sign, SPSS will give you a non-missing answer only if all of the variables added have non-missing values. This can be a very important difference. In this case, we would want to use the plus sign. Otherwise, we would have inaccurately low values for counties that were missing information for either M015 or M016.

SPSS has many useful functions available for both numbers and characters. Look in an SPSS manual for information about others.

DO REPEAT loops can be very useful in creating new variables, as well as transforming old ones. The following statements will calculate what percent of the population that each age group comprises, and assign the values to new variables called P011 to P029.

```
COMPUTE TOTPOP=M010
DO REPEAT
  A = M011 TO M029 /
  B = P011 TO P029 /
COMPUTE B =( A / TOTPOP ) * 100
END REPEAT
```

The DO REPEAT loop processes both arrays. When it is working with M011, it will work with P011, when it is working with M012, it will work with P012, etc. When doing a DO REPEAT like this take note of the order of the variables in your file -- if the variables are not listed in the same order, the results would not be correct. So, without typing in a cumbersome and lengthy set of COMPUTE statements, the DO REPEAT commands above define the following variables: $P011=(M011 / TOTPOP)*100$, $P012=(M012 / TOTPOP)*100$, and so on until $P029=(M029 / TOTPOP)*100$.

V. CREATING AND ACCESSING PERMANENT SPSS SYSTEM FILES

Reading raw data is time consuming for SPSS with big datasets, so it is often useful to create SPSS system files that you can access later. By creating a permanent SPSS system file, you can avoid having to redefine variables and variable labels every time you look at the data, thereby saving yourself time and computer resources. When you access the permanent SPSS system file, it will know everything that you told SPSS when you created the file. If you gave the variables labels, it will know the labels. If you transformed variables, or created new variables, those variables will be stored. If you told SPSS what values are missing, it will remember what you told it.

To save an SPSS system file use a SAVE OUT statement. For example, you might save the DEM1940 file with the following statement:

```
SAVE OUT='/full/path/DEM1940.SPSSFILE'
```

This command should follow all data transformations so that they are all saved in the new SPSS system file.

After you have run the program that defines the variables from the raw data and saved an SPSS system file, you can

access the system file with a "GET" statement instead of redefining all of the variable locations, variable names, and labels from the raw data. For example, you might have the following program called DEM1 to read the newly created system file and get descriptive statistics on all the variables.

```
* DEM1 SPSSX. - USE DATA CREATED BY DEM1940, AND PRINT MEANS, ETC, OF ALL VARIABLES.
GET FILE = '/full/path/DEM1940.SPSSFILE'
DESC VAR = ALL
```

VI. KEEP AND DROP STATEMENTS, AND "SELECT IF" STATEMENTS

Often you will want to work with only some of the variables, and some of the cases in a file. SPSS runs will take longer than necessary, and will require more disk resources than necessary if there are more cases or variables than necessary in your dataset. A KEEP or DROP subcommand can be used with the GET command to prune the number of variables used in the current analysis. For example, suppose you were using the Demographic Characteristics dataset, and were only interested in some of the variables that pertained to 1940. You could use the following statement to keep the variables that identify each county, and that pertain to 1940.

```
GET FILE = '/full/path/DEM1940.SPSSFILE'
  /KEEP = M001 M002 M003 M004 M005 M006 M007 M008 M009 M096 to M112
```

Notice that the KEEP subcommand is indented at least one space and begins with a "/" so that SPSS knows that this is a subcommand of the GET command on the previous line. Also notice that the list of variables after the equal sign contains the variables "M096 to M112." This tells SPSS to keep all the variables in the file between M096 and M112.

The DROP subcommand uses the same syntax as the KEEP statement but tells SPSS to keep all the variables in the file *except* those listed in the DROP subcommand. Both the KEEP and DROP subcommands can be used with the SAVE OUT command as well. Note that a KEEP or DROP statement used with the GET FILE command only applies to the current analysis and does not alter the structure of the SPSS system file you are working with unless you copy over it with a new SAVE OUT command.

To reduce the number of cases for the analysis, you can select a desired subset of cases by using a SELECT IF statement. For example, suppose you were only interested in the counties in Georgia. You could select these counties with the statement:

```
SELECT IF M001=44
```

Or, if you wanted only counties in the Georgia, Florida, Alabama, and Tennessee, you would have the statement:

```
SELECT IF M001=44 OR M001=43 OR M001=41 OR M001=54
```

VII. A SAMPLE SPSS PROGRAM

The following program accesses a raw data file, defines and labels the variables and designates their missing values, selects out a subset of cases, and creates two new variables. The SORT command is used to re-order the cases in the file in a way that will be useful in merging the saved system file in a later step. The SAVE OUT command saves an SPSS system file with only a limited number of variables as designated in the KEEP subcommand.

```
* DEM1940 SPSS. THIS PROGRAM CREATES AN SPSS FILE CALLED DEM1940
* WHICH HAS SELECTED DEMOGRAPHIC INFORMATION ON GEORGIA COUNTIES FOR 1940.
DATA LIST FILE = '/full/path/DEM1940.DATA' FIXED RECORDS = 55
  / 1 M001 3-4 M002 5-8 M003 9-12
  / 11 M096 41-47 M097 48-54 M098 55-61 M099 62-68 M100 69-75
  / 12 M101 13-19 M102 20-26 M103 27-33 M104 34-40 M105 41-47
```

```
M106 48-54 M107 55-61 M108 62-68 M109 69-75
/ 13 M110 13-19 M111 20-26 M112
/ 55
```

VARIABLE LABELS

```
M001 'M001=ICPR STATE CODE' M002 'M002=UNIQUE ID'
M096 'M096=940 TTL MALES, ALL AGES'
M097 'M097=940 TTL MALES, <5' M098 'M098=940 TTL MALES, 5-9'
M099 'M099=940 TTL MALES, 10-14' M100 'M100=940 TTL MALES, 15-19'
M101 'M101=940 TTL MALES, 20-24' M102 'M102=940 TTL MALES, 25-29'
M103 'M103=940 TTL MALES, 30-34' M104 'M104=940 TTL MALES, 35-39'
M105 'M105=940 TTL MALES, 40-44' M106 'M106=940 TTL MALES, 45-49'
M107 'M107=940 TTL MALES, 50-54' M108 'M108=940 TTL MALES, 55-59'
```

* CREATING VARIABLES FOR MERGING LATER

```
COMPUTE STATEID=M001
```

```
COMPUTE COID=M002
```

* KEEP ONLY CASES IN GEORGIA

```
SELECT IF (STATEID=44)
```

* ACCOUNT FOR MISSING VALUES

```
MISSING VALUES
```

```
M001 (99) M002 (9999) M003 (9999)
```

```
M008 (0) M009 (0)
```

```
M096 to M112 (9999999)
```

* SORTING CASES BY NEW STATE AND COUNTY VARIABLES

```
SORT CASES BY STATEID, COID
```

```
DESC VARS= ALL
```

```
SAVE OUT='/full/path/DEM1940.SPSSFILE'
```

```
/KEEP M001 M002 M003 M008 M009 M096 to M112
```

The following program would access the SPSS system file created in the above example.

* DEM19402 SPSS. THIS PROGRAM READS AN SPSS SYSTEM FILE CALLED DEM1940

* WHICH WAS CREATED BY THE PROGRAM DEM1940 SPSS.

* IT HAS INFO ON COUNTIES FOR 1940.

```
GET FILE='/full/path/DEM1940.SPSSFILE'
```

VIII. MERGING SPSS SYSTEM FILES: ADDING MORE VARIABLES

In data analysis, it is common to pull together data from a variety of sources. It is possible to do this with SPSS by merging two system files with a MATCH command. It is important to merge correctly. To do this, the cases in each file should have unique identifying numbers. Just as crucial, the two files must be sorted in the same way. If you are not sure that the cases in each file are sorted by the variable used to do the merge, you must use the SORT command to re-sort the files (see example above). Following is a simple example of merging two SPSS system files that are both already sorted by a common identifying variable called "ID." This example merges scores from two exams.

```
MATCH FILES FILE= '/full/path/EXAM1.SPSSFILE' /
           FILE= '/full/path/EXAM2.SPSSFILE' / BY=ID / MAP
```

```
DESC VARS=ALL
```

```
SAVE OUT='/full/path/EXAM12.SPSSFILE'
```

The MATCH command tells SPSS to match the two files using the common variable "ID" and the MAP subcommand

asks for a record of the variables added from each file to the new matched file. The DESC VARS command asks for descriptive statistics on all the variables in the new matched file and the SAVE OUT command saves an SPSS system file called EXAM12. After merging files, it is important to check your LISTING file and output to make sure that it was done properly. Make sure that the number of cases is correct and that any missing values that were assigned are truly missing. Use the CORRELATIONS VARS to get correlations including variables from both files and check the number of observations.

IX. ADDING SPSS FILES: ADDING MORE CASES

Suppose you had an SPSS system file called INT500 SPSSFILE A with interviews from 500 people. You just completed creating an SPSS system file called INT200 SPSSFILE A for another 200 people, and you want to add those cases to your original file. All of the people were asked the same questions, and both SPSS system files have the same variables. The following program would put these two files together and saves a systems file containing all 700 cases.

```
* INT700 SPSS. CREATE ONE FILE OUT OF INT500 AND INT200
* NOTE, BECAUSE THE FILES ARE ON THE A DISK AND HAVE THE
* FILETYPE OF SPSSFILE, THE FOLLOWING WILL WORK.
ADD FILES FILE= '/full/path/INT500.SPSSFILE' /
          FILE= '/full/path/INT200.SPSSFILE' /
DESC VARS=ALL
SAVE OUT='/full/path/INT700.SPSSFILE'
```

X. AN EXAMPLE OF AN SPSS PROGRAM WITH A VARIETY OF PROCEDURES

Following is an example of an SPSS program that merges two system files, saves the new system file keeping only a subset of variables, selects a subset of cases from the new file, creates new variables, and runs some basic procedures. The numbered lines are program items, and the non-numbered lines are information provided by SPSS.

```
For VM/SP HPO 5                SUNY AT ALBANY                License Number 19995
This software is functional through October 31, 1996.
```

```
1 0 * EXAMPLE1 SPSSX
2 0 * EXAMPLE SPSS PROGRAM FOR INTRO. TO SPSS HANDOUT
3 0
4 0 FILE HANDLE P1983 /NAME='/full/path/PSID1983.SPSSFILE'
5 0 FILE HANDLE P1984 /NAME='/full/path/PSID1984.SPSSFILE'
6 0
7 0 * MERGING 1983 AND 1984 FILES BY COMMON VARIABLE ID
8 0 * BOTH FILES ALREADY SORTED BY ID VARIABLE
9 0 MATCH FILES FILE=P1983/FILE=P1984/BY=ID/MAP
10 0
File '/full/path/PSID1983.SPSSFILE'
  Created: 24-OCT-96 09:47:49 - 14 variables
File '/full/path/PSID1984.SPSSFILE'
  Created: 24-OCT-96 09:46:52 - 14 variables
Map of BY variables
Result  Input1  Input2
-----  -----  -----
ID      ID      ID
Map of the result file
Result  Input1  Input2      Result  Input1  Input2      Result  Input1  Input2
-----  -----  -----  -----  -----  -----  -----  -----  -----
ID      ID      ID      FAMINC83  FAMINC83      NCHILD84      NCHILD84
HDRACE83  HDRACE83      WRKING83  WRKING83      HOUSST84      HOUSST84
AGE83    AGE83        WELFAR83  WELFAR83      PPERRM84      PPERRM84
FEMALE83  FEMALE83      POVT83    POVT83        EDUC84        EDUC84
MARR83    MARR83        REGION83  REGION83      FAMINC84      FAMINC84
```

NCHILD83	NCHILD83	HDRACE84	HDRACE84	WRKING84	WRKING84
HOUSST83	HOUSST83	AGE84	AGE84	WELFAR84	WELFAR84
PPERRM83	PPERRM83	FEMALE84	FEMALE84	POVT84	POVT84
EDUC83	EDUC83	MARR84	MARR84	REGION84	REGION84

```

11 0 SAVE OUTFILE='/full/path/PSID8384.SPSSFILE'
12 0 /DROP=PPERRM83 WRKING83 PPERRM84 WRKING84
13 0

```

File contains 23 variables, 184 bytes per case before compression
9,639 cases saved

0Preceding task required 1.98 seconds CPU time; 18.32 seconds elapsed.

```

15 0 * ACCOUNTING FOR MISSING VALUES IN NEW FILE
16 0 MISSING VALUES
17 0 POVT83 (998,999) POVT84(998,999)
18 0 EDUC83 (00,99) EDUC84 (00,99)
19 0 HDRACE83 (0,9) HDRACE84 (0,9)
20 0 REGION83 (9) REGION84 (9)
21 0
23 0 * SELECTING ONLY THOSE LIVING IN THE WEST IN 1983 OR 84
24 0 SELECT IF (REGION83=3 OR REGION83=5) OR
25 0 (REGION84=3 OR REGION84=5)
26 0
28 0 * CREATING VALUE LABELS FOR RACE VARIABLES
29 0 VALUE LABELS
30 0 HDRACE83 HDRACE84 1 'WHITE'
31 0 2 'BLACK'
32 0 3 'AMER. IND/ALUET/ESKIMO'
33 0 4 'ASIAN/PI'
34 0 7 'OTHER'
35 0
37 0 * COMPUTING AGE-SQUARED VARIABLE
38 0 COMPUTE AGE83SQ=AGE83*AGE83
39 0 COMPUTE AGE84SQ=AGE84*AGE84
40 0 variable labels
41 0 AGE83SQ '1983 AGE-SQUARED'
42 0 AGE84SQ '1984 AGE-SQUARED'
43 0
45 0 * ADJUSTING FAMILY INCOME TO 1983 CONSTANT DOLLARS
46 0 COMPUTE ADJINC83=(FAMINC83*1.00000)/1000
47 0 COMPUTE ADJINC84=(FAMINC84*.90639)/1000
48 0 VARIABLE LABEL
49 0 ADJINC83 'TOT. ADJUSTED FAMILY INCOME, 1983'
50 0 ADJINC84 'TOT. ADJUSTED FAMILY INCOME, 1984'
51 0
53 0 * COMPUTING A DUMMY VARIABLE INDICATING HOME OWNERSHIP
54 0 COMPUTE OWNER83=0
55 0 IF (HOUSST83=1)OWNER83=1
56 0 IF MISSING(HOUSST83)OWNER83=9
57 0 COMPUTE OWNER84=0
58 0 IF (HOUSST84=1)OWNER84=1
59 0 IF MISSING(HOUSST84)OWNER84=9
60 0 VARIABLE LABEL
61 0 OWNER83 'HOME OWNER, 1983'
62 0 OWNER84 'HOME OWNER, 1984'
63 0
64 0
65 0 * GETTING MEANS, SD'S, N'S, ETC. OF SOME VARIABLES
66 0 DESC VARS=AGE83 AGE84 ADJINC83 ADJINC84
67 0

```

-296 bytes of memory required for the DESCRIPTIVES procedure.

0Number of valid observations (listwise) = 3051.00
 0
 Variable Mean Std Dev Minimum Maximum Valid N Label
 AGE83 35.93 16.15 16.00 92.00 3393 AGE, TIME 1
 AGE84 36.11 16.08 16.00 90.00 3489 AGE, TIME 1
 ADJINC83 21.35 19.03 .00 199.66 3393 TOT. ADJUSTED FAMILY INCOME, 1983
 ADJINC84 20.66 19.78 -41.69 235.03 3489 TOT. ADJUSTED FAMILY INCOME, 1984

0Preceding task required .92 seconds CPU time; 5.92 seconds elapsed.

70 0 FREQ VARS=HDRACE83
 71 0 * GETTING FREQUENCIES OF RACE, 1983
 72 0

0There are 1,076,536 bytes of memory available. The largest contiguous area has 675,192 bytes.
 0Memory allows a total of 30,690 values accumulated across all variables.
 There may be up to 7,673 value labels for each variable.

0HDRACE83 RACE OF HEAD, T1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
WHITE	1.00	1153	30.1	34.0	34.0
BLACK	2.00	2140	55.9	63.1	97.1
AMER. IND/ALUET/ESKI	3.00	94	2.5	2.8	99.8
OTHER	7.00	6	.2	.2	100.0
	.	438	11.4	Missing	
	Total	3831	100.0	100.0	

0Valid cases 3393 Missing cases 438

0Preceding task required .24 seconds CPU time; 2.07 seconds elapsed.

74 0 CORRELATIONS VARIABLES=FEMALE83 AGE83 EDUC83 ADJINC83
 75 0 POVT83 POVT84
 76 0 * REQUESTING A CORRELATION MATRIX

0PEARSON CORR problem requires 816 bytes of workspace.

- - Correlation Coefficients - -

	FEMALE83	AGE83	EDUC83	ADJINC83	POVT83	POVT84
FEMALE83	1.0000	.0493**	-.0230	-.1078**	.0568**	.0478**
AGE83	.0493**	1.0000	-.2820**	-.1033**	-.0090	-.0079
EDUC83	-.0230	-.2820**	1.0000	.3864**	-.3068**	-.3111**
ADJINC83	-.1078**	-.1033**	.3864**	1.0000	-.4142**	-.4158**
POVT83	.0568**	-.0090	-.3068**	-.4142**	1.0000	.9201**
POVT84	.0478**	-.0079	-.3111**	-.4158**	.9201**	1.0000

0* - Signif. LE .05 ** - Signif. LE .01 (2-tailed) " . " printed if a coefficient cannot be computed

0Preceding task required .43 seconds CPU time; 2.98 seconds elapsed.

77 0 FINISH
 0 77 command lines read.
 0 errors detected.
 0 warnings issued.
 4 seconds CPU time.
 30 seconds elapsed time.
 End of job.