DEBUGGING AND DATA CLEANING TECHNIQUES WITH SAS[1]

When working with large files, debugging can be time consuming. You can wait a long time for SAS to tell you (in a secret code that you must learn to decipher) that you have misplaced a semicolon or misspelled a word. If you have an error in logic, SAS probably will not be able to identify this. You must coax SAS into helping you find these problems. In this document, I will describe a variety of ways that can help you debug a program, and you will begin to learn to interpret SAS's secret code.

Look at Your Program

If you are using the SAS enhanced editor in version 8 or later, your first step in debugging can be to look at the program. You commands are color coded.

```
* Green commands are comments.  If you forget to put a semicolon at the end of a
comment, your comment will extend into what YOU think is a command.  SAS will disagree,
and SAS always wins this kind of argument. ;

LOOK FOR RED.  Red means that SAS doesn't like the command. ;
```

Blue is good. SAS understands the command, and it is in the right place. Black is neutral. It may or may not be spelled right. SAS is indifferent to purple - the text in a title or a label. BUT, if you see purple where you shouldn't, it means that you may have a missing single or double quote to end a label or title.

```
if region = 1 or region = 2 or region = 4 then south = 0 ;
if region = 3 then south = 1 ;
label south   = 'In South' ;
```

Debug with Fewer Cases

Please note the following: `options compress = yes obs = 0 ;`

The above options line asks SAS to use no observations when the program is run. The program will run very quickly, and you can see if there are misplaced semicolons or misspelled words. In interactive SAS, a setting will be remembered until you change it. So, in order to work with all of your cases again, it is not sufficient to change the line to: `options compress = yes ;` You must change it to:
`options compress = yes obs=max;`

If you set OBS to 0, you will not get output (you will get a log). Examining output is one of the ways that you can tell if your program is doing what you want. You can also set the number of observations to some number between 0 and max. This way you can get output, and save time in debugging if you have a very large data set. If you set obs to 1000, the first 1000 cases will be used.

Sometimes data sets are organized in such a way so that you do not get the variation you need if you select only the cases that are at the beginning of the file. Another option – which will not save as much time because all of the cases must be retrieved first – is to create a random sample. SAS has a several random number generating functions, one of which is "ranuni". The following line would assign a random number that ranges in value from 0 to 1 to each case. The number inside the parentheses is called a seed. Changing the seed will change the values of the random variable that is created. The following lines would create a 10 percent random sample (approximately).

```
ranvar1 = ranuni(395) ;
if ranvar1 le .1 ;
```

---

[1]Prepared by Patty Glynn, University of Washington. March 4, 2001 C:\all\help\helpnew\debug.wpd

Another option is to create a very small sample data that you can use to figure out what is going on. For example.

```
* debug2.sas ;
title1 'debug2.sas' ;
* Read data from within program ;
* Create temporary SAS data set ;

options compress = yes ;

data one;
input name $ age hinch weight gender $ ;
* Note that a variable has been misspelled below. ;
* Error message resulting from this will be shown below. ;
hwrat = hinch / wieght ;
label hwrat = 'hinch / wieght'
      name  = 'First name'
      age   = 'Age in years'
      hinch = 'Height in inches'
      weight= 'Weight in pounds' ;
datalines ;
George 22 72 203 m
Frank  67 65 180 m
Sally  27 62 120 f
Michelle 33 66 145 f
;
proc print ;
proc freq ;
run ;
```

Identifying Some Common Errors

Look in the log for errors and warnings. Debug from top to bottom. Once there is an error, SAS gets totally confused and is an unreliable witness regarding what might be right or wrong in the rest of the program. SAS will let you know if it finds something wrong by underlining text, and screaming ERROR. Find the first notice of a problem, and look from that point and BEFORE for the problem. Correct the problem and rerun the program. Don't trust SAS to correctly tell you about further errors after the first one.

At my last count, there were over a million kinds of mistakes that I have personally made in SAS. I won't be able to show all of them to you're here, but will try to expose you to a few so that you will recognize them.

The most common error in SAS is probably misplacing a semicolon. It takes practice to understand SAS's messages. In this case, there is a missing semicolon in line 101, but SAS starts underlining items on the next line.

```
101  data one
NOTE: SCL source line.
102  input name $ age hinch weight gender $ ;
               -                       -
               22                      22
                 ---                     -
                 202                     200
ERROR 22-322: Syntax error, expecting one of the following: a name, a quoted string, (, /, ;,
              _DATA_, _LAST_, _NULL_.
ERROR 202-322: The option or parameter is not recognized and will be ignored.
ERROR 200-322: The symbol is not recognized and will be ignored.
```

The second-most common error is probably trying to modify a data set after it has been written. New variables cannot be created and sub-setting IF statements cannot be applied after a "proc" statement, or a run statement.

```
89   proc print ; var ranvar1 ; format ranvar1 6.4 ;
NOTE: SCL source line.
90   newvar = 10 * ranvar1 ;
     ------
     180
ERROR 180-322: Statement is not valid or it is used out of proper order.
```

If you try to merge data sets that are not sorted in the right order, you will get a message like the following:

```
174  data onetwo ; merge one two ; by id ; run ;
ERROR: BY variables are not properly sorted on data set WORK.TWO.
id=4 a=a b=a FIRST.id=1 LAST.id=1 _ERROR_=1 _N_=4
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 4 observations read from the data set WORK.ONE.
NOTE: There were 2 observations read from the data set WORK.TWO.
WARNING: The data set WORK.ONETWO may be incomplete.  When this step was stopped there were 3
         observations and 3 variables.
```

It is also important to look at notes and warnings in the log. The following note can mean that your data are note what you expect.

```
314  data onetwo ; merge one two ; by id ;
NOTE: MERGE statement has more than one data set with repeats of BY values.
```

Another note that you may find in your log file that is important is an indication that a variable is not initialized.

```
NOTE: Variable wieght is uninitialized.
```

This may be because it was not included in a keep statement, or because it has been misspelled.

SAS Can't Find Some Problems – it Is up to You!

1. Check you N of cases. When you merge data, rarely should you end up with more cases than you had in any of the files you merged. If you end up with more cases, probably something went wrong.

2. Be suspicious. Look for face validity. If things seem too odd to be true, it might be because of a programming error, or dirt in your data.

3. There are many problems that SAS cannot find for you. If you make a mistake in logic when creating a variable, SAS won't know it. A simple error in a keystroke can make your variable completely wrong. RUN TESTS, LOOK AT OUTPUT! For example:

```
if region = 1 or region = 2 or region = 4 then south = 0 ;
if region = 3 then south = 1 ;
proc freq; tables south * region; run ;
```

The FREQ Procedure
Table of south by region

south	region					
Frequency / Percent / Row Pct / Col Pct	1	2	3	4	9	Total
0	1 25.00 33.33 100.00	1 25.00 33.33 100.00	0 0.00 0.00 0.00	1 25.00 33.33 100.00	0 0.00 0.00 .	3 75.00
1	0 0.00 0.00 0.00	0 0.00 0.00 0.00	1 25.00 100.00 100.00	0 0.00 0.00 0.00	0 0.00 0.00 .	1 25.00
Total	1 25.00	1 25.00	1 25.00	1 25.00	0 0.00	4 100.00

Frequency Missing = 1

4. Beware of missing values and dirt in your data. Run frequencies, means, or univariates to make sure that the values are within the range you expect. Following is an example of how you can change a value to missing. Missing values will not be included in analysis - but the cases are not excluded from the file. They can accidentally be included later by mistake (see the next point).

```
if var1 = -9999 then var1 = . ;
```

5. When you create a variable, make sure that you do not mistakenly include missing values in it. A missing value, in SAS, is a very small negative number. Note the following:

```
if var1 le 3 then newvar = 1 ; else newvar = 0 ;
```

Obs	var1	newvar
1	1	1
2	3	1
3	4	0
4	5	0
5	.	1

Data Cleaning Techniques

It is important to examine your data - especially for the variables you will use. Data cleaning is especially important for data you are preparing. Errors can creep in at many points. People can make mistakes in recording their answers. Interviewers can make mistakes. Errors can be made at the time of data entry, and programmers can make mistakes in variable creation from original variables.

Even data from reliable sources can have errors. For example, in ICSPR Study 8426, “Census of Population and Housing, 1980 [United States]: Summary Tape File 4C SMSA Extract”, in some records, there are redundant cases at the SMSA level for 16 SMSAs. This is a problem that could result in the 16 SMSAs being counted twice. And, if the data were merged with person-level data, all of the people in those SMSAs would be counted twice, if the error were not detected.

SAS procedures can be used to help detect errors. For example, proc freq you can check the

frequencies on an ID variable to make sure there is only one case for each ID. And, proc freq can also be used with categorical variables to make sure that only expected values are present. Proc means and univariate can be used to make sure the minimum and maximums are within the expected range, and means are about what you would expect. Compare means for different groups. Do they conform to what you expect?

Are there variables in the data set from which you can internal checks? Are there ways to check for inconsistencies? With SAS, you can use IF-THEN statements to create flags to alert you to problems. IF information about marriage is asked in more than one question, a flag could be created to note problems. For example, if someone reports in one place that she is currently married, and in another place reports that she has never been married, a flag could be created

```
if curmar = 1 and nevmar = 1 then flagmar = 1 ;
```

While it is possible in SAS for the PC to change values directly in the ViewTable, I DO NOT recommend it. If you change data in this way, you do not have a record of changes made. Instead, I recommend creating a program that gets the original data, makes corrections, document the changes, and saves a new data set. For example:

```
* cleandat.sas ;
title1 'cleandat.sas - Get dirty data, change values, save cleaned';
* This program is used to correct errors found in data.;
* Each time an error is detected, code is written to document ;
* the error, and to correct it.  The original (dirty) data set ;
* will never be deleted, and is always the one accessed by this ;
* program.  The cleaned data is over-written each time. ;

libname mylib 'c:\all\sasclass\saslib' ;

data mylib.cleandat; set mylib.dataold ;

* 2/22/2001, PJG, I checked form, and found a data entry error was made for ;
* case = 725, var1 - should be 10, was listed as 100 ;
if case = 725 then var1 = 10 ;

* 12/03/2001, PJG.  I found that there were several cases that answered ;
* no to var10, but answered yes to var20.  These are contradictory answers;
* I used var32 as an indicator of which is more reliable.
* I checked with other team members to make sure that they agreed with this.;
if var10 = 0 and var20 = 1 and var32 = 1 then var10 = 1 ;
if var10 = 1 and var20 = 0 and var32 = 1 then var20 = 1 ;
if var10 = 0 and var20 = 1 and var32 = 0 then var10 = 0 ;
if var10 = 1 and var20 = 0 and var32 = 0 then var20 = 0 ;
run ;
```