Paper 098-2010

SAS® State of Mind: A Guide to Learning SAS for the Stata User

Sara Beck, Fred Hutchinson Cancer Research Center, Seattle, WA

ABSTRACT

This paper is meant to assist those already familiar with Stata with a goal of learning SAS®. Due to the wealth of macros already available and support for SAS there are many advantages to learning SAS over other analysis tools. Because SAS and Stata are very similar, if one is already familiar with Stata and can adjust their mindset around the conceptual differences, learning SAS can be done with relative ease. This paper contains the following:

- A comparison between the way SAS and Stata store and modify data.
- Important differences in basic SAS and Stata syntax.
- Coded examples of simple tasks in both SAS and Stata.
- A brief introduction to the SAS windows interface.

The goal is to help the Stata user rely on their existing knowledge to more quickly and easily learn SAS. This document is intended to cover only basic principles. Because SAS and Stata do share so many similarities, this paper strives to highlight their differences.

INTRODUCTION

SAS and Stata are fairly similar. On their basic levels everything that can be done in Stata can be done in SAS. Once many of SAS's nuances are learned, 'programming' in SAS should be very intuitive to the Stata user.

DATA AND MEMORY

SAS and Stata treat datasets very differently. The three biggest differences are the following:

- With Stata one is required to estimate how much memory will be used when a data set is loaded. In SAS there is no such requirement.
- A single Stata command modifies the entire data set at one time, whereas a SAS statement modifies a single observation or row at one time.
- Very much unlike Stata, SAS only keeps one observation in its memory at any given moment of time. Because only one observation is in memory at once, certain SAS algorithms will not produce the desired result without a `RETAIN` statement.

THE RETAIN STATEMENT

The RETAIN statement keeps information from the previous observation alive in SAS's memory. Because Stata has the entire dataset in memory, there is no need to use such a RETAIN statement. The following example is a simple one where we sum the number of cups of brewed coffee each coffee shop serves.

Example 1. Consider the data set CoffeeSales. Each observation is an entire drink order made by a customer. A few observations might look something like this:

ShopName	Brew	Mocha	Latte	Cappuccino	Americano
Jitters	2	0	1	0	0
Jitters	1	0	0	0	0
Jitters	0	4	0	0	0
Jitters	0	0	1	0	0
Solid Grounds	0	0	0	1	0
Solid Grounds	0	0	0	0	1
Solid Grounds	0	0	0	0	1
Caffeine Cafe	0	1	0	0	0
Caffeine Cafe	1	0	0	0	0
Caffeine Cafe	1	0	0	0	0
Caffeine Cafe	0	0	1	0	0

We would like to calculate the cumulative total number of cups of brewed coffee sold at all coffee shops in the data set. We will assume the datasets are already in memory.

First, to emphasize the importance of the RETAIN statement, lets examine SAS results without its use. The code might look something like:

SAS without the RETAIN Statement

```
DATA CoffeeSalesNoRETAIN;
    SET CoffeeSales;
    TotalBrew=0;
    TotalBrew=SUM(TotalBrew,Brew);
RUN;
```

An approach like this may work in other software, but in SAS it does not. SAS does not remember what value total brew had in the previous observation. Because SAS processes each observation at a time, every time it sees a new row, SAS sets TotalBrew to zero, and then adds that row's value of Brew to it. The code above will not yield the desired results as it has no memory of TotalBrew's value before it.

```
                              Coffee Sales No RETAIN

                                                                                   Total
  Obs     ShopName          Brew     Mocha     Latte    Cappuccino     Americano     Brew
   1      Jitters             2        0         1          0              0           2
   2      Jitters             1        0         0          0              0           1
   3      Jitters             0        4         0          0              0           0
   4      Jitters             0        0         1          0              0           0
   5      Solid Grounds       0        0         0          1              0           0
   6      Solid Grounds       0        0         0          0              1           0
   7      Solid Grounds       0        0         0          0              1           0
   8      Caffeine Cafe       0        1         0          0              0           0
   9      Caffeine Cafe       1        0         0          0              0           1
  10      Caffeine Cafe       1        0         0          0              0           1
  11      Caffeine Cafe       0        0         1          0              0           0
```

As we can see from our results, as we expected, `TotalBrew` has the exact same values as `Brew`. We'll try again, but now with the `RETAIN` statement. Our code should resemble the following:

SAS	Stata
`DATA CoffeeSales;`	
`    SET CoffeeSales;`	
`    RETAIN TotalBrew 0;`	`gen TotalBrew=sum(Brew)`
`    TotalBrew=SUM(TotalBrew,Brew);`	
`RUN;`	

Our results are also much more favorable, as `TotalBrew` is now the cumulative sum of `Brew`:

```
                                    Coffee Sales

                                                                                   Total
  Obs     ShopName          Brew     Mocha     Latte    Cappuccino     Americano     Brew
   1      Jitters             2        0         1          0              0           2
   2      Jitters             1        0         0          0              0           3
   3      Jitters             0        4         0          0              0           3
   4      Jitters             0        0         1          0              0           3
   5      Solid Grounds       0        0         0          1              0           3
   6      Solid Grounds       0        0         0          0              1           3
   7      Solid Grounds       0        0         0          0              1           3
   8      Caffeine Cafe       0        1         0          0              0           3
   9      Caffeine Cafe       1        0         0          0              0           4
  10      Caffeine Cafe       1        0         0          0              0           5
  11      Caffeine Cafe       0        0         1          0              0           5
```

In both SAS and Stata commands, `TotalBrew` for the *jth* observation represents the sum of brewed cups of coffee for the 1 to *jth* observations. The very last observation in both datasets would be the total number of brewed coffee cups sold.

A LITTLE BIT OF SYNTAX

The syntax differences between SAS and Stata aren't terribly different from syntax differences between SAS and other mathematical software packages. Some of the more common differences between SAS and Stata include the following:

NAMING

In Stata, variable names are case sensitive; variable names in SAS are not. Where `VaRiAbLe`, `variable` and `VARIABLE` are all different variable names in Stata, in SAS they are the same. SAS and Stata have similar naming rules otherwise:

- No blanks can appear within a SAS name.
- The underscore is the only special punctuation character permitted.
- Names can start with either _ or any letter A-Z, each following digit can contain numbers, letters or the underscore.
- The allowed name length in SAS varies based on what is being named. For variables, the maximum SAS name length is 32.

As in Stata, it's good practice to avoid using the underscore as the first character in a variable name, as there are many built in SAS variables starting with the underscore (`_ALL_`, `_N_`, `_ERROR_`, etc.).

STARTING A NEW LINE

Semicolons delimit lines of code in SAS, whereas the invisible carriage return delimits lines in Stata. This isn't a shocking difference or anything remotely difficult to do; however, one can easily avoid many future headaches and errors by being sure to always include the semicolon at the end of the statement or line.

MISSING VALUES

Very much like Stata, missing values in SAS are denoted as a period ("`.`") when numeric and as a blank space (" ") when character. SAS and Stata have the same missing character expression. SAS and Stata are drastically different in how values of missing values are evaluated. A missing numeric values in SAS represents the smallest number possible, whereas in Stata it represented the largest possible value. There are options for extended missing values in both software packages. SAS has a few additional options for extended variables over Stata. The following table illustrates the differences in the comparative values of missing numeric values.

Table 1. Represented values of missing numbers

	SAS	**Stata**
Smallest	`._`	All non-missing numeric values
	`.`	`.`
	`.A` (smallest)- `.Z` (largest)	
Largest	All non=missing numeric values (including negative)	`.A` (smallest)- `.Z` (largest)

Most of the time the representation of a missing value will make no difference. However, when sorting or ordering a data set a certain way, it is wise to pay attention to the represented value as that could impact the final sorted order. Another situation where beginning programmers are prone to missing value related errors is with comparison signs. It is easy to forget that missing values are still included by expressions like `X<0`, as SAS considers missing values less than zero. When using statistical procedures it is important to be aware of how SAS is handling the missing values.

OPERATORS

SAS has a bit more flexibility with operator syntax than Stata. Certain words can be used in place of symbolic operators in SAS, whereas this is not the case in Stata. For example in SAS, `OR` and `|` can be used to represent "or" in a line of code. In Stata, `|` is the only valid way to do this. Another striking difference between the two packages is that in SAS order does not matter when typing an operator. `>=`, `=>` and `gt` all mean "greater than or equal to" to SAS. SAS and Stata are also different in the way they reference groups of variables. In SAS, `date1-date5` is equivalent to:

```
date1    date2    date3    date4    date5.
```

In Stata, `date1-date5` refers to all variables physically located between `date1` and `date5` in the data set. When coding in Stata, `blue*` could be any of the following:

```
blue    bluebook    blueberry    bluebird    blue1.
```

The equivalent to this in SAS would be `blue:`. Any character or numeric can follow the colon, or in the Stata case, the asterisk.

Table 2. Differing Operators in SAS and Stata

SAS	Meaning	Stata
`&, AND`	and	`&`
`^=, NE`	not equal to	`!=, ~=`
`|, or`	or	`|`
`=>,>=,ge`	greater than or equal to	`>=`
`=<,<=, le`	less than or equal to	`<=`
`>, gt`	greater than	`>`
`<`	less than	`<`
`**`	exponent	`^`
`prefix:`	varying characters after `prefix`	`prefix*`

CONDITIONAL LOGIC

The most marked difference between SAS and Stata in terms of conditional statements is order. In SAS we have a conditional, and then a statement, whereas the order is the other way around for Stata conditional lines. An example of the difference between how conditional statements are written in SAS and Stata follows. When making a conditional equality comparison Stata requires two equals signs (==). SAS only requires one equal sign.

When writing code, it can be easy to forget that SAS processes commands one observation at a time. Conditional statements can be especially prone to this error in thinking.

Example 2. Consider the data from the example above with the coffee shops. This time, we'd like to identify and create a new variable that indicates whether or not a customer ordered any espresso beverages at all. We'll construct a new variable called Espresso, which will have a value of 1 when at least one espresso beverage was purchased and 0 otherwise. We will again assume that both datasets are already in memory.

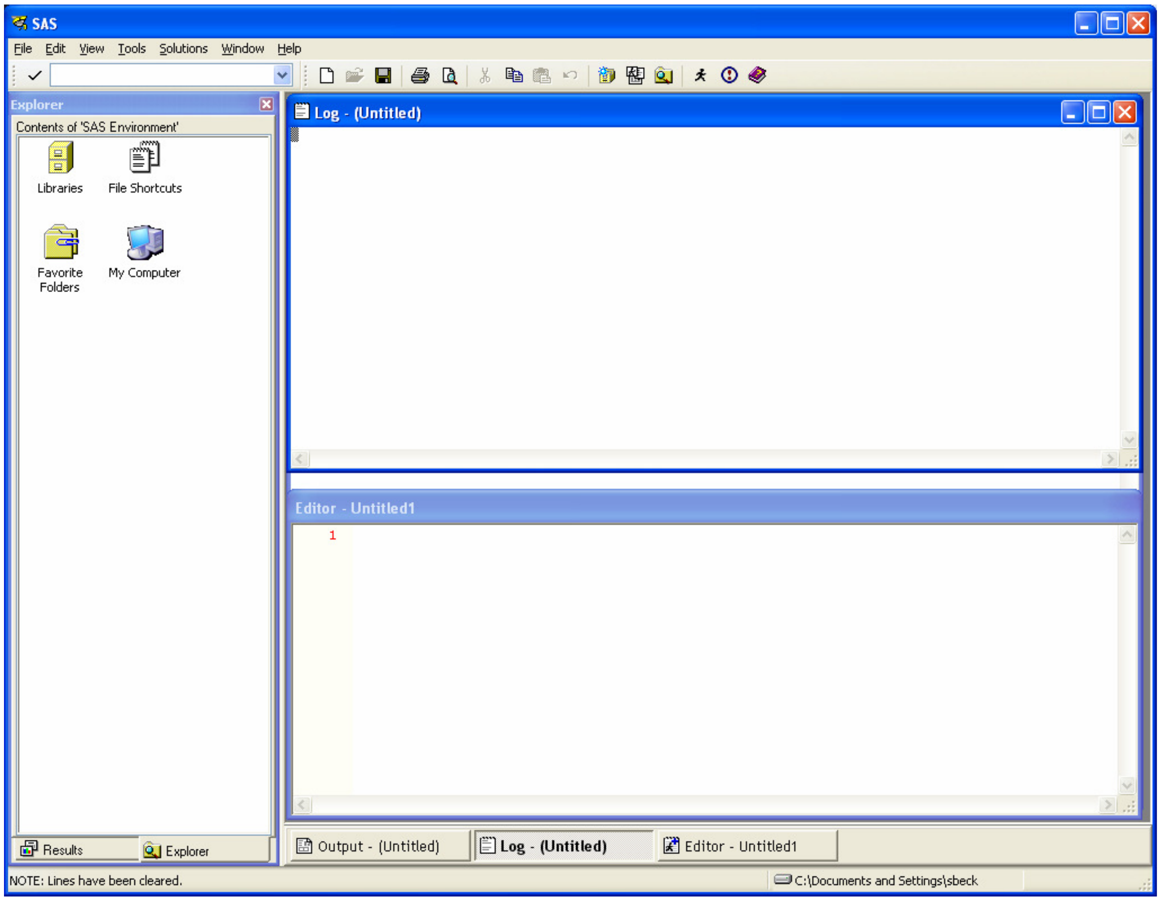
Stata

```
gen Espresso=0
replace Espresso=1 if Mocha >= 1 | Latte >= 1 | Capuccino >=1 | Americano>=1
```

SAS

```
DATA CoffeeSales;
  SET CoffeeSales;
  Espresso=0;
  IF Mocha ge 1 OR Latte ge 1 OR Americano ge 1 THEN Espresso=1;
RUN;
```

Figure 1. SAS interface upon opening.



NOTE: Lines have been cleared.

Also unlike Stata, SAS does not require `replace` or `generate` commands to set variables. In SAS variables can be set or changed by writing a simple statement stating what a variable will now be equal to.

SAS VERSUS STATA WINDOWS INTERFACE

When initially opening SAS, a window similar to Figure 1 should appear. Although this may look quite a bit different than Stata's interface, it is not so conceptually different. The initial SAS interface contains the following five windows: Explorer, Results, Log, Editor, and Output. Upon opening Stata, you may remember the windows interface has four windows: Review, Variables, Results and Command.

EXPLORER WINDOW

Rather than list the variables in memory, the SAS explorer window helps explore the data set environments. Here you can open the libraries to see the organization of data sets. Additional permanent libraries can be set by the user. Data sets in the `work` library are temporary and will no longer exist once the current SAS session ends. In Figure 1, SAS explorer window is the tall rectangular window furthest to the left.

RESULTS WINDOW

Whenever a SAS procedure creates output, an entry is added to the results window. The results window is simply an indexed list of SAS output.The results window is under the explorer window in Figure 1, and can be accessed by clicking the tab on the bottom of the explorer window.

EDITOR WINDOW

The SAS Editor window is similar to the Command window in Stata. Frequently Stata do-files are executed in the command window. Do-files are text files that contain many Stata commands. In this sense, macro statements in SAS are comparable to do-files in Stata. SAS Macro language is much too large of a topic to be touched in this paper. Multiple data steps, procedures, or macros can be written and processed after running the code once. In Figure 1, the editor window is located at the bottom right.

OUTPUT WINDOW

The SAS output window is really no different that the Stata results window. The output window simple displays results from SAS code run in the editor. The output window does not always display results after running code. The output window is layered behind the log and editor windows in Figure 1. Information is not always displayed in the output window after code is run. For instance, if a program simply modifies a data set, nothing will appear in the output window. Information will appear in the output window after running commands that return values.[1]

LOG WINDOW

The log in SAS provides important information about the current SAS session. Using the log is incredibly important in quality control and ensuring SAS code works the way it is intended. Most messages in the SAS log come in the following forms: ERROR, WARNING and NOTE.

ERRORS

When there's an error message in the log it should be very obvious, because your code probably didn't run a data step or procedure due to the nature of the error. SAS errors appear in a dark red color in the log.

WARNINGS

SAS will still complete a job despite warnings. Many times warnings are very harmless, but fairly often something strange or unexpected happened. Because there is great potential for the thing that triggered the warning to wreck havoc on

[1] The `listing` ODS destination must also be turned on (as per the default setting) for results to appear in the output window. This is not important to the beginning SAS programmer, but it is something to be aware of.

your data, it is important to confirm that the warning's message says something expected and appropriate. The default color for warnings in the SAS log is green.

NOTES

Notes usually contain information about what SAS procedure was used, or information about the dataset being manipulated or how long it took. More simply put, notes inform the uses of their job's status. Some of the information found in notes -- such as the number of variables or observations read from a data set can be very useful in quality control and debugging programs. In the SAS log the default color for notes is medium blue.

CONCLUSION

SAS and Stata are really not that different. SAS is an incredibly useful tool for analysis and data management. Once one already knows how to use Stata, one only needs to devote a little time to practice SAS, and one will be able to 'program' in SAS everything they knew how to 'do' in Stata.

REFERENCE

SAS Institute Inc. *SAS 9.2 Language Reference: Concepts, Order of Missing Values* Cary, NC: SAS Institute Inc. 2010.

ACKNOWLEDGEMENTS

The author would like to thank Nate Derby for his valuable advice and guidance that contributed to quality of this paper. Additional thanks to Stata users at Fred Hutchinson Cancer Research Center for their input regarding difficulties learning SAS.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sara Beck
Fred Hutchinson Cancer Research Center
sarjbeck@gmail.com