Paper SBC-129

Tips and Tricks to Make SAS® Life Easier

Claudine Lougee, Dualenic, LLC, Glen Allen, VA

ABSTRACT

What would you want to know if only given a day to learn SAS®? What SAS tips would make your life easier? What would you want another programmer to do for you in their code to help you "debug" their inefficiencies? This paper will attempt to show the beginner, intermediate, or even advanced SAS programmer that even an old dog can learn new tricks. Oh, and you might even learn why some errors occur!

INTRODUCTION

This paper covers a wide range of solutions and tips to learn new tricks, find errors, and/or make code more efficient. There is no particular theme other than providing some ad-hoc ideas that might be of interest to a wide range of audience. Beginners will learn what NOT to do, intermediate programmers might learn a new concept and for those with advance skills, there may be a solution to an old problem. Hopefully this paper covers at least one item of interest to all who decide to read it.

Sample syntax within the paper will be shown as follows:

```
LIBNAME TIPS "C:\SESUG\SIMPLE BUT CLEVER";

PROC CONTENTS DATA = TIPS._ALL_;
RUN;
```

ONE DAY TO LEARN SAS

If someone was to ask an experienced programmer how to code something or what would be the "fastest" solution, you might have a myriad of answers. This is one of the typical questions I have seen asked and have also asked myself to provide solutions for a client or customer. The answers over the years have varied, but with one common theme. "It depends". The hardest part, or most interesting part, of programming is finding solutions to problems. The good news is that answers are available for most, if not all questions. These are some sample Q & A topics for anyone just starting out or maybe a little bit rusty from all the elaborate code and just needs to get back to basics.

WHAT IS THE BEST WAY TO LEARN SAS?

This question is asked often by new programmers. Even skilled SAS programmers continue to add knowledge with the various SAS products available and continuous upgrade additions. Advice will vary depending on who is asked. Some answers might be: Take a class, read a book, go online, hire a tutor, practice using SAS Quick-start guide found within the Help Documentation. (Yes, there is actually help available within PC SAS software.) The answer to this question is really determined by the user's learning style. It's suggested to find out the type of student who is actually learning and match that style to the teaching method. If someone is a hands-on learner, they will not pick up a great understanding by reading a book.

WHERE CAN I FIND THE PROPER SYNTAX?

What is "proper syntax"? According to Wikipedia, syntax can be defined as "the rules and principles that govern the sentence structure of any individual language". The basic principal of any language is to understand the rules and then put them into a structure to use them for a purpose. SAS is a language which has a set of rules. You might learn the proper rules before understanding the syntax. It's a bit of trial and error, but knowing that a Data step needs a Run statement is not exactly true. In fact, you can run several Data steps without a Run statement, as long as you know the rules.

WHAT ADVICE CAN YOU GIVE A NEW PROGRAMMER?

Ah, advice, well who really wants to hear advice when just starting out? We all just really want to know how to code and get the work done. Ask around and you may find that this answer varies as well. My suggestion for this

question is to find out what skills are of interest and then seek and find the answer to a more specific question like, should a new programmer use SAS/GRAPH® or SAS/INSIGHT® to create code? There is too much to learn in one day and who really knows what SAS/INSIGHT is anyway? (I'd like to see a few more papers on this topic personally.)

HOW DO I FIND THE "BEST" CODE EXAMPLES?

The "best" code examples are found from other users. If you are trying to code something from scratch, chances are that someone has already coded it or something similar. There are a variety of internet search engines that provide links to numerous sites. SAS Online Docs has some good examples to follow as well as some SAS-L conversations. (Search SAS-L online, if not familiar with already)

Check out the RECOMMENDED READING section for some site examples.

MAKING SAS LIFE EASIER

So, now you have some idea about using SAS and you want to make things easier.
Why should I comment? What to do with my log and should I save it?

COMMENTS

Commenting is a SAS programmer's best friend. Actually, it's a programmer's best friend. If a SAS program is not properly commented and is poorly documented, there is little chance for a new programmer (and sometimes experienced one) to figure out the purpose of the code and how it should run. A lot of programmers will say that **not** commenting is called *job security*, but I think it's more of a concern of not making time or effort to add comments. I find that even if my own code is not commented properly and I need to modify it later, it is MORE time-consuming to try and trace back what the code is actually doing then to add comments in the first place.

SAVE A LOG

Who saves logs? I don't. Well, it's now become a common practice of mine to save a log where feasible.
There are a few reasons why I save a log.

1) Documentation – It helps to be able to go back and review previous jobs or projects to see how long something may have run or how many records were on a file and how long it took to read the file. Logs provide lots of details that output may not and saves time for future projects.

2) Organization – It helps to keep track of progress on various projects. Have you ever left work on a Friday and come back Monday and forgotten what step was last run? Ok, how about taken a vacation for a week or a long holiday and was not able to finish those last steps because your PC crashed at 4:30p Friday and it was all you could do to save your log by 5p. In theory, these things do not happen, but in reality – they do!

3) Time-Saver – Instead of trying to cut and paste the log and output various pieces of information, saving a log in SAS can easily be opened in a Text Editor and is much more efficient to search and find the information needed than trying to use SAS Log and Output windows to find, cut and paste into whatever document needs the information.

If you have a program that is automated, the log can be auto-saved as well just by adding a few lines of code.

HELPFUL SOLUTIONS

Many solutions exist that may not be advertised because they are not elaborate. However, many solutions needed are very simple. Microsoft Office Excel is software that many use in the business world. SAS provides several tools, including the new SAS® Add-In for Microsoft Office software to connect their software to others. Simple frequencies can be transformed into usable data and moved to a software program that others understand.
How can I get this report into Excel?

PROC FREQ USING LIST OUT=

The following code creates a dataset from the frequency and includes missing values, if any exist.

```
PROC FREQ DATA=TIPS.FINAL_DATASET;
    TABLES STATE*TEST_ID / LIST MISSING OUT=TIPS.TEST_RPT;
RUN;
```

PROC EXPORT

PROC EXPORT and IMPORT are used to bring data into and out of the SAS environment.

Which one of the following pieces of code looks correct?

```
PROC EXPORT DATA = TIPS.TEST_RPT
    OUTFILE= "C:\SESUG\SIMPLE BUT CLEVER\TEST_ID.XLS"
    DBMS=EXCEL REPLACE;
    SHEET="TEST_RPT";
RUN;
```

```
PROC EXPORT DATA = TIPS.TEST_RPT;
    OUTFILE= "C:\SESUG\SIMPLE BUT CLEVER\TEST_ID.XLS"
    DBMS=EXCEL REPLACE;
    SHEET="TEST_RPT";
RUN;
```

It would seem to make sense that the second piece of code would be correct, but it's not. The semi-colon after the dataset name will cause an error. The following error message is received and is very deceiving.

ERROR: FILE or TABLE= is required and must be specified.

EFFICIENT CODE, ERRORS AND DEBUGGING

One thing I personally strive to do is write efficient code. I don't like code that has a macro created and then runs inside of a Data Step, or has the Run statement on the same line as the IF then ELSE. For those who remember the TV game show "Name That Tune", I like to try to "Code that Code" in 10 lines or less, if possible. Who needs to write code that takes up a whole page when it can be done in one Data Step or within one or two PROC statements?

KEEP STATEMENT

The KEEP statement is a truly helpful statement and can be applied to most anywhere, including PROC FREQ, Data Step, & SET statements.

```
DATA TIPS.FINAL_TEST_ID;
    SET TIPS.FINAL_DATASET (KEEP= TEST_ID);
RUN;
```

WHERE CLAUSE/STATEMENT

The WHERE statement is flexible in more than one location of code, the PROC FREQ statement has different placements shown below. The first is a dataset option where the second is a statement of its own. Both work fine.

```
PROC FREQ DATA=TIPS.FINAL_DATASET (WHERE = (TEST_ID = 1234));
    TABLES STATE*TEST_ID / LIST MISSING OUT=TIPS.TEST_ID;
RUN;

PROC FREQ DATA=TIPS.FINAL_DATASET;
    WHERE TEST_ID = 1234;
    TABLES STATE*TEST_ID / LIST MISSING OUT=TIPS.TEST_ID;
RUN;
```

COMBINING STATEMENTS

Why not combine several dataset options? As many as are allowed can be used as long as the expected results are received. The code should always be checked against the data to make sure no errors were made in the code.

Which one of the below statements is correct?

```
PROC FREQ DATA=TIPS.FINAL_DATASET ((KEEP=STATE TEST_ID WHERE = (TEST_ID = 1234)) ;
    TABLES STATE*TEST_ID / LIST MISSING OUT=TIPS.TEST_ID;
RUN;

PROC FREQ DATA=TIPS.FINAL_DATASET ((KEEP=STATE WHERE = (TEST_ID = 1234)) ;
    TABLES STATE / LIST MISSING OUT=TIPS.TEST_ID;
RUN;
```

Well, it may LOOK like they both will work, but the second will NOT because the KEEP statement does not include the variable TEST_ID. Because it's used within the WHERE dataset option, it still needs to be kept.

Of course, the error message again is not so clear and implies the variable does not exist on the dataset.

ERROR: Variable id is not on the file TIPS.FINAL_DATASET

PROC FORMAT MATCH MERGE

PROC FORMAT is another useful tool that can be used not only for creating formats, but for matching. Check out several papers written about this subject. (I don't want to steal any thunder, as this is a HUGE timesaver for match/merging larger datasets.) One thing to look out for when using PROC FORMAT to match/merge is if the key is numeric or character. You can create a key anyway you like, but if the file you are matching is not the correct format; you will receive the following intuitive message.

ERROR: The format $KEY was not found or could not be loaded.

You will be driving yourself crazy knowing that you just assigned a format so why can't it be found? Make sure you assign the correct type to your match key variable. Also, make sure the key is not missing, and remove duplicates from key file.

FINDING VARIABLES ON FILES

Another ERROR message I found very hard to decipher. Usually this error occurs not because the variable is *not* "referenced" but because it *does not exist on the file*. As the error message includes DROP, KEEP, RENAME – these are the typical locations where a variable is listed, but not found.

ERROR: The variable ID_NUMBER in the DROP, KEEP, or RENAME list has never been referenced

CONTAINS

If you don't want to write out several pieces of code to SUBSTR or find length of something, start position, end position and then search the string – try the CONTAINS clause. However, be cautious when using – results can vary depending on use.

What is the difference in the following two pieces of code?

```
%LET TEST_LIST = 1, 4, 5, 7;

DATA TIPS.NAMES_LIST;
    SET TIPS.FINAL_DATASET (
    WHERE = (TEST_ID IN (&TEST_LIST.)AND ACCOUNT_ID NE. AND
    (NAME CONTAINS 'JOHN' AND 'MIKE')));
RUN;
```

```
PROC SQL;
    CREATE TABLE TIPS.NAMES_LIST AS
    SELECT *
    FROM TPS.FINAL_DATASET
    WHERE TEST_ID IN (&TEST_LIST.)AND ACCOUNT_ID NE. AND
    (NAME CONTAINS 'JOHN' AND NAME CONTAINS 'MIKE')
;
QUIT;
```

1st difference – one uses Data Step, one uses PROC SQL
2nd difference – "MIKE" is ignored in the Data Step. The code runs without a problem, but the CONTAINS clause does not look for "MIKE" within the NAME variable.

RENAME STATEMENT

The rename statement is especially useful when matching two datasets that have different names variables that need to be matched. A few helpful tips:

The previous or "OLDNAME" is always on the left side of the statement.

```
RENAME STATEMENT (OLDNAME=NEWNAME)
```

Where can RENAME statement be used?

- PROC FREQ
- DATA Step (in-stream or option)
- SET statement
- PROC DATASETS
- COMBINATION OF OPTIONS (WHERE, KEEP, DROP)

RUNNING OUT OF ROOM

If there are lots of datasets created within a program, you may be able to delete or drop some of the data not needed to save some work and memory space. Memory doesn't seem to be an issue within some work environments, but there are others than are always running out of space either on their own PCs or different platform environments. Datasets even in work space should be deleted if not needed once the final dataset is created.

AUTOEXEC.SAS

This file is initiated at the time of SAS start up. I like to put my passwords to the mainframe and user id for different platforms within this file. I never have to code it again later in any of my personal programs. It also hides anything within the file from the log, if you have the log saved for others to view.

RESOURCES

Where to find resources is as easy as finding the code that comes with it. The internet now has lots of postings from previous conferences and some conferences have many give-away CDs from prior proceedings. You can always find the presenter who wrote a particular paper and contact them directly for a copy of their paper. They may even send you their presentation (some presenters will convert it to PDF for public use.) I would even encourage trying to meet up with some of the presenters at a conference. They usually LOVE to talk about their experiences and why they wrote their particular paper. Some stories are as fascinating and interesting as the presenter. Attending local conferences and meetings is another way to meet up with some new faces and gain a network resource.

CONCLUSION

In summary, there are many options to help a programmer increase efficiency, write cleaner code, or make SAS life easier. But, the best solution to any problem is to research the answer before spending time jumping into the code. Networking is always a good idea if researching is too time consuming. Most times, another programmer has already solved the same problem and written a paper about it. If you happen to be acquainted with any programmers who use similar tools, usually an answer is just awaiting a question!

REFERENCES

SAS Online Docs V9.1.3
www.sas.com
http://en.wikipedia.org/wiki/Syntax

ACKNOWLEDGMENTS

I would like to thank and acknowledge all of my contacts and networks I have met since SAS became part of my technical background. Many friends have been acquired through networking and usually by means of presentations and code Q & A.

Also, thanks to the conference chairs, section chairs and all of the volunteers who make the conferences possible. (It's very easy to share ideas when a forum and the resources are provided.)

RECOMMENDED READING

The following sites were found during the creation of this paper and provide more details about discussion points. (All sites current at time of paper creation.)

http://www.caspur.it/risorse/softappl/doc/sas_docs/conn/signon.htm
http://www.caspur.it/risorse/softappl/doc/sas_docs/os390/zlibname.htm
http://analytics.ncsu.edu/?page_id=210 (SESUG Proceedings 2007)
http://www.lexjansen.com (Paper Proceedings)
http://www.sconsig.com/ (lots of links to numerous other SAS sites)

CONTACT INFORMATION

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

Claudine Lougee
Dualenic, LLC
PO Box 1108
Glen Allen, VA 23060
Phone: 724-467-0559
Email: claudine@dualenic.com
Web: http:/www.dualenic.com