Paper 4232 -2020

Tricks of the Trade: Streamlining and Enhancing your SAS® Sessions

Debra A. Goldman, Memorial Sloan Kettering Cancer Center

ABSTRACT

We are all here today because we have chosen SAS®; coding in SAS is likely a part of daily life, whether as programmers, database managers, or statisticians. Some of us with options at our companies have chosen SAS over other coding languages, such as R or Python. SAS is incredibly powerful and flexible; its PROCedures are validated and its code tested. It provides an impressive amount of options within PROCedures, and by default, the output contains all the relevant information for model fitting, statistical tests, and parameter estimates.

Unfortunately, SAS, by design, requires more code to be written to perform the same operation compared to other languages and much of the output is valuable only for review and not necessarily for reporting. We live and die by the semi-colon, run statements, and procedure calls. From my R user colleagues, the sheer differential in writing is enough to turn them off. However, coding in SAS need not be so cumbersome and reporting so tiresome. Features exist to make our code easier to read and to reduce the amount of code we write. Today, I'm going to cover a range of streamlining methods in SAS, such as including abbreviations, shortcut key macros, and using macros with ODS output to combine and present tables. This talk is intended for SAS Enhanced Editor users who want to have easier navigation, cleaner code, and streamlined output.

INTRODUCTION

The underlying concepts of efficiency presented here are about streamlined visualization, shorter/heuristic syntax, and reducing repetitiveness. My goal is for readers to walk away with at least one tip that resonates and can be implemented to make their SAS sessions more efficient.

I'll start with setting up SAS and the enhanced editor. These first tips require that users are allowed to make changes to the underlying SAS user profile. The second section deals more with general process for how to write code more efficiently in SAS.

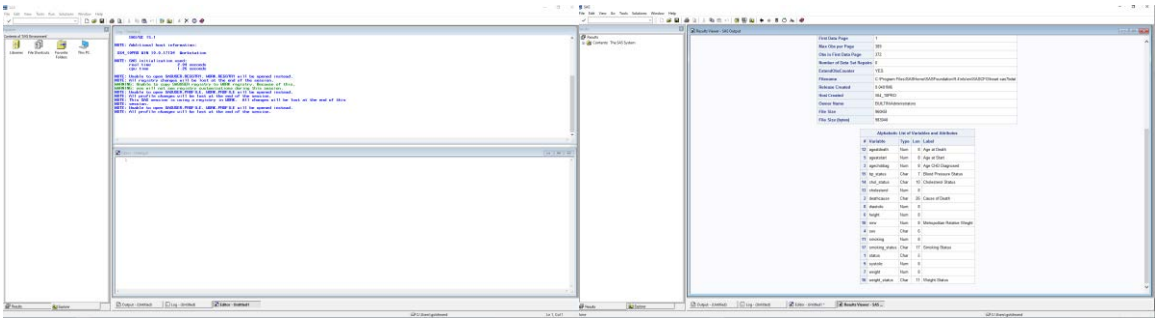
This paper will demonstrate how to:

- Alter window sizes
- Use abbreviations
- Create keyboard macros
- Write a basic macro for a repeatedly used procedures

The relevant talk at Global Forum 2020 will go into these topics more in depth. This paper is intended for those who use SAS 9.4 Enhanced Editor.

STARTING YOUR SAS SESSIONS: WSAVE

With default standard setup, SAS opens with the following window:

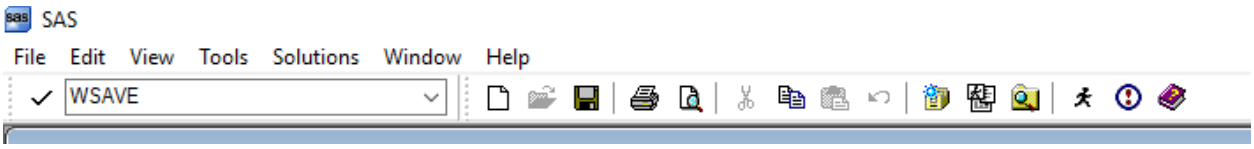


Display 1. Standard SAS Display

When we run our code, the output replaces the log and the editor window, and we have to flip back and forth, particularly if we've made errors, between these views. This type of process likely made sense in the era of running a single procedure, but if your work is anything like mine, I may be calling a procedure hundreds of times before needing to look at the output. One doesn't need to have a PhD in cognitive processing to realize how inefficient this is. One should be able to see the relevant code, log, and output simultaneously. However, if you have to spend every session resizing windows, or resizing the output every time you clear the viewer, it's not necessarily more efficient. That's where WSAVE comes in.

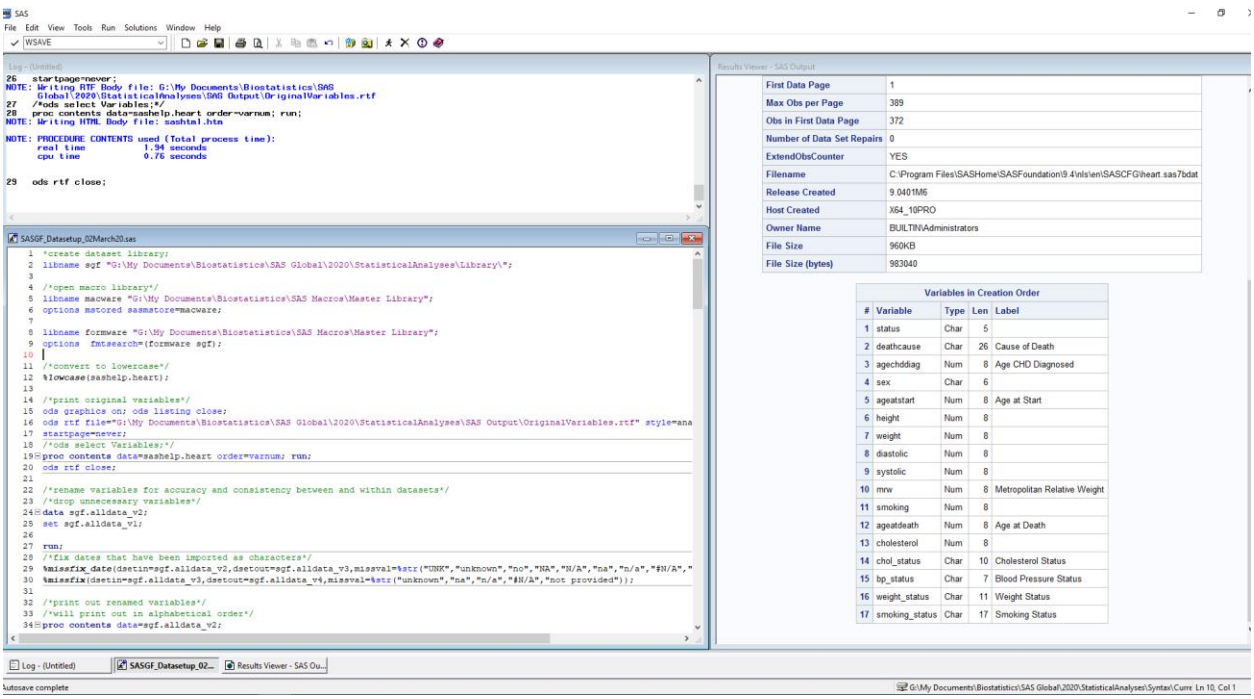
The WSAVE command can be put into the command prompt as follows for the log, HTML output windows. Here are the steps:

1) Resize the relevant window to the desired size and stay on that window
2) Enter WSAVE into the command prompt
3) Hit enter
4) Repeat for log and syntax



Display 2. Where to Insert WSAVE Command

After this is done, the next SAS session will open with the windows saved to the specified sizes. The only caveat here is that the explorer, results, and 1994 style output window will still open, and if you don't want to see these, you have to X out of these (or use CTRL+F4) every time it opens. It would be great if SAS allowed us to disable certain output at start up. Here is what my SAS looks like on typical start up:



First Data Page	1
Max Obs per Page	389
Obs in First Data Page	372
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Program Files\SASHome\SASFoundation\9.4\nls\en\SASCFG\heart.sas7bdat
Release Created	9.0401M6
Host Created	X64_10PRO
Owner Name	BUILTIN\Administrators
File Size	960KB
File Size (bytes)	983040

Variables in Creation Order

#	Variable	Type	Len	Label
1	status	Char	5	
2	deathcause	Char	26	Cause of Death
3	agechddiag	Num	8	Age CHD Diagnosed
4	sex	Char	6	
5	ageatstart	Num	8	Age at Start
6	height	Num	8	
7	weight	Num	8	
8	diastolic	Num	8	
9	systolic	Num	8	
10	mrw	Num	8	Metropolitan Relative Weight
11	smoking	Num	8	
12	ageatdeath	Num	8	Age at Death
13	cholesterol	Num	8	
14	chol_status	Char	10	Cholesterol Status
15	bp_status	Char	7	Blood Pressure Status
16	weight_status	Char	11	Weight Status
17	smoking_status	Char	17	Smoking Status

Display 3. Alternative SAS View Setup

As you can see, the log is up top with a larger space for syntax underneath, and the results filling up the full length and about 40% of the width. Feel free to play around with this feature and figure out a setup that works best for you.

SMART SEARCHING AND TYPING

This section will focus on how to set up abbreviations and keyboard macros. I'll provide an example of how I've used abbreviations for more complicated text.

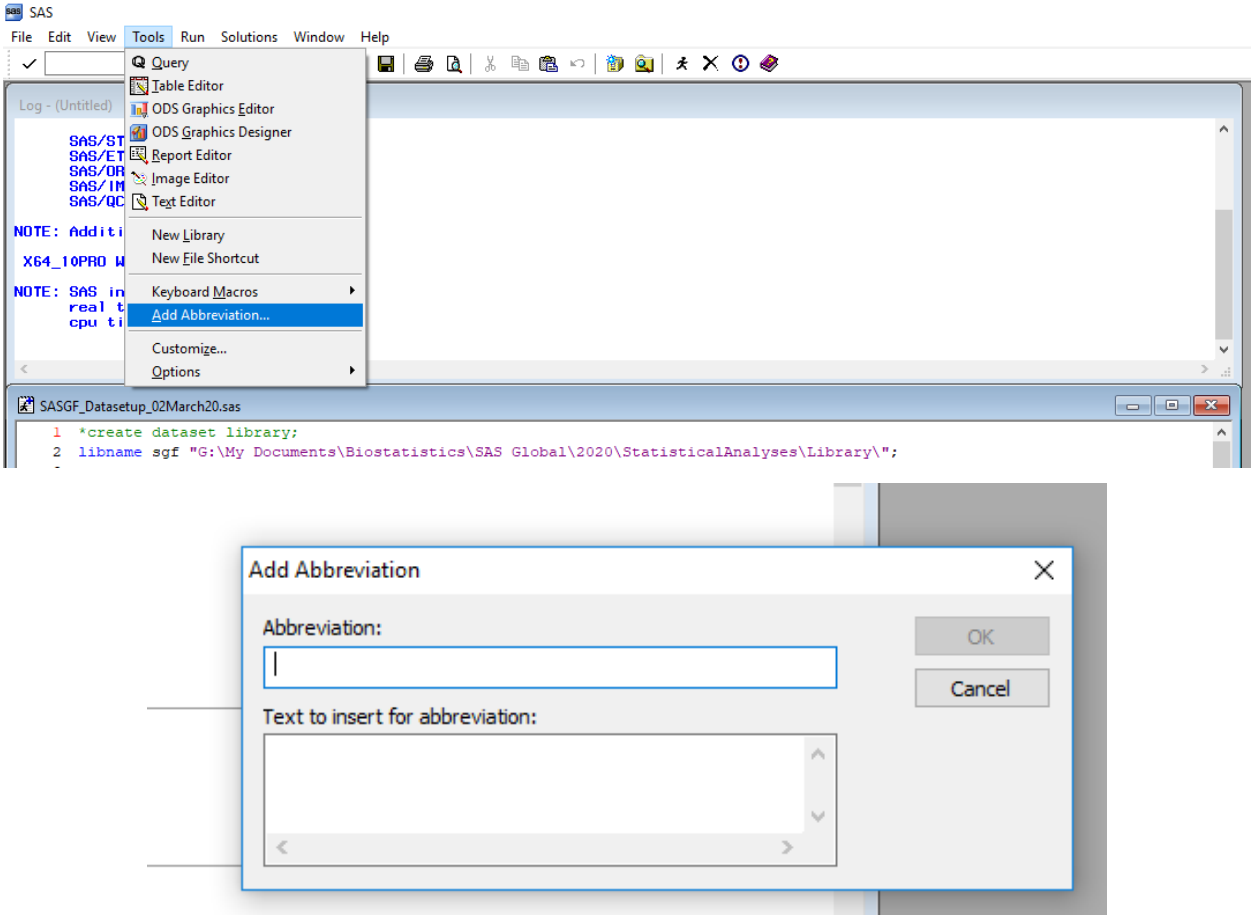
ABBREVIATIONS

Abbreviations in SAS work as abbreviations do in speech; they stand in for a much longer piece of syntax. You can think of these abbreviations as your LOL, ONW, and WFH of your SAS sessions.

You can use abbreviations for any syntax including:

- Frequently used tables in PROC REPORT
- Standard options /customizations for models that one is prone to forgetting
- Macro factors
- Complicated ODS text statements

To add an abbreviation, go to tools→ add abbreviation.



Display 4. Where to Find Abbreviations

The abbreviation goes into the first box with the text you want to appear in the second box. My advice is to use words or phrases that are easy to remember, but are not standard SAS syntax phrases. For instance, the example below uses "table 1" to stand for my standard PROC REPORT for descriptive characteristics. If I used "REPORT," every time I typed out "PROC REPORT," which I'd use for every other kind of table, I would end up with the table 1 text. Also, since I use "table1" as the name for my macro to add data to this table, I'm putting a space here to avoid the PROC REPORT table 1 text appearing every time I call the table 1 macro.

```
28  %table1(data=sashelp.heart,rowvar=ageatstart,rowvart
29
30  table 1
31  proc report data=work.table_pati...(Abbrev)
32
33
34  %table1(data=sgf.alldata_final,rowvar=age_atdx,rowva
```

Display 5. Example of an Abbreviation

```
proc report data=work.table_patientchar nowindows headline  missing spanrows;
 column rowvar_name ordering rowvar_value combo dummyvar ;
 define rowvar_name/group order=data '' format=$desc_variablef. ;
 define ordering / group order=internal "" noprint;
 define rowvar_value/group order=data '' style=[textalign=left];
 define combo/display 'N (%)' style=[textalign=center cellwidth=1.5in background=$combof.];
 define dummyvar/computed noprint ;
 compute dummyvar;
 dummyvar=1;
 endcomp;
 where combo ne " " and rowvar_value not in ("N Total");
 compute rowvar_name;
  if rowvar_name = "sample_size" then call define (_row_,'style','style=[borderbottomwidth=1 borderbottomcolor=black
                                                   borderbottomstyle=dashed');
 endcomp;
 run;
```

Display 6. Syntax that Appears for Abbreviation "table 1"

KEYBOARD MACROS

Keyboard macros are similar to simple operations, such as copy (CTRL+C) and paste (CTRL+V), but they can perform a variety of tasks. For those that have iPhones, this type of operation is also available with iOS 13. Clearly, SAS isn't the only one that realizes the benefits of automating tasks.

I mostly use keyboard macros as navigational tools, but effectively, you can use these to combine any series of operations. I organize my syntax files with PROC for statistical models followed by a report of the summarized output. However, if a project has many, many models, it's not so easy to find the start or the end of the report. That's where keyboard macros come in. At the beginning of an ODS output report, I add the following syntax:

```
/*###################################*/
/*START REPORT*/
/*###################################*/
```

At the end of an ODS output report, I add:

```
/*###################################*/
/*END REPORT*/
/*###################################*/
```

I recorded a macro that uses the search function (CTRL + F) to find the phrase, /*START REPORT*/, and set it to the shortcut key CTRL + R. When I type CTRL +R, it moves to the beginning of the ODS report. Similarly, I set CTRL + E to find the end of the report. As with abbreviations, I recommend using a shortcut that is related enough to remember and doesn't stand in for something else. For instance, I could change CTRL + S to find the start of the report, but because CTRL + S almost always stands for save in windows, this would be harder to remember.

OTHER SHORTCUT TIPS

DON'T START FROM SCRATCH: KEEP A STANDARD SYNTAX FILE

If your job involves you working on many projects simultaneously, it can be hard to keep track of when and where you once added that one thing to a graph that clients really liked. It's helpful to keep a running syntax file for special modifications that you may need to use again in the future. I start all SAS projects with a standard folder organization, which contains the standard syntax file.

BOOKMARKING WITH CTRL+F2

CTRL+F2 adds a marker in the left margin in the current syntax file and F2 toggles in between bookmarked locations. The markers appear as light blue rectangles. Using this technique is incredibly helpful when the PROC syntax is quite far away from the reporting syntax, or if one needs to move in between a PROC TEMPLATE graph and the PROC SGRENDER call. Unfortunately, these bookmarks disappear when the current SAS session ends, so they are more so meant for temporary troubleshooting rather than permanent navigation.

```
27  %create_table1_blank(tablename=table_patientchar);
28  %table1(data=sashelp.heart,rowvar=ageatstart,rowvartype=2,rowvarformat=5.0,tablename=table_patientchar)
29  /*...*/
30  /*HUNDREDS OF LINES OF SYNTAX IN BETWEEN*/
31  /*...*/
32  proc report data=work.table_patientchar nowindows headline  missing spanrows;
33  column rowvar_name ordering rowvar_value combo dummyvar ;
34  define rowvar_name/group order=data '' format=$desc_variablef. ;
35  define ordering / group order=internal "" noprint;
36  define rowvar_value/group order=data '' style=[textalign=left];
37  define combo/display 'N (%)' style=[textalign=center cellwidth=1.5in background=$combof.];
38  define dummyvar/computed noprint ;
39  compute dummyvar;
40  dummyvar=1;
41  endcomp;
42  where combo ne " " and rowvar_value not in ("N Total");
43  compute rowvar_name;
44   if rowvar_name = "sample_size" then call define (_row_,'style','style=[borderbottomwidth=1 borderbottomcolor=black
45                                                    borderbottomstyle=dashed');
46  endcomp;
47  run;
```

Display 7. Example of Bookmarks in Lefthand Margin

MACROS: THE BREAD AND BUTTER OF EFFICIENCY

Macros are the most efficient use of space that SAS offers and will save an incredible amount of typing time. The key to creating efficient and widely useful macros is to figure out the following information:

The way in which I go about creating a macro is to figure out:

1. What changes between each iteration of the procedure?
2. What needs to be stored from the procedure?
3. How should the data be stored?
4. How should the data be displayed?

The beauty of macros is that you can modify them as the use of the macro or procedure grows. Many others have written about macros[1,2] and SAS frequently hosts classes on the subject, so the example here is more to illustrate how efficient the code writing is once written. Please see the references and recommended reading for more step by step tools.

The example here uses the SASHELP dataset, BWEIGHT, which contains infant birthweight data from the National Center for Health Statistics in 1997. The macro below will take the data out of PROC CORR, adds the output to a central dataset, and outputs to proc report when finished.

CREATING A CENTRAL MACRO LIBRARY

Macros are stored in libraries like any other SAS object. While they can be stored locally within each project, creating a central repository will allow one to call the same macro for different projects.

The macro library is created with the following call:

```
libname macware "G:\My Documents\Biostatistics\SAS Macros\Master Library";
```

And the option statement loads the stored macro library:

```
options mstored sasmstore=macware;
```

To store a macro, simply add store and source as options after the macro statement:

```
%macro spearman_corr(...)/store source;
```

When you run the macro, it will save into the repository called in sasmstore.

CREATING THE MACRO

Question 1: What changes need to be made between each iteration of the macro?

For the example, we'll use spearman's correlation with a simple data situation. The dataset used will utilize all variables with a single factor. Using PROC SQL, I've taken all other variables and loaded into a central macro variable. However, the macro that will presented here will work for a list or a single variable.

```
proc sql ;
select name
into  :corrlist separated by ' '
from dictionary.columns
where memname="BWEIGHT" and libname="SASHELP"
 and upcase(name) not in ("WEIGHT");
quit;
```

```
proc corr data = sashelp.bweight spearman fisher(biasadj=no);
var &corrlist;
with weight;
run;
```

For the initial build of the correlation macro, four parameters could change with each iteration

1. The dataset
2. The variable or list of variables
3. The variable we'll want to check the correlation with
4. Bias adjustment

These should be the four starting parameters of the macro. Here is what the base macro looks like with those four parameters:

```
%macro spearman_corr(data=,var=,withvar=,biasadj=)
      proc corr data = &data spearman fisher(biasadj=&biasadj);
      var &var;
      with &withvar;
      run;
%mend;
```

When it comes to naming parameters, I recommend using words that are not challenging to decode. It's easy to remember that data stands for the dataset as we all have to write data= for almost every SAS procedure. Using dset or dt or dat are fine as well, but I find it easier to remember when I align the name close to the related syntax.

Question 2: What needs to be stored from the PROCedure?

For any SAS procedure, SAS creates temporary datasets of the output.[3] Each table or figure in SAS has a corresponding dataset. These datasets can be viewed by using `ods trace on` prior to the PROC statement or by looking in the results window. From **PROC CORR**, the data I want is from the spearman correlation matrix. To pull this data out, the following statement needs to be added above the **PROC CORR** call or within the **PROC**:

```
%macro spearman_corr(data=,varlist=,withvar=,biasadj=)
ods output FisherSpearmanCorr= spearman_v1;
proc corr data = sashelp.bweight spearman fisher(biasadj=&biasadj);
var &corrlist;
with weight;
run;
%mend;
```

One can view the data using **PROC PRINT** and see details of the variables using **PROC CONTENTS**. Below is the code I'm using to modify the underlying dataset. To note, this step can be skipped if you're okay with how the data looks. One reason I tend to modify is to make variable naming consistent within procedure and between procedures. Not all SAS procedures use the term "var"; some will use "variable," some will use "parameter". Therefore, if I call them all var_name, it is easier to remember should I want to call variables from this table later or combine with other procedures. I've also created a combined variable for easier display. The data can be displayed without this combination, but when I use in-line text, it is simpler to call a combination variable rather than three separate variables.

```
data spearman_v2;
set spearman_v1;

rename var = var_name;
rename withvar = withvar_name;
rename nobs = freq;
rename corr = spearman;
rename lcl = spearman_lcl;
rename ucl = spearman_ucl;
spearman_combo = trim(left(put(corr,4.2))) || " (" ||
trim(left(put(lcl,4.2))) || "-" || trim(left(put(ucl,4.2))) || ")";

keep var withvar nobs corr lcl ucl spearman_combo pvalue;
run;
```

Question 3: How should the data be stored?

The data can exist and be called from its original table. However, if I had additional with variables, or if I called each variable in the dataset separately, I would overwrite the modified dataset. As a result, it's helpful to create a central repository with the underlying structure of the dataset to report on. Further, by creating a central repository, one prevents truncation of character variables due to an earlier variable being of a shorter length. Below is the code for this central, blank dataset.

```
%macro create_spearman_table(tablename);
%IF %length(&tablename)>0   %THEN %DO;
      data &tablename;
      length var_name $ 40;
      length withvar_name $ 40;
      length spearman_combo $ 50;
      var_name = " ";
      withvar_name = " ";
      spearman = .;
```

```
      spearman_lcl = . ;
      spearman_ucl = . ;
      spearman_combo = " ";
      pvalue = . ;
      freq=.;
      dataset=" ";
      run;
%END;
%ELSE %DO;
      data spearman_combined;
      length var_name $ 40;
      length withvar_name $ 40;
      length combo $ 50;
      length spearman_combo $ 50;
      var_name = " ";
      withvar_name = " ";
      spearman = .;
      spearman_lcl = . ;
      spearman_ucl = . ;
      spearman_combo = " ";
      pvalue = . ;
      freq=.;
      run;
%END;
%mend;
```

You may have noticed that I provided an optional positional parameter within this call. One additional aspect to storing the data is whether you want unique table names or a generic. I've also now added this tablename parameter as a keyword parameter to the spearman correlation macro.

To add to the central dataset, all one needs to do is include a PROC APPEND within the macro.

```
      %if %length(&tablename)>0 %then %do;
            proc append base = &tablename data = spearman_v2 force nowarn;
            run;
      %end;
      %else %do;
            proc append base = spearman_combined data = spearman_v2 force
            nowarn; run;
      %end;
```

Question 4: How should the data be displayed?

The data need not be displayed within the macro, but providing the option shortens the syntax that needs to be written. Below is the code for the PROC REPORT call within the macro. I've also added a parameter to the macro called view that displays the table if view =1. In the talk, I will touch further on how to format more complex tables

```
%if &view = 1 %then %do;
      %if %length(&tablename)>0 %then %do;
            proc report data=&tablename nowindows headline missing spanrows;
            column withvar_name var_name  (spearman_combo pvalue ) ;
            define withvar_name /group '' order=formatted  ;
            define var_name /group order=data 'Factor';
            define spearman_combo/"Spearman's Rho (95% CI)" display
            style=[textalign=center];
            define pvalue/'p-value' analysis format=pval_32p.
            style=[fontweight=pval_fw.];
```

```
            where spearman ne .;
            run;
            %end;
            %else %do;
            proc report data=spearman_combined nowindows headline missing
            spanrows;
            column withvar_name var_name  (spearman_combo pvalue ) ;
            define withvar_name /group '' order=formatted  ;
            define var_name /group order=data 'Factor';
            define spearman_combo/"Spearman's Rho (95% CI)" display
            style=[textalign=center];
            define pvalue/'p-value' analysis format=pval_32p.
            style=[fontweight=pval_fw.];
            where spearman ne .;
            run;
            %end;
      %end;
```

RUNNING THE MACRO

Instead of all the syntax above, below is the only text that's needed to run the macro and view the results in the table.

```
%create_spearman_table(spearman_bweight_combined);
%spearman_corr(data=sashelp.bweight,var=&corrlist,withvar=weight,biasadj=no,
               tablename=spearman_bweight_combined,view=1);
```

The SAS System

	Factor	Spearman's Rho (95% CI)	p-value
Weight	Black	-.16 (-.17 - -.15)	**<.001**
	Married	0.16 (0.15 - 0.17)	**<.001**
	Boy	0.12 (0.11 - 0.13)	**<.001**
	MomAge	0.12 (0.11 - 0.12)	**<.001**
	MomSmoke	-.15 (-.16 - -.14)	**<.001**
	CigsPerDay	-.15 (-.16 - -.14)	**<.001**
	MomWtGain	0.20 (0.19 - 0.21)	**<.001**
	Visit	0.07 (0.06 - 0.08)	**<.001**
	MomEdLevel	0.01 (-.00 - 0.02)	0.11

Output 1. Results from *Spearman_corr* Macro Call

Now, anytime I want to create a table for a series of spearman correlations, I simply need to load my macro library and run the above macro. As this example illustrates, the amount of code writing and details to remember is astoundingly shorter. It not only reduces the amount of syntax, but it's also generally easier to follow and read. Although my macros are not streamlined for public use, I've made some macros available on my github (link at end of the paper). Feel free to browse these for inspiration, or take and modify for personal use.

CONCLUSION

The tips and tricks in this paper establish a foundation for which to build a more efficient SAS session. As one advances with these techniques, each session and project should become more efficient and user friendly. Hopefully, with future releases of SAS, features will be added that further enhance efficient. The related global forum talk with cover these topics more in depth and also touch on dynamic variables within PROC TEMPLATE.

REFERENCES

1. Coleman, Ron. 2019, "SAS® Macros: Beyond the Basics". *Proceedings of the SAS Global 2019 Conference.* Dallas, Tx: The SAS Institute. Available at: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3511-2019.pdf.
2. Lafler, Kirk Paul. "Hands-On SAS® Macro Programming Essentials for New Users". Proceedings of the SAS Global 2019 Conference. Dallas, Tx: The SAS Institute. Available at: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3184-2019.pdf
3. Goldman, D. A. 2017. "Creating Complete Automated Reports in SAS using the ODS RTF Destination: Protect your Results from Human Error". Proceedings of the Western Users of SAS Software Conference 2017. Long Beach, CA. SAS Institute Inc., Available at: https://www.lexjansen.com/wuss/2017/32_Final_Paper_PDF.pdf

ACKNOWLEDGMENTS

Thank you to the Global Forum Academic Committee for inviting me to give this talk.

RECOMMENDED READING

Carpenter, Art. 2016. *Carpenter's Complete Guide to the SAS Macro Language, Third Edition*. Cary, NC. The SAS Institute.

Burlew Michele M. 2014. *SAS Macro Programming Made Easy, Third Edition.* Cary, NC. The SAS Institute.

CONTACT INFORMATION

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

Debra A. Goldman, MS
Memorial Sloan Kettering Cancer Center
646-888-8331
goldmand@mskcc.org
https://github.com/DGStat/
https://www.mskcc.org/profile/debra-goldman