Paper 272-2011

Tips and Techniques for the SAS® Programmer

Helen Carey, Carey Consulting, Kaneohe, HI
Ginger Carey, Carey Consulting, Kaneohe, HI

ABSTRACT

If you are new to programming or even are an experienced programmer, you will benefit by learning tips and techniques that will help you be more productive. This presentation offers tips in programming, efficiency, work habits, and where to find answers to your SAS questions. These tips come from our own experience and from learning from others through their presentations and papers.

INTRODUCTION

While helping many SAS users at a university, we have found them writing programs the hard way. If they had only learned about dates or functions or this or that procedure or learned how to use the available resources, they could have saved themselves a lot of time and frustration. However, when using SAS software, you quickly discover that there are many ways to achieve your results.

A search on the internet for *SAS tips*, *SAS resources* or *SAS efficiency* results in thousands of hits. Also there are SAS Press books that focus on SAS tips. Our tips range from simple timesavers to opinions on how to generate the results. Many of the tips are brief with a reference to a paper or a page on a website with in depth information about the topic. The topic is worth pursuing in greater detail than can be given in this paper. The *Recommended Reading* section at the end of this paper lists where to find the papers.

One of SAS Institute's registered trademarks is the slogan ***The Power to Know®.*** We are going to use the word POWER to reveal some fundamental tips. Ginger thought that the title of this paper should have been "The SAS Solution: The Power You Don't Know You Have." That sounds like a future paper.

These are the topics identified by the acrostic POWER.

P – Procedures
O – Output
W – Work Habits
E – Efficiency
R – Resources

The topic *data* is not part of the acrostic POWER. Because everything starts with the data, that is where we will start.

DATA

Know Your Data

Needing to know the data is something that we learned first-hand. We were analyzing and reporting on data collected from senior citizens. The analysis indicated that many were exercising more than 4 hours a day. That was not how I pictured retirement. The data was supposedly "clean" and it was. By running a PROC FREQ and PROC UNIVARIATE, we discovered many retirees enjoyed gardening for 4 to 8 hours daily. Gardening was considered an exercise. Another time we had to inform a researcher that his published results on a public website were incorrect because he did not understand how the missing values were coded.

Therefore, know your data. Check data values, range of values, frequencies of values, missing values, index variables with unique values, complete and consistent dates, required variables, and duplicate records.

A point and click way to check your data is to use SAS Enterprise Guide and the task *Characterize Data*. With a wizard, you can easily create summary report, graphs, frequency and univariate statistics.

For normally distributed data, you can use PROC SQL to select extreme values.

```
proc sql;
  select *, avg(score) as mean,
            std(score) as sd
  from mylib.scores
  group by gender
  having abs(mean-score)>2*sd;
run; quit;
```

One way to check your data is to use PROC FORMAT to define valid groups and run PROC FREQ on those groups.

```
proc format;
  value dosefmt
      0,.01-2='valid'
      .      ='missing'
      other  ='invalid';

proc freq data=trial;
   tables Dosage / missing;
   format Dosage   dosefmt.;
run;
```

Ron Cody's book *Cody's Data Cleaning Techniques* includes programs and macros for typical data cleaning tasks.

Use PROC DATASETS or PROC SQL with the view SASHELP.VCOLUMN discussed below to learn more about the variables and their attributes.

Data Step

If you are not familiar with the data vector, variable attributes, and how SAS processes the data while creating a SAS data set, then read *The Essence of DATA Step Programming* by Arthur Li. This paper explains what happens "behind the scenes." A good, basic understanding of how SAS processes and stores data is essential to being a good SAS programmer.

Along with data values, each SAS data set contains metadata or data about the data. This information, recorded in the descriptor portion of the data set, contains information like the names and attributes of all the variables, the number of observations in the data set, and the date and time that the data set was created and updated.

PROC DATASETS

To view the descriptor portion, you can right click on the data set in the SAS Explorer window and select **view columns** or print it with PROC DATASETS. The DETAILS option lists the number of observations, the number of variables, and the label of the data set. This is also a way to find typos of the variable names.

For example:

```
proc datasets library=work details;
     contents data=highschool;
run;
```

Michael Raithel's paper *PROC DATASETS; The Swiss Army Knife of SAS®* has everything you want to know about PROC DATASETS, a powerful procedure. If you are just changing the attributes of the variables, such as their names, informats and labels, then use PROC DATASETS to do the work for you, not the data step.

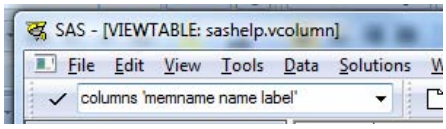
Use PROC DATASETS instead of the data step to concatenate SAS data sets. For example:

```
proc datasets library=youthlib;
     append base=allyears
     data=year1997;
run;
```

ViewTable Window

The ViewTable window in a SAS session, is an interactive way to view, enter and edit data. It is accessible from the SAS Explorer window by clicking on the data set or view or using the **viewtable** (abbreviated **vt**) command from the command box. The command box is below the menu bar.

Once you open the ViewTable window, you can select to view only specific columns by typing the columns command from the command box, such as `columns 'memname name label'`. This is the same as using the hide/unhide on the Data Menu of the ViewTable window.



Close the ViewTable window before submitting the program that recreates a SAS data set. More than once, I have not closed the window, have not read the log, and wondered why the results did not change. This is an example of where it is important to read the log after every run. By reading the log, I would have found out that I could not re-

create my data set because it was open in the ViewTable window and "*The SAS System stopped processing this step because of errors*." Reading the log after every run is a good practice to follow.

The Appendix contains information about opening a ViewTable window the way you want it.

Dictionary Tables

A DICTIONARY table is a read-only SAS view that contains information about SAS libraries, SAS data sets, SAS macros, and external files that are in use or available in the current SAS session. Each DICTIONARY table has an associated PROC SQL view in the SASHELP library. You can see the entire contents of a DICTIONARY table by opening its SASHELP view in the ViewTable window. These SAS views name starts with V, for example, VCOLUMN or VMEMBER. SASHELP.VCOLUMN was the view that we were using above in the ViewTable Window section.

Here is an example of accessing SASHELP.VCOLUMN using PROC SQL.

```
proc sql;
  select memname format=$8.,
         varnum,
         name format=$15.,
         label
    from sashelp.vcolumn
   where libname='SASHELP'
     and memname='ZIPCODE';
run; quit;
```

Results

Member Name	Column Number in Table	Column Name	Column Label
ZIPCODE	1	ZIP	The 5-digit ZIP Cod
ZIPCODE	2	Y	Latitude (degrees)
ZIPCODE	3	X	Longitude (degrees)
ZIPCODE	4	ZIP_CLASS	ZIP Code Classifica
			Blank=Standard/non-
ZIPCODE	5	CITY	Name of city/org
ZIPCODE	6	STATE	Two-digit number (F
ZIPCODE	7	STATECODE	Two-letter abbrev.
ZIPCODE	8	STATENAME	Full name of state/
ZIPCODE	9	COUNTY	FIPS county code.
ZIPCODE	10	COUNTYNM	Name of county/par

Functions

Have fun with functions. Use them to save programming time, make the code easier to read and possibly execute faster. There are hundreds of functions in categories ranging from arithmetic functions to variable information functions. For an in-depth look at SAS functions, read Cassidy's paper *Building a Good Foundation with Functions*. Numerous functions are added in every release of Base SAS software. Review the new functions in the enhancements documentation.

Here are some examples.

The IFC or IFN function may be more convenient than using an IF/ELSE statement. IFC returns a character value and IFN returns a numeric value. For example, instead of using

```
if results > 70 then grade = 'pass';
else grade = 'fail';
```

you can use this instead:

```
Grade  = ifc(results > 70,'pass','fail'); /* store character value */
```

To check for number of values that are missing, use the NMISS function

```
if nmiss(of q1-q20)>7 then delete;
```

Some functions allow a list of variables as arguments. Use of the keyword OF gives the user the flexibility to include variable lists, array elements and other shortcuts for referencing variable names. The following 4 examples using the MEAN function are finding the mean of the variables q1, q2, q3.

```
avg=mean(q1,q2,q3);
avg=mean(of q1 q2 q3);
avg=mean(of q1-q3);

array qarray(3) q1-q3;
Avg=mean(OF Qarray(*));
```

Use the fact that the comparison equal to (=) operator returns either the value 0 or 1. This example sums up the number of questions with the value 3.

```
Total=sum(q1=3, q2=3, q3=3, q4=3);
```

The %SYSFUNC , %QSYSFUNC macro functions can execute most SAS functions and return the results with an optional format. Here is an example to put the current date and time in a desired format by using the DATE and TIME functions in %SYSFUNC.

```
title  "%sysfunc(date(),worddate.)";
title2 "at %sysfunc(time(),time.) ";
```

Results: April 7, 2011
at 08:00:00

The IFN and IFC functions can also be used in %SYSFUNC. The macro variables minAge and maxAge are created in the following PROC SQL code and then used to create the footnote with the &SYSFUNC and IFC function. The footnote will have the value of either 'Ages OK' or 'Out of Range' depending on the range of age values. In the SQL code, the macro variables minAge and MaxAge are proceed by a colon (:).

```
proc sql;
   select min(age), max(age)
     into :minAge, :maxAge
     from sashelp.class;
run;
%put Min age is &minAge Max age is &maxAge;
Footnote1
"%sysfunc(ifc(&minAge>11 and &maxAge<17,
          'Ages OK','Out of Range'))";
```

To review the new and enhanced functions in SAS 9.2 Base software, see the *What's New in SAS 9.2* documentation on the SAS Support website. There are more than 100 new functions, such as CAT, CATS, CATT, CATX (trims and concatenates), PROPCASE (for proper case), CALL SYMPUTX (trims blanks before assigning the macro value), MEDIAN (get median) and PCTL (get percentiles), LARGEST, SMALLEST, ZIPCODE and ZIPDIST functions. Also in SAS 9.2, you can write your own functions in C, C++, or the SAS language. So check out *What's New in SAS 9.2*.

SAS Data Set Options

To create multiple SAS data sets, use multiple OUTPUT statements or the WHERE option.

```
data good check;
   set trans;
   if amount<0 then output check;
   else             output good;
run;
```

Use the WHERE option:

```
data good (where=(amt>=0))
     check(where=(amt< 0));
  set trans;
run;
```

Test with A Subset of Your Data

If the data are organized in some order, take a random sample. Use the WHERE option with the RANUNI function to create a test file with about 10% of records. The RANUNI function returns a random value from a uniform distribution.

```
data testdata;
  set mylib.proddata;
  where ranuni(729)<=.10;
run;
```

Macro Language

The SAS macro language is a powerful tool that writes SAS statements. It is extremely useful for repetitive tasks. The basics are relatively easy to learn and you can reap immediate benefits. To get started using the macro language, read the paper *SAS® Macro Programming for Beginners by* Susan Slaughter and Lora Delwiche. Get a sound understanding of how SAS works before delving into the power of the macro language. You can start small with

simple macro variable substitutions and slowly build your macro skills.

We all know it is a good idea to document our SAS code with comments. To document your macro code, use the macro comment (**%* … ;**). Better yet, use delimited comments (/* … */). Neither macro comments nor delimited comments (**/* … */**) are passed out of the macro, when it is resolved.

The statements in the macro are not run until the macro is called. You can use this to your advantage to skip statements that you no longer want to run. Surround the code with %macro and %mend statements. See our paper *Just Skip It* to learn various ways to skip the code.

At some point, you will start writing more complex macro code. When you start writing complex macro code with %do loops, %if's, multiple ampersands, and complicated quoted strings, you might want to see the generated SAS code without wading through all the log comments. You can do this by directing the code that is generated by the macro to a specified file by using the MFILE and MPRINT system options. The syntax is as follows:

```
filename mprint 'c:\hips\programs\testmac.sas'; *'pathname and name of file';
options mprint mfile;
%your_macro
```

The fileref must be MPRINT. The pathname must include the name of the external file where the code that is generated by the macro is to be stored. After this program finishes executing, you can take the code generated and run it to understand what is actually happening and fix any problems.

Another good paper on macros with references to other papers is Dalaney and Carpenter's paper Macro: *Symbols of Frustration? %Let us help! A Guide to Debugging Macros*.

The best way to end this topic on the macro language is with this quote from a respected SAS expert:

> *To be a complete SAS programmer one has to learn how to use macro, conquer its weaknesses and difficulties, not avoid macro altogether.*–Ian Whitlock

PROCEDURES

Do Not Over Code

DATA steps typically create or modify SAS data sets, but they can also be used to produce custom-designed reports. Instead of building reports in the data set, use the output data set capabilities of procedures. We took a report written in the data step that consisted of over 100 pages of code and re-wrote it in only 5 pages using macros, PROC TABULATE, PROC SUMMARY and user-defined formats. And it was easier to read and maintain.

Investigate using the UNIVARIATE, FREQ, RANK, SORT, SQL, SUMMARY, TABULATE, REPORT and TRANSPOSE procedures. By using PROC TRANSPOSE, you can rearrange a SAS data set so that observations become variables and variables become observations. Although you can do the same process with a data step that uses array processing, PROC TRANSPOSE requires minimal programming.

Never assume you know everything about a procedure. Review a procedure a couple of weeks after you first start using it to learn about useful features and capabilities.

Don't overlook learning about PROC REPORT and PROC TABULATE. Both are awesome and powerful procedures. Check out the SAS Press books on both procedures and the numerous conference papers presented throughout the years.

PROC FREQ

Most SAS programmers have used the FREQ procedure to evaluate basic data quality. Many valuable features of PROC FREQ are not used even by experienced SAS users. Before turning to more complex procedures, check out what is available in PROC FREQ.

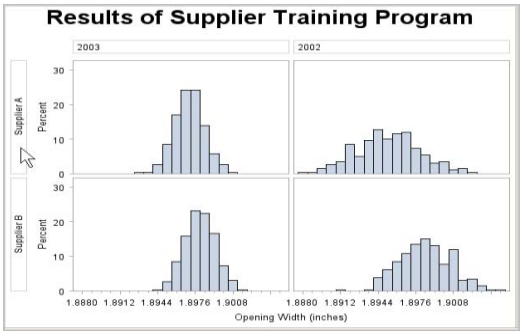
The FREQ procedure can now produce frequency plots, cumulative frequency plots, deviation plots, odds ratio plots, and kappa plots by using ODS Graphics.

PROC UNIVARIATE

This example is from the UNIVARIATE Procedure documentation of *Base SAS(R) 9.2 Procedures Guide: Statistical Procedures, Third Edition*. Search for "univariate Two-Way Comparative Histogram" to find the example on the SAS website.

This example shows how easy it is to graphically compare data from 2 suppliers for 2 years using PROC UNIVARIATE.

```
proc format ;
      value mytime  1 = '2002' 2 = '2003';
title 'Results of Supplier Training Program';
proc univariate data=disk noprint;
    class supplier year /
      keylevel = ('supplier a' '2003');
    histogram width / intertile = 1.0
       vaxis = 0 10 20 30
       ncols = 2 nrows = 2;
  run;
```



PROC SQL

When SQL was first introduced in SAS, my student assistant always used PROC SQL. I thought “why isn’t he using PROC SORT, PROC MEANS, and PROC PRINT and merge (join) the data sets in the data step. Hmm, he was able to use one procedure to do many tasks." It still took me a while to take the plunge and learn SQL. After all, all my old tools still worked. There is a learning curve, because it is difference from the SAS procedures. I learned about Cartesian joins and how not to create thousands and thousands of unwanted records. I learned to test joins on subsets of my tables. I learned to include commas to separate the variables in the SQL statement. Today, it is the first tool I reach for, instead of the previously mentioned procedures.

Easy solutions to complex tasks are available with PROC SQL. If you are not using PROC SQL, consider these reasons for learning it: PROC SQL is easy and efficient to use for summarizing data and remerging the summarized data with the original data. PROC SQL is a natural for complex joins (merges) and has the ability to query up to 32 tables (data sets) simultaneously. SQL is a standardized, widely used language in data processing. By knowing SQL, you are more marketable and you know a common language to communicate with your data processing colleagues.

These examples show you how easy it is to produce results in PROC SQL.

Get summary statistics by status and year.

```
proc sql;
   select Sex, Age,
      count(*) as n,
      avg(Weight) as avg format=6.2,
      std(Weight) as std format=6.2,
      min(Weight) as min,
      max(Weight) as max
     from sashelp.classfit
   group by Sex, Age;
run;
```

Sex	Age	n	avg	std	min	max
F	11	1	50.50	.	50.5	50.5
F	12	2	80.75	5.30	77	84.5
F	13	2	91.00	9.90	84	98
F	14	2	96.25	8.84	90	102.5
F	15	2	112.25	0.35	112	112.5
M	11	1	85.00	.	85	85
M	12	3	103.50	22.77	83	128
M	13	1	84.00	.	84	84
M	14	2	107.50	7.07	102.5	112.5
M	15	2	122.50	14.85	112	133
M	16	1	150.00	.	150	150

A powerful feature of PROC SQL is the ability to place the results of the select statement into a macro variable. Here is an example of getting the list of variable names in the data set SASHELP.ZIPCODE and putting them in the macro variable &myvar. Notice that in the SQL code the macro variable myvars is preceded by a colon (:).

```
proc sql noprint;
   select name into :myvars
          separated by ', '
     from sashelp.vcolumn
    where libname='sashelp'
      and memname='zipcode';
run; quit;
%put MyVars: &myvars;
```

Results:

```
MyVars: ZIP, Y, X, ZIP_CLASS, CITY,
STATE, STATECODE, STATENAME,
COUNTY, COUNTYNM, MSA, AREACODE,
AREACODES, TIMEZONE,
GMTOFFSET, DST, PONAME, ALIAS_CITY
```

Sorting Without Regard To Case In SAS 9.2

To sort data alphabetically, regardless of case, use Proc SQL.

```
proc sql;
  create table Sortsale as
  select * from Sales
  order by
    upcase(Company);
quit;
```

Before SAS 9.2, to sort regardless of case, you either used the PROC SQL with the upcase function, as shown above, or a two-step process. That two-step process was to create an uppercase version of the by variable in the data step and then sort using that new variable. There's a simpler solution with the SORT procedure in SAS 9.2. The SAS Usage Note 31369: *Sorting Text Without Regard to Case in SAS 9.2* explains how to use the SORTSEQ=LINGUISTIC option with PROC SORT. Linguistic Collation is the culturally expected ordering of characters in a particular language. Here's the example from that Usage Note. The linguistic rule to treat alphabetic characters equally regardless of case is adequate for many applications. Check out the documentation to find out what options to use for various languages.

```
proc sort data=maps.names out=territories
          sortseq=linguistic(strength=primary);
  by Territory name;
  where Territory contains "France";
run;
proc print data=territories;
  var Territory Name;
  title1 "Territories of France";
run;
```

SAS Enterprise Guide

Use SAS Enterprise Guide to do your analyzes. SAS Enterprise Guide is a powerful user interface to SAS. It's easy to use and ready-to-use tools and tasks enables you to focus on the task, not the code. It accesses the power of SAS without learning the SAS programming language. It generates the SAS code and produces the results through a point-and-click interface. It is also for programmers. In SAS Enterprise Guide, you can use the code editor to create new code or modify existing SAS programs. You can modify tasks by inserting code. I find it easier to create my PROC TABULATE code by running the summary task in SAS Enterprise Guide and using the created code as a starting point.

So how important is the SAS Enterprise Guide? Have you noticed that there is a section at SAS Global Forum devoted to SAS Enterprise Guide. There are many papers from this and previous Global Forums for you to learn more. Also there are on-line tutorials, the book *SAS for Dummies* and the book *The Little SAS Book for SAS Enterprise Guide*.

The SAS Online Resources for Statistics Education website has steps with short movies on how to perform basic statistics using SAS Enterprise Guide. Go to the website, select which version of SAS Enterprise Guide to view, select an analysis from the list in the left frame to open a simulation window. There you will be able to view the steps for the analysis in SAS Enterprise Guide in a brief movie.

If possible, use the latest version of SAS Enterprise Guide because the enhancements are worth it to upgrade.

OUTPUT

Focus On The Results

Focus on the results and on the user of the results. Ask yourself how can you deliver the information in a form that is best understood and used by the decision makers.

Data presentation comes in many forms – summary reports, detail reports, lists, statistical tables, exports to Excel and graphs. Visualize how you want the data displayed. Look at examples and other company reports but think about how you can help the reader of your reports better understand the information displayed.

Look at the delivery mechanism of your reports. Instead of a printed report, you could produce graphs, put it on the company website, put it in a PDF or Excel spreadsheet or use SAS to email the results. Even consider if the report is

really needed or reporting some other results might help the decision making process. With the Output Delivery System, you have the means to deliver the results in so many ways and forms.

Understand the data to determine the size of your report and what labeling, formatting or grouping is needed. When developing your tables with PROC TABULATE, first explore and understand the data by running procedures like DATASETS, UNIVARIATE, FREQ, and MEANS. Create a mock-up of how you want the tables to appear. Build the tables with test data, and check the result. Double check the values of summary information and percentages. Determine how you want to handle missing values. When you like the layout and the test results are correct, turn your attention to making the table look good. Use titles, formats, labels, and options to make your report more readable and impressive.

After knowing how you want to present your data, you need to decide which SAS procedure or technique is best to use. Taking the time to choose the best tool to use will save you considerable time in the long run and be more efficient.

Spend the time up front to save time in the long run.

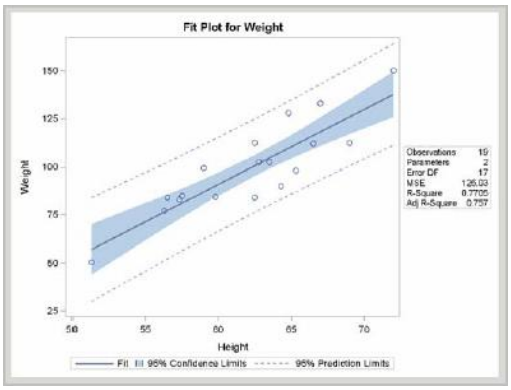
Use Graphs

Effective graphs clarify and focus the data. Ideally they provide information quickly and unambiguously. Just do a Google search on "sas effective graphs" to find many SAS papers on designing effective graphs. Tufte's book *The Visual Display of Quantitative Information, 2nd edition* shows how to display data for precise, effective and quick analysis. It can be borrowed from your local library or added it to your own collection.

Programmers with SAS/Graph experience will be amazed at how simple it is to create complex statistical graphs with ODS Statistical Graphics. New programmers may not be amazed because they do not know the drudgery of writing SAS/Graph code to get the results and layout they want. To learn more, see the SAS TALKS Webinar *Getting Started with ODS Statistical Graphics in SAS® 9.2* by Bob Rodriguez. He's a statistician (fellow of the American Statistical Association) who gives an enlightening talk. If your company licenses SAS/Graph, be sure to view this Webinar.

Here's an example:

```
ods graphics on;
ods html style=statistical;
ods select parameterestimates fitplot;
proc reg data=sashelp.class;
  model weight=height;
quit;
ods html close;
ods graphics off;
```



Useful Printer System Options

If your printout in the SAS session begins with page 256, it reveals the number of times you ran the program before printing. Reset the beginning page number before your final run.

To reset the page number, the code is:

```
options pageno=1;
```

To set orientation and paper size:

```
options orientation="landscape";
options orientation="portrait";
options papersize="legal";
```

To remove page breaks and put dashes (-) across the page:

```
options formdlim="-";
```

To reinstall page breaks (there is no space between the quotes):

```
options formdlim="";
```

To left justify output:

```
options nocenter;
```

Better Titles For BY Groups

To put the variable names and/or values in TITLE, use #BYVAR(byvariable) for the BY variable's name and #BYVAL(byvariable) for the BY variable's value. The statement OPTIONS NOBYLINE; eliminates the default by lines.

```
options nobyline;
proc print data=school;
title "scores grade #byval(grade)";
   by grade;
   pageby grade;
run;
```

Formats

A format tells SAS how to write data values or group data values together for analysis. SAS has a library of numerous built-in formats for character, numeric, and date and time values. You can also create user-defined formats with PROC FORMAT.

A user-defined format can be applied to many variables. Therefore, you do not need to create identical user-defined formats with different format names for different variables. In this example, the $YesNo format is used for all three variables.

```
proc format;
   value $YesNo '0'='No'
                '1'='Yes';
proc freq data=mylib.weather;
   tables sunny cloudy rain;
   format sunny cloudy rain $YesNo.;
```

Instead of using a series of IF statements, recode a variable into a new variable by using a user-defined format and the PUT function.

```
proc format;
    value Department
      1,3,5='Admin'
      2,4  ='ITS'
      6,8,9='HR';

 data Newdept;
    length Dept $5;
    set Employee;
    Dept = put(Deptno,Department. );
run;
```

To learn about formats see Lund's paper *Using SAS® Formats: So Much More than "M" = "Male."*

Output Delivery System (ODS)

The Output Delivery System (ODS) can change your standard dull listings to amazing output. ODS controls the look of all procedure output. Each output is composed of two component objects: a ***data*** component, which contains the data values and a ***template*** component, which contains how the piece should look.

The output destinations can be listing (the default), HTML, output data sets, PDF, RT, XML, Excel (via the ExcelXP tagset) and more. Even if you don't use an ODS statement, you are using the Output Delivery System. The default is the standard listing.

ODS gives you many features and capabilities, such as traffic lighting illustrated here. If you leave out the ODS HTML statements in this example, you will get the standard listing.

```
Proc format;
    value $sexfmt
          'M'='light blue'
          'F'='pink';
run;
```

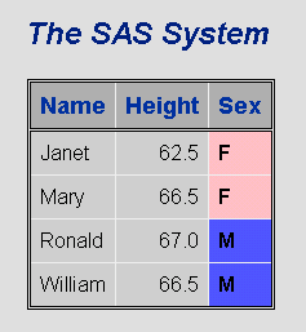
```
ods html file='printout.html';
proc print data=sashelp.class  noobs;
   where Age = 15;
   var Name Height;
   var Sex /style=[background=$sexfmt.
                   font_weight=bold];
run;
ods html close;
```

Standard Listing

The SAS System

Name	Height	Sex
Janet	62.5	F
Mary	66.5	F
Ronald	67.0	M
William	66.5	M

HTML Output



Check out Lafler's paper *Output Delivery System (ODS) – Simply The Basics* and Haworth's paper *ODS Tips & Tricks.* (WUSS 2010). Lauren Haworth, Cynthia Zender, Michele Burlew wrote a comprehensive ODS book called *Output Delivery System: The Basics and Beyond*. It is available on Amazon as a paperback and Kindle edition.

WORK HABITS

Are You Qualified?

This is a composite of actual job descriptions. *A SAS Programmer exhibits creativity and systematic work habits; professional behavior and work habits; well-organized, meticulous work habits and efficient work habits.*

Here are some tips to help you fulfill those job requirements.

Organization And Programming Style

Most papers on programming tips, will mention the need for good documentation, programming guidelines, and organization. Axelrod's paper *Boot Camp for Programmers: Stuff You Need To Know That's Not In The Manual – Best Practices to Help us Achieve Reproducibility* is a selection of concepts and organizing principles that focus on these 4 areas: documentation, processing, programming guidelines, and file storage. She also refers readers to a worthwhile article by Philip Bewig entitled *How do you know your spreadsheet is right? Principles, Techniques, and Practice of Spreadsheet Style*.

To manage project information, we use Microsoft OneNote. There are other tools, like Evernotes. Evernotes is free and is popular on the IPad and the Internet.

Modular Programming

Modular programming is used to break up a large program into manageable units. We develop and keep our code to do different tasks in separate SAS files. Then we use a %INCLUDE statement to include each external file into the current program. The SOURCE2 option causes the inserted SAS statements to appear in the SAS log, otherwise those source statements do not appear.

Here's an example of my program. Because I have already created the data set and checked the data, I placed the asterisk in front of the first two %INCLUDE statements to turn them into comments so they are not executed.

```
/* Master program for High School analysis */
*%include 'C:\HIPS\programs\readdata.sas';
*%include 'C:\HIPS\programs\Checkdata.sas';

title "HIGH SCHOOLS Year 2009";
proc print data=mylib.school;
run;
```

```
%include 'C:\HIPS\programs\summary.sas'    /source2;
%include 'C:\HIPS\programs\finalreport.sas' /source2;
```

If you store your SAS programs in a storage location such as a folder or PDS, you can assign the fileref to that storage area using a filename statement. Then you can just reference the fileref in the %INCLUDE statements. For example, you could write:

```
filename hipspgm 'C:\HIPS\programs';

%INCLUDE hipspgm(readdata);
%INCLUDE hipspgm(checkdata);
```

Recycled Code

There are hundreds of sample programs for most of the procedures and many examples found in SAS manuals. Sample programs makes it easy for you to try out new procedures, view how to write the code correctly, and see the generated output. There are several sources of sample code available for your use. *Samples and SAS Notes* is available at the SAS website under the tab *Knowledge Base.* You can find SAS code from various SAS Press books by selecting the book under the tab *Bookstore* and selecting *SAS code and data* under *More about this book* on the left side of the screen. You can view the SAS Sample Library within SAS help. It is located under SAS *Help and Documentation*. Select *Learning to Use SAS* and *Sample SAS Programs* under the *Contents* tab.

Write, test, document and store your own re-useable code. Keep a log and library of your code. Use macros to repeat code rather than coding repeatedly.

When you run a task in SAS Enterprise Guide, it generates SAS code. Save the code generated by SAS Enterprise Guide and reuse it. I like the drag and drop features of Enterprise Guide. I prefer to run my summaries tables in Enterprise Guide and copy the code to a SAS session where I modify and tweak it for the final results.

Keep An Open Mind

If the program no longer works, ask yourself, “What changed?” Is it different data, a different day, an added improvement to the program, a change in the system or computer. Look for a simple solution first. I will never forgot one episode at the help desk when I asked the student 'what did you change?' He insisted that the program use to work and he didn't change a thing. After I found the error in the code, he replied "I didn't change anything, my roommate did." So the question is "what changed?" not "what did you change?"

Look at statements and techniques with an open mind. Be open to new information and a new perspective and not limit yourself to the same old ways. Even consider whether using SAS is the best choice.

The following is two ways of solving the same problem. The "one way" was the program that was quickly written without forethought. The "simpler way" was the program that resulted after some thought and a desire to keep it simple. The task is to get frequencies of zip code areas defined by the first 3 digits.

One way:

```
data temp;
  set Accounts;
  Zip3 = substr(Zipcode,1,3);
run;
proc freq data=Temp;
   tables Zip3;
run;
```

Simpler way:

```
proc freq data=Accounts;
   tables Zipcode;
   format Zipcode $3.;
run;
```

Enhanced Editor

This may not seem like an efficiency hint but it is a time saver when you are running in an interactive SAS session and have mismatched or unbalanced quotes. Submit this code to fix unmatched comment tags, unmatched quotation marks, and missing semicolons.

```
* '; * "; */;
```

Color code your SAS statements. I set the background color of my comments to be yellow to make it easier to see in the program. Learn to use shortcut keys. Change lines of code to a comment by selecting the text and pressing the Ctrl+/ keys. Remove the /* */ tags by pressing the Ctrl+Shift+/ keys. Use abbreviations to save time typing. Doing the simple things mentioned here can reap benefits. A comprehensive paper to learn more is Harkins' paper *SAS® Enhanced Editor: Efficient Techniques.*

We have been using Ultraedit, a powerful text editor, for years and even purchased a lifetime license. Many of the features that were only in Ultraedit are now in the SAS Enhanced Editor. However, things like searching for text in all the files in a directory or sub-directory, using regular expressions for the search, ease of working with text in columns and other features still make it worthwhile for us. So be on the lookout for other tools and software that will make your work go smoother.

Finally, consider using SAS Enterprise Guide as your everyday SAS editor.

Break Time

Take care of yourself. When your brain is tired, it doesn't work well. Take breaks, stretch, rest your eyes by placing your palms over them for a few minutes and stand up and walk around. Many websites praise the advantages of a stand up desk on your health and weight. Others recommend sitting on an exercise ball. There are many ergonomic tips on the Internet to help you be more comfortable and productive while working long hours on your computer.

Learn new things outside of SAS programming. You don't have to go far – just a click away. Watch online videos from a front row seat and or jog along with a lecture downloaded to your IPod. Do some heavy viewing and listening by visiting the free online academic courses. Some recommended sites are: Open CourseWare (ocwconsortium.org), Khan Academy (khanacademy.org), ITunes U (apple.com/education/itunes-u) and TED(ted.com). These are just a start. One of my favorite TED talks is Jill Bolte Taylor's *Stroke of Insight*. Learning takes time but is worth the effort in the knowledge gained and generation of new ideas.

EFFICIENCY

What Matters Most?

In programming, to be efficient you will want to make good and optimal use of your resources. First ask “what matters most?” Is it computer time, memory, disk storage, computer memory, your time now, or your time in maintaining the program? Look at the tradeoffs. Decide which resources to optimize and which ones you can afford to use less efficiently. If you or someone else needs to maintain the code, write the code using good programming style and organization skills.

Think

Thinking and planning before programming is one of the best tips on efficiency. Marje Fecht in her paper *THINK Before You Type… Best Practices Learned the Hard Way* gives best practices to minimize effort and maximize results. I still have my IBM desk placard that says “Think."

Keep It Simple

Keep it simple, easy and efficient. Use procedures instead of data steps. Avoid unnecessary data steps. For example, permanently save only the final data set. Use temporary data sets to store intermediate results.

Instead of this:

```
data  Mylib.New;
   set Mylib.Sept Mylib.Oct;
   programming statements
run;
data Mylib.District;
   set Mylib.All Mylib.New;
   programming statements
run;
```

Use this:

```
data New;
   set Mylib.Sept Mylib.Oct;
   programming statements
run;
data Mylib.District;
   set Mylib.All  New;
   programming statements
run;
```

Save Disk Space

Save disk space and make your data sets easier to understand by inputting only the data you need. Drop variables you no longer need.

```
DATA mylib.yearly(DROP=Rain1-Rain12);
  SET Old(DROP=Snow1-Snow12);
  Total = SUM(of Rain1-Rain12);
  programming statements
RUN;
```

Drop DO loop indexing variables.

```
data Mylib.NewCost (DROP=i);
   set Mylib.Cost;
   array Amt(100) Amt1-Amt100;
   do i=1 TO 100;
       Amt(i)=MAX(0,Amt(i));
   end;
run;
```

Store numeric categorical data in character variables to save space.

```
length quest1-quest40  $ 1;
```

Suppose you had a 40 question survey with 500 respondents and categorical responses from 1 to 9. It would take 20,000 bytes to store it as 1 character (40 x 500 = 20,000) versus storing it as a numeric of length 8 (160,000 bytes).

The default length of numeric variables in SAS data sets is 8 bytes. To save space, store integer numeric variables in a length less than 8, if you can, by using the LENGTH statement for integer variables. For example, dummy variables that would have only a value of 0 or 1 is a good candidate for having a length of 3. Use the LENGTH statement only for variables whose values are always integers. Non-integer numbers lose precision if they are truncated.

```
length Id s1-s5 4
       Income 8
       default=3;
```

Be careful when choosing the length, because the largest integer represented varies from one host system to another. The largest integer number that you can store under z/OS with a length of 3 bytes is 65,536, on Unix and Windows it is 8,192. See the SAS Companions for a specific host system for more information.

Use Functions For Data Conversion

Use the INPUT function for character-to-numeric conversion and the PUT function for numeric-to-character. The value of CharAmt in the following example is the character string $1000.99.

```
Amount=1000.99;
CharAmt=put(Amount,dollar10.2);
```

In version 9, data conversion can also be done using the VVALUE function. It returns a character string. There is only one argument and it is the variable name whose value is to be converted. It uses the format associated with the variable. Also, the format can be assigned in the program as show in the code below.

```
format Amount dollar10.2;
CharAmt = (Amount);
```

The more interesting function is the VVALUEX function. This allows you to use the value of one variable to get the value from a different variable in the DATA step.The argument is a character expression which must evaluate to the name of a variable. The power comes from being able to define during executing which variable you want to use. This example returns the formatted value of the variable Amount. The name of the variable to use (Amount) was assigned during execution. If the variable "Amount" does not exist, then you will get an error message.

```
format Amount Dollar10.2;
UseThisVar = "Amount";
CharAmt=vvaluex(UseThisVar);
```

The Constant Stays The Same

Assign a constant to a variable in a RETAIN statement instead of an assignment statement.

```
    data School;
      retain TestYear 2008
             District "Honolulu";
      programming statements
run;
```

Null and Void

Use _NULL_when you do not need to create a SAS data set.

```
data _null_;
  set Company;
  Amount=SUM(OF Sold1-Sold31);
  if Amount<90 then
     put Product= Amount=;
run;
```

Avoid Infinity

Avoid division by zero by testing for it. This will save computer time.

```
if Number ne 0 then Avg=Total/Number;
else                Avg=.;
```

Save Time Sorting

Use the NOEQUALS option on PROC SORT to reduce computer time and memory. By default SORT keeps the same order of the observations with identical BY variable values as they were in the input data set. The NOEQUAL option tells SAS to ignore the order of observations within BY groups.

```
proc sort
   data=Survey noequals;
   BY Dept;
run;
```

Read More About It!

There are many SAS Global Forum (formerly known as SUGI) and regional conference papers on programming efficiency techniques. Read Virgile's paper *The Most Important Efficiency Techniques* and Benjamin's paper *Leave Your Bad Code Behind: 50 Ways to Make Your SAS Code Execute More Efficiently*.

RESOURCES

Knowledge is of two kinds: we know a subject ourselves, or we know where we can find information upon it. — Samuel Johnson

Information about SAS software comes from many sources and is contributed by many people, including professional writers, SAS developers, conference presenters, and people asking and replying to questions on the discussion groups SAS-L and SAS Forums. If you only use SAS Institute's Customer Support Center (support.sas.com), you are missing a wide range of resources.

For more extensive information on resources, check out our paper *How to Find Answers: Using SAS Resources.*

SAS Institute's Customer Support Center

Besides using the SAS System help from the toolbar, visit support.sas.com for a world of information about SAS software. Take time to visit the many topics listed in the left navigation bar on their start page. The topics under the *Knowledge Base* is a good place to start. Here you will find the SAS manuals under *Documentation*, technical papers under *Papers*, and sample SAS code under *Samples and SAS Notes*. The *Focus Areas* has informative papers on new and existing products. Select *Base SAS* under *Focus Areas* for papers on ODS and tips sheets on ODS, ODS graphics, and Perl regular expressions.

You will want to view the Webinar SAS Talk *Tips and Tricks for Finding SAS® Information* by Renee Harper, SAS Online Customer Support Specialist. She answers the questions about how to best use the SAS site for finding answers using the search tools. Besides telling about handy search features, she also informs you of ways to stay informed through RSS feeds and watches. This way you will be one of the first in your office to know about new features of SAS.

For a list of SAS blogs that you may want to read or watch, go to blogs.sas.com.

What's New in SAS is another way to learn out about the new features and procedures. It is available at Knowledge Base/Product documentation page. Repole's paper *Don't Be a SAS®Dinosaur: Modernizing Programs with Base SAS 9.2 Enhancements* is another resource.

Step-by-Step Programming with Base SAS Software is in the online documentation on the SAS website and under *SAS Help and Documentation* in the Help Drop down menu of your SAS session. Select the Contents tab of the SAS documentation. Then select *SAS Products*, *Base SAS*, and *Step-by-Step Programming with Base SAS Software.* This book is available for purchase as a SAS press book and for viewing as a PDF. It has an extensive of coverage of SAS programming, 788 pages to be exact. It is a good place to learn about the data step, SAS procedures and the Output Delivery System.

People Power

None of us is as smart as all of us. —Anonymous

Click on COMMUNITY in the left navigation pane at support.sas.com to find out about events like SAS Talks, SAS blogs, forum groups, e-newsletters and much more contributed by SAS employees and SAS users. Here you will find links to sasCommunity.org and SAS-L. Take a look at what people are contributing that will help you in learning more about SAS. At the SAS-L page of the sasCommunity website, you will find links to other cool websites.

SAS Global Forum and most of the regional users group have a presence on twitter, Facebook, and other social media sites. Also check the SAS videos by SAS employees and SAS users at YouTube.

SAS Conference Papers

One way to find SAS papers is to do a Google search. Another way is to visit Lex Jansen's website at **lexjansen.com**, a favorite site of many SAS users.

Save a tree. Do not print out every SAS paper or article that you find that you might want to read one day. They are easier to find if you use a bookmarking tool, save web pages using Instapaper, or write your own list to keep a record of the papers you might want to read later. You can download papers to store on your computer or e-reader. The IPad and e-readers can store and read PDFs and allow markup or bookmarks. With an e-reader you can even read in bed.

A good read for entry level programmers is Varney's paper *Making the Transition from College to Industry*.

Your Own Log

Keep your own log of useful information and write in your SAS manuals your own notes and insights. Document those things you may need to know again, like what did not work and what did.

Retrieving information from memory is not always as quick and easy as we think it will be at the time. Finding information in a notebook or journal, whether on paper or on the computer, is easier than searching for loose pieces of paper scattered about.

CONCLUSION

"*You have your way. I have my way. As for the right way, the correct way, and the only way, it does not exist.*"
Friedrich Nietzsche

Many of these tips pointed to other papers to read. If you spend the time to learn more from those papers, you will reap benefits in the long run. When using SAS software, you quickly find out that there are many ways to achieve your results. There is no the way. So use the tips that work for you. And more power to you.

REFERENCES

Base SAS® 9.2 Procedures Guide: Statistical Procedures, Third Edition, (May 2010), 497 pages, Cary NC, SAS Publishing Inc.

Carey, Helen and Carey, Ginger (2008) *How to Find Answers: Using SAS® Resources,* Proceedings of WUSS 2008, November 2008, Available at http://www.lexjansen.com/wuss/2008/ess/ess05.pdf

Carey, Helen and Carey, Ginger (1997) *SAS Today! A Year of Terrific Tips,* Cary, 396 pages, Cary NC: SAS Publishing Inc.

Carey, Helen and Carey, Ginger (2006) *Paper 132-31: Just Skip It,* Proceedings of the Thirty-first Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc.
Available at http://www2.sas.com/proceedings/sugi31/132-31.pdf

Usage Note 23138: *How can I remove macro code from my original program and create a file of the code that is generated by SAS so debugging the code is easier?,* Samples & Notes, SAS Institute Inc.,
Available at http://support.sas.com/kb/23/138.html.

Usage Note *31369: Sorting Text Without Regard to Case in SAS 9.2,* Samples & Notes, SAS Institute Inc.
Available at: http://support.sas.com/kb/31/369.html

ACKNOWLEDGMENTS

We have learned so much from the volume of papers presented at both SAS Global Forum (formally known as SUGI) and the regional conferences. It only takes writing a paper to feel especially grateful for the authors that gave their time to pass on their experience and wisdom to other SAS users.

RECOMMENDED READING

Data

- Cody, Ron (2008) *Cody's Data Cleaning Techniques Using SAS, Second Edition*, 272 pages, Cary, NC: SAS Publishing Inc.
- Cassidy, Deb (2011) *268-2011: Building a Good Foundation with Functions*, Proceedings of the SAS® Global Forum 2011 Conference. Cary, NC: SAS Institute Inc.
- Dalaney, Kevin, and Carpenter, Art (2004) *128-29 Macro: Symbols of Frustration? %Let us help! A Guide to Debugging Macros*, Proceedings of the Twenty-Ninth Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc.
- Li, Arthur. (2011) *269-2011: The Essence of DATA Step Programming*, SAS Institute Inc. 2011. Proceedings of the SAS® Global Forum 2011 Conference. Cary, NC: SAS Institute Inc.
- Raithel, Michael (2011) *138-2010: PROC DATASETS; The Swiss Army Knife of SAS® Procedures,* Proceedings of the SAS® Global Forum 2011 Conference. Cary, NC: SAS Institute Inc.

- Herbison, Randy (2008) *Replace variable labels with variable names*, SAS-L@LISTSERV.UGA.EDU, Available at http://listserv.uga.edu/cgi-bin/wa?A2=ind0803A&L=sas-l&P=R23858
- Usage Note 23138: *How can I remove macro code from my original program and create a file of the code that is generated by SAS so debugging the code is easier?,* Samples & Notes, SAS Institute Inc., Available at http://support.sas.com/kb/23/138.html

Procedures

- Rodriguez, Bob. SAS TALKS Webinar Series: *Getting Started with ODS Statistical Graphics in SAS® 9.2*, SAS Institute Inc.
- *SAS Online Resources for Statistics Education*. SAS Institute Inc., Available at: http://support.sas.com/learn/statlibrary/
- Usage Note *31369: Sorting Text Without Regard to Case in SAS 9.2,* Samples & Notes, SAS Institute Inc. Available at: http://support.sas.com/kb/31/369.html

Output

- Haworth, Lauren, (2010) *ODS Tips & Tricks,* Proceedings of WUSS 2010, September 2010.
- Haworth, Lauren, Zender, Cynthia, Burlew, Michele. (2009) *Output Delivery System: The Basics and Beyond,* 636 pages, Cary, NC: SAS Publishing Inc.
- Lafler, Kirk (2011) *273-2011 Output Delivery System (ODS) – Simply The Basics*, Proceedings of the SAS® Global Forum 2011 Conference. Cary, NC: SAS Institute Inc."
- Lund, Pete (2011) *271-2011 Using SAS® Formats: So Much More than "M" = "Male"*, Proceedings of the SAS® Global Forum 2011 Conference. Cary, NC: SAS Institute Inc.
- Tufte, Edward R. (2001)*The Visual Display of Quantitative Information, 2nd edition,* 197 pages, Graphics Press

Work Habits

- Axelrod, Elizabeth (2010) *Boot Camp for Programmers: Stuff you need to know that's not in the manual – Best Practices to Help us Achieve Reproducibility.* Proceedings of the SAS® Global Forum 2010 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings10/148-2010.pdf
- Bewig, Philip. *How do you know your spreadsheet is right? Principles, Techniques, and Practice of Spreadsheet Style*, July, 2005. Available at http://www.eusprig.org/hdykysir.pdf
- Fecht, Marje (2009) *133-2009: THINK Before You Type Best Practices Learned the Hard Way*, Proceedings of the SAS® Global Forum 2009 Conference. Cary, NC: SAS Institute Inc.
- Harkins, Kathy (2010) *SAS® Enhanced Editor: Efficient Techniques*, Proceedings of NESUG 2003, Available at http://www.nesug.org/Proceedings/nesug10/ff/ff08.pdf

Efficiency

- Benjamin. William (2010) *FF-08 Leave Your Bad Code Behind: 50 Ways to Make Your SAS Code Execute More Efficiently*. Proceedings of South Central Users Group.
- Raithel, Michael and Rhoads, Mike (2009) *041-2009: You Might Be a SAS® Energy Hog If ....* Proceedings of the SAS® Global Forum 2009 Conference. Cary, NC: SAS Institute Inc.
- Virgile, Bob (2007) *209-2007: The Most Important Efficiency Techniques*, Proceedings of the SAS® Global Forum 2007 Conference. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2007/209-2007.pdf

Resources

- Carey, Helen and Carey, Ginger (2008) *How to Find Answers: Using SAS® Resources,* Proceedings of WUSS 2008, November 2008, available at http://www.lexjansen.com/wuss/2008/ess/ess05.pdf
- Harper, Renee. SAS TALKS Webinar Series: *Tips and Tricks for Finding SAS® Information*, SAS Institute Inc.
- Repole, Warren (2009) *143-2009: Don't Be a SAS® Dinosaur: Modernizing Programs with Base SAS 9.2 Enhancements* , Proceedings of the SAS® Global Forum 2007 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/143-2009.pdf

- SAS Institute Inc. *Step-by-Step Programming with Base SAS® Software*. Cary, NC: SAS Institute Inc. 788 pages, 2001. Available at http://support.sas.com/documentation/cdl/en/basess/58133/PDF/default/basess.pdf
- Varney, Brian, (2000) *220-25: Making the Transition from College to Industry*, Proceedings of the Twenty-fifth Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi25/25/po/25p220.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Helen Carey
Carey Consulting
808.235.4070
carey@hawaii.edu

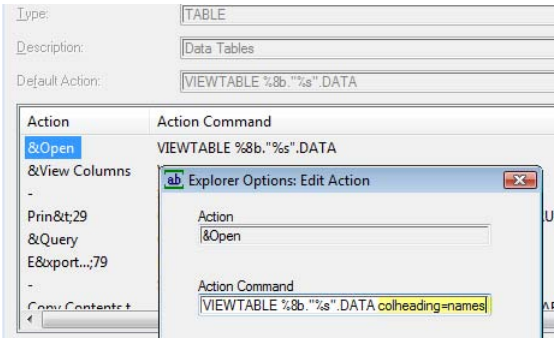
Ginger Carey
Carey Consulting
808.235.4070
ginger@hawaii.edu

APPENDIX: OPENING THE VIEWTABLE WINDOW YOUR WAY

These tips on how to change the default column headings and to easily open an often-used table with the ViewTable window do not fall into the category of "Foundations and Fundamentals." However, in developing and debugging programs, I wanted an easier way to get the view that I wanted of my data set in the ViewTable window. These are some of the tips that I used to make that task easier. It takes some time to get the initial setup but it saves time in the long run.

I use the ViewTable window often in a SAS session. To open a data set for viewing, just click on the data set name in the SAS Explorer window. I wanted to have the column names, instead of labels, as the default column headings. Otherwise after opening the ViewTable window, I would need to click on the *View Menu* and then choose *Column names*. Randy Herbinson offered the following solution in his tip on SAS-L. To change the default action of the library member, do the following.

1. The menu bar is context sensitive. Therefore, first click in the SAS Explorer window in your SAS session to make it the active window.
2. Select **Tools ➧ Options ➧ Explorer**. from the menu bar.
3. Select the **Members** tab in the Explorer Options window.
4. Select **Table** under the column type and then click the *Edit* button.
5. Select the action **&Open** and click the **Edit** button.
6. Add *colheading=name* argument to the action command. *Colheading=label* is the default.
7. Repeat steps 4 to 6 for member type VIEW.



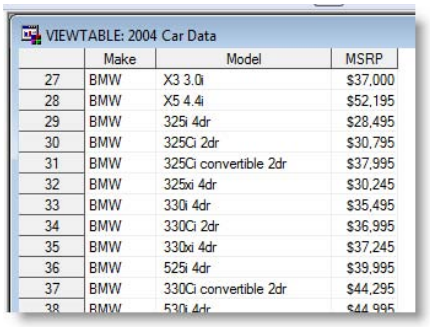
Next I wanted to take it one step further. Because I open the same table with the same layout over and over again, I use this command in the command box. This command will be available in the drop down box for subsequent use.

```
vt sashelp.cars colheading=names columns='make model MSRP'
```

That worked so well, I decided to add selecting certain rows. The WHERE option is used to select only certain rows (observations). Adding the WHERE option makes the command rather long. I decide to embed the VT command in a DM (display manager) statement. The DM statement would be a statement in my SAS program. The DM statement submits the VT command, which must be enclosed in quotes. When you start having quotes within quotes within quotes, it gets tricky. If you are interested in this technique, copy and paste the example and change what is needed. When the dm statement is submitted, the ViewTable window opens. You will need to click its tab at the bottom of the screen to see it.

When submitted, this command opens the sashelp.cars table and displays only the make, model, and MSRP values of BMW cars.

```
dm "vt sashelp.cars
   where="" make='BMW' ""
   colheading=names
   columns=' make model MSRP ' ";
```



	Make	Model	MSRP
27	BMW	X3 3.0i	$37,000
28	BMW	X5 4.4i	$52,195
29	BMW	325i 4dr	$28,495
30	BMW	325Ci 2dr	$30,795
31	BMW	325Ci convertible 2dr	$37,995
32	BMW	325xi 4dr	$30,245
33	BMW	330i 4dr	$35,495
34	BMW	330Ci 2dr	$36,995
35	BMW	330xi 4dr	$37,245
36	BMW	525i 4dr	$39,995
37	BMW	330Ci convertible 2dr	$44,295
38	BMW	530i 4dr	$44,995

Here is another example of using the VT command in a DM statement. Use the macro variable &syslast to open the most recently created dataset.

```
dm "vt &syslast colheading=names"; *;
```