Paper 243-26

Fast and Easy Ways To Annoy a Statistician: The Sharing and Presentation of Data Between a SAS® Programmer and a Statistician

Rick M. Mitchell, Westat, Rockville, MD

ABSTRACT

Although even the best novice SAS programmers may adapt quickly to their work environments, it often takes many years of hard work and millions of lines of SAS code before one may fully understand some of the best and most effective ways to annoy a statistician. When one is having a bad day, or is just plain bored, it is often fun to generate abundant amounts of output where statisticians are forced to sift through page after page for their precious analysis results. It is also amusing to provide as little amount of output and documentation as possible to help clarify discrepancies or oddities of the data for which one knows the statistician will surely question at some point down the road. This paper discusses how a SAS programmer may "make or break" a relationship with a statistician based on the effectiveness of what data are shared and how data are presented.

INTRODUCTION

If one had the power to turn back the clock to their first SAS programming request, there would most likely be a strong desire for some type of framework that the beginning SAS programmer could follow to create an instant bond between themselves and the statisticians for whom they work. When forming a new bond, it can be difficult knowing right away what a statistician's expectations are, and if these are known, then will the programmer have the skills to meet these expectations? Programmers must often face statisticians of varying skill levels who have different preferences and styles of how results should be presented. By understanding the basic approaches to examining data and the standard output that should routinely be provided, the programmer and the statistician will be able to communicate more effectively and prevent an endless number of special requests to get the optimal results. Of course, if a programmer were more fascinated with driving the statistician crazy, then alternatives to these standard approaches should be seriously considered. Throughout this paper, "**Fast and Easy Tips**" are provided on ways to "improve" the relationships between a programmer and a statistician. One may either stop there or continue to read on - this all depends on just what type of mood you're in!

OVERVIEW

This paper will explore the sharing and presentation of data between the SAS programmer and a statistician where four primary topics will be addressed including:

- Preliminary data review
- Giving data the right look
- Analytical checks
- Other useful analytical tools

All of these topics together can form a process that can be interwoven into the working relationship between a SAS programmer and a statistician. While different work environments and players may have an impact on how one's team works together, the basic concepts are similar throughout the world of analytical SAS programming.

PRELIMINARY DATA REVIEW

As a first step to any analytical programming task, it should be obvious to all team members that a preliminary data review needs to be conducted. If either the programmer or the statistician "drop the ball" on this, then serious consequences may arise in the future. This review helps to serve as the foundation for everyone to become more familiar with the data, as well as to bring up any important issues that could potentially cause problems down the road. Two important concepts involved in the preliminary data review include getting to know the data and being fully aware of the missing data.

GETTING TO KNOW THE DATA

Fast and Easy Tip #1: Don't worry about the little details. Surely no one would be sending data filled with errors or discrepancies!

One of the most important elements in maintaining a strong relationship between the programmer and the statistician is a mutual understanding and appreciation of the data. Basically, one should see the data, touch the data, and be the data! This concept is not always routinely followed as programmers are sometimes known to use or are told to use the "push button" approach and deliver results without really knowing or understanding the nuts and bolts of the data. The programmer may simply take for granted that the program is doing what it is supposed to be doing, or may go on the assumption that the statistician will worry about these details. On the other hand, the requestor of a task may also ignore the basics, having assumed that with so much hands-on experience that the programmer better understands the data, or perhaps it is believed that the process is foolproof and that it only requires the push of a little button to produce such wonderful and accurate results. One must keep in mind that the SAS statistical procedures will run on practically any data that are provided and will generate some type of results. The validity of the data is not an issue that SAS will address - only the programmer can control this. Remember - garbage in means garbage out!

Let us review some of the basic steps that one would expect both a SAS programmer and a statistician to understand and follow before plowing too far forward into an analysis. At a minimum, these steps should include a review of the contents of files, a listing of sample observations, frequency distributions, and mean or univariate statistics.

Contents of Files

Although the PROC CONTENTS procedure is one of the simplest SAS tools to run and interpret, it offers a wealth of information that can be very much underappreciated. Before any data processing or analyses take place, there are several important questions that can and should be answered, all by taking advantage of a procedure as basic as PROC CONTENTS. These questions include:

- Do all of the variables appear that are expected?
- Are variables appearing that were not expected?
- Are the variable lengths appropriate?
- Are the variable types of character or numeric being applied appropriately?
- Are variable labels included, and if yes, do they provide appropriate descriptions?

By fully understanding these questions, a SAS programmer should be able to confidently move forward to a closer examination of the variables and their data within the data files themselves.

Listing of Sample Observations

Using PROC PRINT, one may choose to create a listing for X number of records where the length of this listing really depends on the programmer's needs and how well the data are understood. If there are many different combinations of data or odd occurrences of the data that are rarely seen, then one may need to expand this listing to several hundred records. A more simple set of data for which one has worked with more frequently may require a smaller listing of perhaps 25 to 50 records. With this listing, the programmer may consider such questions as:

- Are there unique identifiers to merge the files?
- Are there any special patterns in the data?
- Do all of the variables appear to have data?
- Do text (char.) fields contain meaningful information?
- Are there decimal data or codes within the data?

After a careful review of a listing of sample observations, one is then ready to go down to a finer level and begin examining the specific data within the variables.

Frequency Distributions

Having a good initial feel for what values or codes are present for all of one's analysis variables can be extremely valuable down the road for both the programmer and the statistician. SAS code is written, tested, and implemented for analyses based on the makeup of these data. Discrepancies and outliers should be identified and investigated before getting too deep into an analysis. By utilizing PROC FREQ to run frequencies on the data, one is also able to get a good assessment of the potential impact of missing data, as well as an overall understanding of what data there are to work with. Although frequency distributions are often targeted at categorical variables, it is sometimes helpful to see all possible values for continuous variables as long as one does not generate an excessive amount of output. One might run a frequency distribution on a continuous variable such as gestational age for an infant since this would often be limited to integers, with a range such as 30 to 44 weeks. For variables such as the date of birth for the entire United States population - well, the several hundred pages of output that this could potentially generate may be best saved for one of those days when your friendly statistician dumps a request on you at 4:00 on a Friday afternoon!

Mean or Univariate Statistics

Continuous variables can often be given a good preliminary examination by running some basic descriptive statistics. Even without running frequency distributions on all possible values, one may gain wonderful insight into the makeup of their data, perhaps even a better look. One may have varying needs for the many options that are available in these procedures based on the complexity of the data and their prior knowledge and understanding of the data. For more basic variables, PROC MEANS can nicely present the number of available values, the mean, standard deviation, and minimum and maximum values. While this information is often sufficient, more detailed statistics may be needed for wider range variables for which data distributions may have a strong effect on an analysis. PROC UNIVARIATE can provide additional measures such as percentiles, normality tests, distribution plots, histograms, and for those rare times that one needs to refer to that "useful" text book example of the old stem and leaf plot.

AWARENESS OF MISSING DATA

Fast and Easy Tip #2: Ignore all missing data. If the data were that important, they wouldn't be missing!

While it is sometimes difficult to hold back one's excitement to immediately begin programming away on an analysis, it is important for both the programmer and the statistician to understand and appreciate the impact of missing data. Most data files have at least some pieces of information for which there are no data values. These missing data will ultimately effect an analysis. Some questions that should be asked when trying to understand missing data include:

- What data are missing?
- Is there a reason why the data are missing?
- Can the missing data be obtained?
- Can and should the data be imputed?
- Are there any special patterns (e.g. only 1 of 20 sites has missing data)

Regardless of the type of analysis that is being run in SAS, serious implications may exist if one is not fully aware of what data are filtered into and out of the analysis. For example, if a significant number of records contained missing data for a variable, then an analysis could potentially lose 99% of the data without the programmer or the statistician realizing this. Even a smaller percentage of missing data is an issue that should be carefully considered. Putting various checks in place and conducting a thorough review of the SAS output and logs are important steps that should routinely be integrated into any analysis.

One should be aware that there are various types of mechanisms within SAS to deal with information on the status of missing data. These mechanisms vary across different procedures so it can be difficult to fully understand what data have been included and what data have not. Below in Figure 1 are some of the SAS/STAT® procedures and how they alert the user about missing data within a given analysis. By understanding the subtleties of these procedures, both the programmer and the statistician may better understand the impact of missing data on respective procedure analyses.

	Is Missing Data Information Provided?	
Procedure	**Log**	**Output**
ANOVA	No	Somewhat - must be calculated
CATMOD	No	Yes
FREQ	No	Yes
GLM	No	Somewhat - must be calculated
LOGISTIC	No	Yes
MIXED	Yes	Yes
NPAR1WAY	No	Somewhat - must be calculated
PHREG	Yes	No
REG	Yes	No
TABULATE	No	Not unless specified
TTEST	No	Somewhat - must be calculated

Figure 1

Note that while some procedures indicate the number of missing records excluded from an analysis in the SAS log, others provide this information in the SAS output. Then again, there are other procedures for which the reader of the SAS output is presented with the number of records that were begun with and the number of records that were actually used in the analysis. For these procedures, some minor calculations must be performed in one's head - a task that has the potential to result in error not to mention the annoyance!

GIVING DATA THE RIGHT LOOK

With the preliminary data review completed, one is then ready for the processing and presentation of analysis results. Every set of output results require a special look that is appropriately tailored to the task and that meets the skills and desires of all team players. Achieving the right look requires that one first determine the expectations of a task, understand the advantages and disadvantages of automating the task, and understand the importance of describing the data and the process.

WHAT ARE THE EXPECTATIONS?

Fast and Easy Tip #3: Never reveal how good of a programmer you are and you'll never let anyone down.

To help avoid needless coding and processing of runs as well as generating excessive amounts of output, an initial "kick-off" meeting is highly recommended so that the programmer may grasp early on exactly what great expectations the statistician has in mind. Although often difficult to obtain, it is always helpful to get requests put in writing accompanied by examples or even pictures of what the final product should look like. With this prior information, a programmer can take better advantage of the many SAS tools that are available to tailor one's work to their own personal statistician. The following issues should be considered when identifying the statistician's expectations:

- What does the statistician need to see?
- What would the statistician like to see?
- What can actually be given to the statistician?
- How might this information be needed later?
- What SAS tools are available to accomplish the task?

Of course, each statistician will have personal opinions on what information is needed to move forward on an analysis. A statistician can often get in a routine of always seeing results presented in a certain way. It may not be realized that other valuable information is accessible, or that the programmer has the know-how to better present the data in a format that is more efficient and more understandable. After taking all of these issues into consideration, the programmer should think carefully about what can actually be delivered to the statistician. A product that can be developed in a timely and efficient manner will very much depend on the programmer's skills and patience, as well as the time that is allotted to spend on the task. All of these factors will have some weight on the programmer's ability to either succeed or fail at meeting the statistician's expectations.

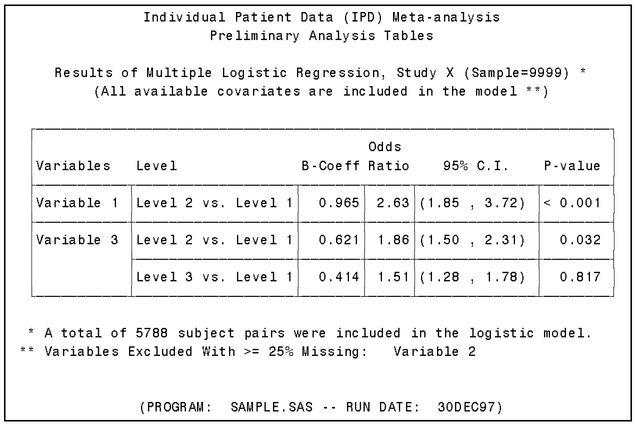
TAKING ADVANTAGE OF AUTOMATION

Fast and Easy Tip #4: Just say those hard tasks can't be done. If SAS had intended for you to do that type of work, then it would have given you a procedure for it!

The last thing that a statistician wants to hear is that the programmer cannot meet the expectations. The statistician often does not care how the programmer gets to the finish line - just get there! In order to accomplish this, it is often necessary to take advantage of the automation potential in SAS. Programmers can sometimes be narrow minded in realizing what possibilities exist when facing a new and challenging task, especially when under a tight timeline. It is sometimes difficult to see beyond one's own skills and experience, and the chance for creativity is not always taken. The author's view is that there is no task that cannot be done with SAS. Those difficult tasks just might take a little extra time! Once the time and energy have been invested, the payoffs can be extremely lucrative if algorithms can either be applied to similar problems in the future, or modified to quickly address new problems. Below are three examples of automated processes that were created by the author to meet very specific client requirements. After reviewing these examples, perhaps new ideas will be sparked as the reader may begin to see that life is not limited to those ever so useful examples provided by SAS Institute!

Automated Example #1

With some careful planning and tricky manipulation of PROC LOGISTIC results, one can package relevant information into a desired presentation format by utilizing SAS output datasets where the final results are presented with PROC REPORT as shown below in Figure 2 (Mitchell, 1998).



```
            Individual Patient Data (IPD) Meta-analysis
                  Preliminary Analysis Tables

     Results of Multiple Logistic Regression, Study X (Sample=9999) *
         (All available covariates are included in the model **)

 ---------------------------------------------------------------------
                                                Odds
 Variables   Level                     B-Coeff Ratio   95% C.I.    P-value
 ---------------------------------------------------------------------
 Variable 1 |Level 2 vs. Level 1|  0.965|  2.63|(1.85 , 3.72)|< 0.001
 ---------------------------------------------------------------------
 Variable 3 |Level 2 vs. Level 1|  0.621|  1.86|(1.50 , 2.31)|  0.032
            |-------------------------------------------------------
            |Level 3 vs. Level 1|  0.414|  1.51|(1.28 , 1.78)|  0.817
 ---------------------------------------------------------------------

  * A total of 5788 subject pairs were included in the logistic model.
 ** Variables Excluded With >= 25% Missing:   Variable 2


               (PROGRAM:  SAMPLE.SAS -- RUN DATE:  30DEC97)
```

Figure 2

In a single page, a wealth of information is provided including level comparisons by variables, parameter coefficients, and odds ratios. Lower and upper confidence intervals for these odds ratios are also presented along with the probability that each variable is significant after controlling for the effects of other covariates. Also note that the table has taken full advantage of other SAS tools by automating a process to incorporate important details within the titles and footnotes that can be applied to any given analysis. These details include items such as what variables are excluded from the analysis, the number of missing values, and the number of subjects in the population. One may better appreciate this table when considering a larger version that could be generated to represent many more analysis variables. Without such a table, one may not be able to avoid producing an enormous amount of output that would add more time and effort to the statistician's job. While the original creation of this table required a series of complex programming steps in SAS Version 6.12, ODS in SAS Version 8 now offers the programmer a much easier mechanism for extracting and displaying this information.

Automated Example #2

Perhaps one of the most challenging automated processes that the author has created pertains to the graph shown in Figure 3 below (Mitchell, 2000). This graph represents an automated process of displaying an analysis comparison of odds ratios where data are pulled from PROC LOGISTIC and presented with SAS/GRAPH® after utilizing some nifty macros. This type of graph itself is not anything special as it is commonly seen in statistical journals worldwide. What is special about the graph is that it is created entirely in SAS using many basic tools that force SAS/GRAPH to meet the programmer's statistical needs. While SAS provides powerful tools to assist the statistician in analyzing and interpreting data, the presentation of results in SAS can sometimes be only of fair quality. By taking advantage of the graphical capabilities that are offered in SAS, a programmer can dazzle and impress co-workers, clients, and the world!

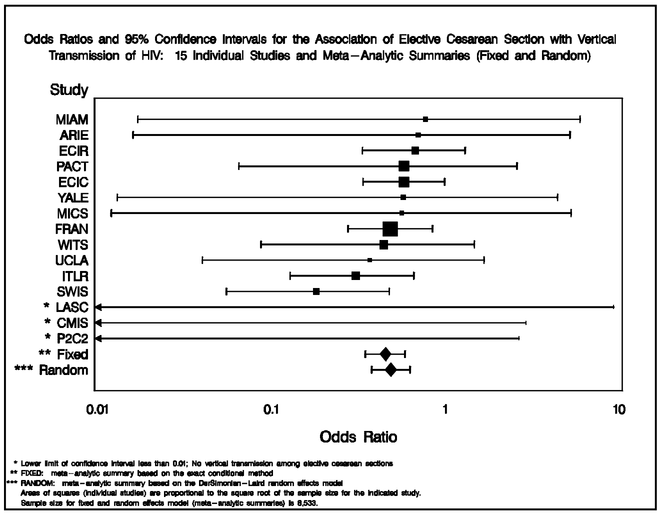


Figure 3

Automated Example #3

As a last example in our automation discussion, Figure 4 below shows how a set of regression result comparisons may be better presented to a statistician. The example shows how a single page can summarize a large number of regression result comparisons when only a few results and calculations are needed. Originally, there were 26 different outcome variables (noted as "Marker") that were each run twice - first with interaction terms, and then without. For each outcome variable, the regression results were compared and summarized for easy viewing and interpretation by the statistician. The ultimate stack of output could have resulted in 52 runs (26 markers vs. with and without interactions) where each run would have generated several pages. Consequently, over 100 pages of output have been condensed into a single summary page. Since the statistician was only concerned with the change in F values, the p-values, and the number of variables and sample sizes, the additional statistics routinely provided in a regression run were eliminated and therefore did not "clutter" the statistician's thinking when reviewing the output. This process not only helped please the statistician, but it also helped save a tree!

```
     Project X - Model Comparisons For With and Without Interactions

                                    Change   # Variables  # Variables
                             Change  In F       With        Without
 # Marker                     In F  P-Value Interactions Interactions  Size

 1 WBC (W. Blood Cell)        0.57   0.6355       7            4        226
 2 Absolute Monocyte          2.96   0.0333       6            4        221
 .
 .
 .
 26 NK Function Assay         0.94   0.4214       7            4        193
```

Figure 4

By taking advantage of the many valuable tools that SAS has to offer, creative programmers may begin to use these tools to their advantage to focus on what data are important, and how these data can be best presented. There are an endless number of approaches that one may take in coming up with an optimal solution for a wide-range of analytical tasks. The use of automation to resolve a problem can save many hours down the road for both the programmer and the statistician.

BEWARE OF AUTOMATION!

Fast and Easy Tip #5: If it worked in the past, then it will work in the future.

While automation is obviously beneficial, one does need to be somewhat careful not to over-automate one's work. The advanced programmer is often able to easily present only the key elements that a statistician is interested in while avoiding endless pages of the same types of output, many of which may never be read. While it is easy and useful for one to scan down a listing of variable comparisons by just their p-values, ignoring the behind-the-scenes process for determining these data could result in gross misinterpretations, especially if there are many records excluded because of a variable containing a large number of missing values. For example, if only the weekly mileage of 10 runners was represented out of an entire running club population of 400 runners, then without the proper review of logs and output, the data could easily be misinterpreted. In general, no matter how much automation is built into a process, or how "pretty" the statistician wants the reported results to look, the programmer must still be responsible for including some type of quality assurance checks with some capability to detect new, unanticipated values or results.

An important factor that should be considered before undertaking any automated process is determining how much time and effort the process will take versus the benefits that will be gained. Some issues that should be considered include:

- Will the process ever be used again?
- Could other software better meet your needs?
- Could the process be applied to other projects?
- Does programming efficiency matter?

Taking all of these issues under review, one may best determine whether it is worth pursuing an elaborate process. In some cases, a more basic and non-technical approach may be more feasible. If a five minute job of typing in a few numbers to a word processing package will do the trick for a one-time only task, then an elaborate automated process probably is not worth the effort.

DESCRIBING THE DATA

Fast and Easy Tip #6: As long as you understand what you are producing, there is no need for variable labels, titles, and footnotes.

A programmer may become so familiar with the data that it is taken for granted that everyone will immediately understand the file structures and corresponding variable descriptions. Whether output results are intended to be distributed as an official deliverable to a client, or if they are going to just be handed out in-house as a draft for basic review, essential basic ingredients should be used routinely including variable labels, titles, and footnotes.

Variable Labels

SAS can only describe a variable as accurately as the information that one gives it, and in some cases, only if specifically requested. Although it may be obvious to some that a variable named DEMO_Q4 is the 4th question on a study's Demographics Form, this doesn't provide much helpful information to the client who may better know this variable as something like "subject age." On the following page in Figure 5 and Figure 6, one can see the difference that a few basic labels can make when presenting data in something as simple as PROC PRINT. While Figure 5 could be sufficient if logical variable names were created that were more informative to the reader, Figure 6 provides an easy mechanism for the reader to instantly get the grasp of what is being displayed. Whether one is looking at complex multiple regression output, or something as simple as a PROC

PRINT listing or a variable comparison in PROC FREQ, explanations beyond the variable name can prevent the statistician from having to ask questions later or from perhaps misunderstanding what is being looked at.

```
                 Sample PROC PRINT - Without Labels and Formats

          SUBNUM        Q1           Q2              RPT_I

            1            1           30                0
            1            2           50                1
            2            1           32                0
```

Figure 5

```
                 Sample PROC PRINT - With Labels and Formats

          Subject      Week         Total          Reported
          Number       Number       Mileage         Injuries

            1            1           30               No
            1            2           50               Yes
            2            1           32               No
```

Figure 6

Labels are also very beneficial when running regression procedures in SAS. Detailed variable descriptions as well as indications of variable levels and reference groups all make for easy interpretation of regression model effects and interactions. For the example in Figure 7 below, if no labels were present, then it may not be obvious that the variable GTYPE represents Gender. Also, it is not known whether males are being compared to females or if females are being compared to males. By providing this information in the variable label (e.g. GTYPE='Gender: Female (1) vs. Male (0)'), one does not need to flip through multiple pages of output to understand all of the model components. In earlier versions of SAS, the display of labels was the default while version 8 "conveniently" makes the default to be no label. Now, one must remember to use the PARMLABEL option in the PROC LOGISTIC MODEL statement for the labels to appear. Despite this annoyance, note that version 8 has included the format category (e.g. the levels of 0-20 and 21-40 are presented next to the variable label, "Mileage Category") along with the label which makes for easier understanding.

```
Standard
Parameter          DF   Estimate    ...   Pr > ChiSq     Label

Intercept           1   -0.6832     ...   0.3177         Intercept 1
Intercept2          1    1.0970     ...   0.0996         Intercept 2
m_cat     0-20      1   -0.9454     ...   0.2675         Mileage Category 0-20
m_cat     21-40     1   -1.1349     ...   0.1088         Mileage Category 21-40
gtype     Female    1   -0.2236     ...   0.6638         Gender:  Female (1) vs. Male (0)
```

Figure 7

Titles

Fast and Easy Tip #7: Use the default title "The SAS System" so that everyone knows what software you are using!

No matter who is going to look at a set of output, title information should be provided that immediately lets the reader know a variety of important facts about the task. These facts will not only catch the reader up to speed on what is being looked at, but they will also come in handy if one were to set the output aside and come back to it several months or even years later. Basically, every piece of output that is generated should have a unique title (or combination of titles) and this information should be assigned as code is written, not coded later. Some of the basic facts that should routinely be included in the title statements include:

- Line 1 - Project name
- Line 2 - Specific task, requestor, and date of request
- Line 3 - Blank (white space makes reading easier!)
- Line 4 - Component that is being presented

An example of these key elements are shown in the SAS statements below:

```
TITLE1  'Serious Runners Club of America (SRCA)';
TITLE2  'Memberships (Requested By Ryan on 1/1/00)';
TITLE3;
TITLE4  'Listing of Fees That Are Overdue By 30 Days';
```

By following a standard format, output items within a given project will begin to look similar and allow the reader to better understand them and get used to looking at certain areas of the output for the relevant information that is needed.

Footnotes

No output should ever be generated without the ever so important footnotes. There should always be something that the programmer wants to note on the output that will either be currently important, or that will be helpful sometime in the future. Some basic items that should routinely be included are:

- Program name, files used, and directory sources
- Freeze date or version date of the data
- Date that the program is run
- Any other notes of interest regarding the output

It is the little things that help bring out meaning to a programmer’s output. By integrating simple items such as labels, titles, and footnotes, output will not only be more informative to the reader, but it can also serve as a reminder to programmers as to what and why they are doing something.

IMPORTANCE OF SAS LOGS

Fast and Easy Tip #8: Ignore the SAS logs. All they give you are repeated warning messages indicating that your license is getting ready to expire.

The SAS log is often taken for granted when a process becomes routine and "push button." However, accompanying deliverables with the SAS log can provide valuable feedback on how and why various steps of a task are implemented. The SAS log is not only helpful in understanding the current process that has been run, but this can also be priceless many months or even years later when someone picks up the program again to run or remind themselves of what has taken place. The log is especially valuable with regression output since it can provide important feedback on whether or not a run is successful as well as to point out any relevant notes regarding a particular regression run. As explained earlier in this paper in the "Awareness of Missing Data" section, the log can sometimes be the only link that will give the reader an idea of how missing data have impacted the run. Below are some important notes that are generated with the SAS log that a statistical programmer should take particular interest in:

```
179 observations read.
NOTE: 38 observations have missing values.
NOTE: 141 observations used in computations.
```

Figure 8

The information shown in Figure 8 above is a wonderful presentation of the data filtering process in PROC REG. One can easily see from this how many records go into the process, how many are excluded, and how many are ultimately included in the analysis. In this example, these numbers are not all provided in the output, so if one were not to look at the SAS log, then it would be impossible to tell how much data are actually represented in the final analysis.

Other warnings and notes that are provided by the SAS logs that are not always shown in the SAS output can result in misinterpretation of the data if they are not thoroughly reviewed. Note that in Figure 9, there is a singularity issue that the statistician must consider as two variables in the PROC REG model have identical values for all observations.

```
WARNING: The average covariance matrix for the SPEC test has
been deemed singular which violates an assumption of the test.
Use caution when interpreting the results of the test.
```

Figure 9

Perhaps one of the most important DATA STEP notes that can be produced is the alert that data sets are not being merged by unique identifiers as shown in Figure 10 below. If this note is not seen by the programmer, then this type of MERGE could produce grossly inaccurate data.

```
NOTE: MERGE statement has more than one data set with repeats
of BY values.
NOTE: There were 316 observations read from the dataset OUT.D1.
NOTE: There were 316 observations read from the dataset OUT.D2.
NOTE: The data set WORK.RICK has 316 observations and 5
variables.
```

Figure 10

These are only a few examples of the notes and warnings that are provided by SAS to aid in data processing and analytical programming. No matter how routine a task becomes, one should never ignore the importance of the SAS log or the valuable information that it can potentially provide.

ANALYTICAL CHECKS

There are a variety of checks that can be easily implemented in one's analytical work to ensure that one is getting exactly what is expected. This area of the paper will discuss some helpful hints that may be used to confirm that variables have been categorized correctly, that one understands the relationships among multiple variables, and that there is an understanding of how the data are ordered.

CONFIRMING CATEGORIZATIONS

Fast and Easy Tip #9: Confirm what? If you have followed all of the specifications, then what else could possibly be expected from you?

After categorizing values into groups, some type of confirmation is recommended. This will not only provide support to the programmer's belief that the coding is correct, but it will also provide concrete proof to the statistician that the request meets the specifications. One method of confirming that there are no unexpected outliers or that values have not been incorrectly categorized is to run PROC MEANS using the same variable for both the CLASS and the VAR statements as shown below:

```
proc means data=testdata nmiss n min max maxdec=1 missing;
   format mile_wk mile_f.;
   class mile_wk;
   var mile_wk;
run;
```

By applying a format to the variable, MILE_WK, results would look similar to those shown in Figure 11 below.

```
                    Analysis Variable : mile_wk

               N        N
mile_wk      Obs     Miss      N          Minimum          Maximum
------------------------------------------------------------------
Missing        1        1      0                .                .
0-20          14        0     14            -50.0             17.0
21-40         46        0     46             23.0             40.0
>40           36        0     36             40.2           9999.0
```

Figure 11

Note that in this example, at least two outliers have been identified (a minimum value of -50 and a maximum value of 9999). Since the values obviously fall outside of the labeled ranges, either the programmer needs to rework the algorithm or the statistician needs to modify the specifications.

This type of check is also useful when applied to a continuous variable that is converted to a categorical variable. For example, instead of applying a format to the variable MILE_WK in Figure 11 above, one may choose to create a new variable called MILE_CAT. Values may then be assigned to different ranges where "0" represents 0-20 miles, "1" represents 21-40 miles, and "2" represents >40 miles. Such a categorization is commonly seen when one needs to run some type of analysis requiring an ordinal variable.

By confirming that variables have been categorized correctly, one can not only ease the mind of both the programmer and the statistician, but one can also provide the opportunity to identify and resolve "dirty" data. The detection of data problems early in an analysis allows the team time to either query a source for further investigation or eliminate the discrepant data altogether.

VARIABLE COMBINATIONS

Fast and Easy Tip #10: Generate as many pages as possible so that the project is thoroughly documented.

A clear understanding of the relationship between multiple variables is essential in understanding the big picture, as well as cutting back on many unneeded pages of output. Too often, programmers do not take full advantage of simple options in PROC FREQ such as LIST and SPARSE. The use of these options can sometimes result in the presentation of information on just a single sheet of paper as opposed to numerous pages of crosstabulations where the reader must constantly flip from one page to another. The LIST option provides an alternative look at the data with several advantages:

- Output can be consolidated into fewer pages
- Output can be easier to read and interpret
- Derived variables can be easily checked against the multiple variables that created them

By including the SPARSE option, one is able to also see the zero cells so that all of the possible categories are represented. Below in Figure 12, one can see how the standard row and column format is transformed into a single listing by using the LIST and SPARSE options.

```
                              The FREQ Procedure



Cumulative
gender                 level       mile_cat       Frequency        Percent
Frequency
────────────────────────────────────────────────────────────────────────
Female    Beginner     Missing              1          5.00              1
Female    Beginner     0-20                 1          5.00              2
Female    Beginner     21-40                1          5.00              3
Female    Beginner     >40                  0          0.00              3
Female    Intermediate Missing              0          0.00              3
:              :         :                  :           :                :
Male      Advanced     >40                  3         15.00             20
```

Figure 12

This table can be very useful when one wants to examine all of the variables that were used to derive a variable. While this output turns what would have been 2 pages of output into 1 page, one can imagine the length of output that would be generated if one wanted to look at all of the possible combinations for a greater number of variables. For example, if a 3 by 3 table (*var_e* vs. *var_f*) was generated for all of the possibilities within 4 variables with 5 levels each (*var_a*, *var_b*, *var_c*, and *var_d*), then the statement

```
TABLES var_a*var_b*var_c*var_d*var_e*var_f ;
```

could potentially generate 5x5x5x5, 3 by 3 tables, resulting in a total of 625 tables. Although the LIST option could not possibly summarize this information on a single page, it would drastically reduce the number of pages and perhaps make for easier reading. One should keep in mind that there is a limit to how many variable combinations may be included depending on the amount of memory that one's computer has available. The restriction is not based on the number of variables, but more on the number of combinations. So, SAS could potentially process 10 variables that produce 3 combinations each while 5 variables with 1000 combinations each would inevitably result in an "out of memory" response from SAS.

UNDERSTANDING DATA ORDERING

Fast and Easy Tip #11: Don't worry about whether a yes/no variable is coded as "1" and "0" or as "1" and "2." SAS is very sophisticated and it will know how to process the data.

When first diving into analyses that are utilizing statistical procedures that are relatively new to a SAS programmer, it can be expected that one may be oblivious to the importance of how data are coded. This issue can very much affect the interpretation of the results and even throw a wrench into an automated process if this "minor" detail is overlooked.

Common Questions That A Programmer Will Be Asked

A programmer should be aware of some of the basic questions that will inevitably be asked regarding regression output results and the ordering of data. These questions may include:

- How is the variable coded?
- Is the variable direction correct?
- What is the procedure testing?
- What are the reference groups?

If the programmer does not immediately know the answers to these questions, then a closer examination of the data should take place in case the old "push button" approach has been applied to the analysis. There are various checks that may be performed to ensure that the results are being interpreted correctly. For example, if results of an analysis suggests that subjects are more likely to run faster if they weigh over 300 pounds, then one might suspect that there could be an ordering problem. By flipping the results of an outcome variable, one also flips the interpretation. Therefore, when one is modeling, it is essential that the programmer not only knows whether variables are coded as 0's and 1's or 1's and 2's, but that the programmer must also have a clear understanding of the reference group (the variable level to which another level is being compared). Basically, the sign of the parameter estimate tells the statistician in what direction a variable is headed. If the sign is positive, then the event is more likely to occur, while a negative sign indicates that an event is less likely to occur.

Checking the Odds Ratios

Fast and Easy Tip #12: SAS procedures are tested, so they must give the correct results.

When running PROC LOGISTIC, serious mistakes may be made if one does not fully understand the ordering of the variables. Having variable data flipped in the wrong direction can result in flipped odds ratios and opposite sign directions for the parameter estimates. There are several ways that the programmer can check that the odds ratios are correct.

As a first step, the programmer should read the SAS log to confirm what is being tested. Note that in Figure 13 below that PROC LOGISTIC has indicated that it is modeling the probability that INFECTED=0 (No).

```
NOTE: PROC LOGISTIC is modeling the probability that INFECTED=0
```

Figure 13

As a default, this procedure tests for the probability of the lowest value within a variable unless instructed to do otherwise (e.g. applying the DESCENDING option). It is often preferable to test that an event will occur rather than not occur such as INFECTED=1 (Yes). Therefore, if one did not realize what a procedure was actually testing, then results could potentially be "flipped" and parameter estimates could change directions (positive to negative and negative to positive).

Another check that a programmer should always perform is a review of the variable formats that are being applied. Generally, FORMATTED is the default ORDER of how data are processed in the statistical procedures. Even if one requested that PROC LOGISTIC model the probability that INFECTED=1 by utilizing the DESCENDING option, the use of formats of 0="Good" and 1="Bad" would still result in the procedure testing for the value of 0 since "Bad" comes alphabetically before "Good."

As a last check of the odds ratios, one may compare the PROC LOGISTIC results to PROC FREQ results using the OR option. If the odds ratios do not match exactly, then the data ordering should be reviewed again. One can also apply the coding checks presented earlier in this paper to confirm that the data are ordered appropriately.

<u>OTHER USEFUL ANALYTICAL TOOLS</u>

This paper will touch on just a few additional analytical tools that a programmer may take advantage of in order to better meet the needs of the statistician. Data transformations and the use of SAS/IML® are two very different topics, both of which are sometimes used sparingly in the workplace. However, these topics are worthy of mention as they can be extremely beneficial to one's work.

TRANSFORMATIONS

Fast and Easy Tip #13: Never change the data. If someone had intended for the data to be on a log scale, then that's how it would have been collected.

Whether or not data need to be transformed for an analysis depends on whether the data are normally distributed. A programmer may often get in such a routine of running exactly what is being asked that the distribution of data are sometimes ignored. Providing the necessary output will allow the statistician to make important decisions regarding the appropriate actions to take. There are several basic steps that a programmer should follow when first introduced to continuous data. Since one can generally count on the statistician to eventually want to see the data distributions, these steps should become part of one's routine process. A general process in one's transformation exploration should include the following steps:

- Examine variables with PROC UNIVARIATE
- Review normality tests, percentiles, plots, etc.
- Apply transformations to better distribute the data
- Rerun transformations with PROC UNIVARIATE

When first beginning one's examination of the data distributions, selected variables should be run through PROC UNIVARIATE with any necessary categorizations where differences may be expected. For example, a distribution of height for males may be different from females, and a distribution of CD4+ counts for HIV+ subjects may be very much different from that of HIV- subjects. By utilizing the NORMAL and PLOT options within this procedure, one is better able to understand the data distribution with the normality statistics, as well as being able to eye the data with the simple plots that are produced. After applying any transformation that is needed, it is necessary to rerun the univariates and reexamine the distributions to determine if the transformation has actually improved the distribution.

After reviewing data, it is often determined that no transformations are necessary since the data are already normally distributed. When transformations are needed, it is most common to take the LOG 10 of a variable, however, there are other transformation possibilities that should perhaps be explored. Below in Figure 14, a standardized report has been produced that allows the statistician to avoid perhaps hundreds of pages of output and have a quick, summary look at the relevant information for possible transformations on a series of variables.

Transformation Exploration
Shapiro-Wilk Test For Normality

Marker	# Subjects	Untransformed (W)	Untransformed (P)	Log 10 (W)	Log 10 (P)	Square Root (W)	.....	Inverse Square Root (P)
CD3	84	0.754	0.000	0.986	0.876	0.940	.....	0.000
CD4	83	0.829	0.000	0.970	0.283	0.920		0.018
CD8	81	0.410	0.000	0.959	0.092	0.711		0.000
CD16	84	0.949	0.007	0.983	0.774	0.971		0.017
CD19	84	0.933	0.002	0.976	0.498	0.963		0.066
CD38	82	0.832	0.000	0.988	0.942	0.936		0.003
WBC	83	0.978	0.594	0.716	0.000	0.915		0.000
ALC	84	0.313	0.000	0.875	0.000	0.600		0.000
ANC	80	0.498	0.000	0.972	0.365	0.862	.....	0.125

Figure 14

This report provides data for a variety of different variables (noted as "Marker") that are displayed in rows, while the transformation results are displayed in the columns including items such as the number of subjects, the raw data (untransformed), a LOG10 transformation, a square root transformation, etc. Other transformations may include items such as the inverse square root, the inverse, etc. Note that both the Shapiro-Wilk Test For

Normality (noted as "W") as well as the probability (noted as "P") are provided for each transformation. The statistician may then easily review each variable to determine the best transformation. While this extensive review can be excessive at times, it is extremely helpful when needing to review a large number of outcome variables that have a wide variety of distribution types. Transformations of data are an essential part of any statistical analysis and can have a strong impact on the results. Therefore, the development and implementation of a routine process to examine data distributions should be seriously considered by all.

LEARNING AND USING SAS/IML

Fast and Easy Tip #14: If you don't use SAS/IML very often then it's probably too difficult to pick up. Suggest that the statistician choose an easier approach.

SAS/IML is one of the most powerful features in SAS, but perhaps one that is seldom used by the majority of SAS programmers. The basic principle behind IML (Interactive Matrix Language) is the data matrix, where the language has the ability to perform complex tasks such as matrix inversions and operations on entire data matrices. At first glance, one may be overwhelmed by the complexity of SAS/IML, however, once some of the basics are learned, a programmer may then begin to utilize some of its useful tools.

An example that may give one a better idea of when SAS/IML can be used is a task that the author was faced with when needing to perform a complex calculation called the "Mahalonobis Distance" (MD). The MD associates the correlation between parameters and standardizes each parameter to zero mean and unit variance by using the covariance matrix in the distance calculation as a distance measurement in cluster analysis where a scale independence is needed. This task could perhaps be programmed without SAS/IML, but it would require extensive programming and many hours of effort. Let us take a dataset of 200 subjects categorized into 26 groups. These data would need to be converted to matrices representing covariances (26x26), means (1x26), and the means for all subjects (200x26). Operations would need to be performed such as the inversion of a matrix, the extraction of the diagonal values in a matrix, and the transposition of a matrix. After the MB formula (not shown) is calculated, a matrix would then need to be converted back to a dataset with the final MB results. This process can be implemented within SAS/IML in just a few simple steps where SAS does all of the work!

Given that one understands the SAS/IML language, what at first may seem to be a monumental task can turn into a problem that can be solved fairly easily. By forcing oneself to learn SAS/IML and apply its strong capabilities to work whenever possible, the SAS programmer may better be able to meet a statistician's highest expectations and mathematical needs.

CONCLUSION

There are an endless number of problems that can be introduced either deliberately or by accident into one's daily routine as data are shared and presented between the SAS programmer and the statistician. As when dining out at a restaurant, presentation counts for major points in one's assessment of their satisfaction. If the food looks good, then it's going to taste good - at least you'll think that it tastes good! This concept also holds well in the presentation of data. The reader of any output should not have to go to great trouble to be able to understand the information that is being presented. Any tricks that can be performed to give the reader a faster and easier understanding of the data should be taken full advantage of. How much effort is made to examine and understand the data in the preliminary stages of an analysis will impact how successful the relationship will be between the SAS programmer and the statistician. By beginning to apply at least some of the concepts presented in this paper to one’s work, a SAS programmer may be better prepared to meet whatever challenges that a statistician chooses to present.

REFERENCES

- Mitchell, R.M. (2000). Forcing SAS/GRAPH software to meet my statistical needs: A graphical presentation of odds ratios (pp. 882-887). *Proceedings of the 25th Annual SAS Users Group International (SUGI) Conference*.

- Mitchell, R.M. (1998). Reporting results of multiple logistic regression models depending on the availability of data (pp. 1227-1231). *Proceedings of the 23rd Annual SAS Users Group International (SUGI) Conference*.

ACKNOWLEDGEMENTS

The author wishes to give special thanks to Larry Muenz, Ph.D. and Jonas Ellenberg, Ph.D., my statistician friends who have tolerated my annoyances for many years, and who both have a great sense of humor.

SAS, SAS/GRAPH, SAS/STAT, and SAS/IML are registered trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

Rick M. Mitchell
Westat
1650 Research Boulevard, WB 496
Rockville, MD 20850
(301) 251-4386 (voice)
(301) 738-8379 (fax)
MITCHER1@WESTAT.COM