Paper 266-28

Fast and Easy Ways To Annoy a SAS® Programmer: A Statistician's Revenge!

Rick M. Mitchell, Westat, Rockville, MD

ABSTRACT

This paper gives a perspective from the statistician's point of view and discusses how a statistician may "make or break" a relationship with a SAS programmer based on the effectiveness of how analysis plans are presented and implemented. When we last met, statisticians were scurrying around in an attempt to destroy all copies of "Fast and Easy Ways to Annoy a Statistician" before their programmers could absorb any of the helpful tips provided in the paper. A memorable conference attendee best summarized her annoyance by saying strongly, "OUR STATISTICIANS DO NOT LIKE BEING ANNOYED!" Now statisticians may seek their revenge by coming up with their own plan to wreck havoc on the life of the very programmers who thought that those fast and easy tips were cute and humorous.

INTRODUCTION

While a SAS programmer may affect the progress of an analysis by either accidentally or intentionally creating mass quantities of output, as well as confusing output, the statistician also has a great deal of power to annoy the programmer in return. The bond that is formed between these two individuals and the approach that they take together in working through an analysis will result in either joint happiness, or in someone being annoyed, confused, and just plain left out in the cold. While it is highly recommended that the standard steps and procedures be taken in conducting an analysis, a statistician who has been working in the field for 20 or 30 years may get bored with these. In such cases, the statistician may choose to follow some alternative standard approaches that are noted throughout this paper as "**Fast and Easy Tips**" (Mitchell, 2001). Statisticians have been known to use many of these steps, some quite often. The end result is that the programmer is often left wondering if the statistician has just not thought everything through clearly, or if indeed the statistician has deviously been plotting out every detail to take revenge on the programmers who found it so amusing to generate abundant amounts of confusing output.

OVERVIEW

This paper will focus on the many steps from the start to the end of an analysis where the burden of implementing these steps are on the shoulders of the statistician. The steps that will be discussed include:

- Kick-off meetings
- Analysis plans
- Changing the plan
- Variable standardization
- Regression basics
- Illogical or confusing requests
- Utilization of resources

The integration of these topics into an analysis will not only keep the programmer and the statistician in tune with the entire road map for the analysis, but it will also help avoid much frustration down the road, most likely on the behalf of the SAS programmer.

KICK-OFF MEETINGS

> **Fast and Easy Tip #1**: Avoid any type of meeting that will waste precious programming time. Meetings are only initiated by people who want to feel important.

What a concept! A SAS programmer and a statistician actually sitting down at the beginning of a new analysis to discuss goals, expectations, and potential problems. This type of kick-off meeting is essential in ensuring the success of any analysis and is the perfect time to begin ironing out many of the finer details of the task. Important topics for discussion in this kick-off meeting may include:

- Introductions and communication approaches
- Description of the project
- Review and discussion of the analysis plan
- Review and discussion of the analysis request
- Discussion of a timeline with milestones

Introductions and Communication Approaches

Fast and Easy Tip #2: Begin an analysis by sending the programmer an e-mail introducing yourself as the statistician and provide an attachment of some vague requests for the new project. Point out that you need this request by tomorrow.

Believe it or not, statisticians have been known to begin analyses without actually meeting the SAS programmer who will actually be doing all of that clever and often difficult coding work. With the convenience of e-mail these days, it can sometimes be overused and the valuable face-to-face meeting can sometimes be ignored. By meeting in person, one is able to gain valuable information that isn't always relayed in e-mail correspondence.

Description of the Project

Fast and Easy Tip #3: Avoid detailed descriptions of the project since this could easily confuse the programmer. The programmer's main focus should be on the SAS code.

A clear, detailed description of the project at the start of the analysis is essential. The programmer's job should really be more than just typing away and handing off some meaningless output. Without understanding what the project is about and what the goals of the project are, programmers may often overlook problems or errors in their output that may appear to be perfectly normal.

Review and discussion of the analysis plan

Fast and Easy Tip #4: Avoid discussions of any plans if possible. This will allow you a chance to update the plan or modify your mistakes without anyone finding out.

The analysis plan will lay out exactly what the statistician plans on doing including hypotheses, statistical tests that will be utilized, how different scenarios will be handled, and the reasons why certain decisions were made. A clear written description of this plan not only helps the programmer see the big picture and understand where they are headed, but it also proves that the statistician has carefully thought out what things will be done and why. Such planning will help avoid unnecessary reruns, reruns, and more reruns down the road. Analysis plans are discussed in greater detail later in this paper.

Review and discussion of the analysis request

Fast and Easy Tip #5: Give minimal documentation regarding your requests (verbal requests are preferable). The less proof of your screw-ups, the better!

Once a plan is in place, the statistician needs to make an official request for how the programmer should address each component of the plan. The statistician and the programmer should also discuss all requests prior to coding to ensure that all parties are on the same page. These requests will also serve as valuable documentation for the analysis such that it will be clear what tasks were performed, when they were performed, and hopefully why they were performed. Such information will be of extreme importance if an actual manuscript is written at a later stage, or even if only some type of summary report were to be written.

Discussion of a timeline with milestones

Fast and Easy Tip #6: Don't discuss due dates. It's easier to blame programmers for missing important milestones if they don't know what they are.

One cannot measure the success or failure of a task without understanding when deliverables are due, or when important milestones are met. With SAS offering such a vast variety of features and options throughout the programming phase, the length of time to code a task could vary greatly depending on how pretty the statistician wants the final product and how many "bells and whistles" are expected. In addition to laying out all important dates on the table, having the ability to identify future dates to meet and discuss issues, output, and progress is imperative.

ANALYSIS PLANS

Fast and Easy Tip #7: Change analysis plans midstream, or better yet, just make it up as you go along!

Some programmers may be surprised at the discussion of analysis plans as they may ask "You get analysis plans?" A plan prepared by the statistician perhaps in collaboration with the programmer will not only make it clear that the analysis has been carefully thought out ahead of time from start to finish, but it will also let all team members in on the "big picture" from early on. A detailed analysis plan will allow the statistician and the programmer to sit down together and discuss the goals of the analysis and the steps that will be needed to reach these goals. Most importantly, the analysis plan will serve as documentation of the analysis should someone need to refer back later to determine what was done and why it was done.

An example of a project that has effectively integrated the analysis plan concept is one in which the author manages the programming of over 50 manuscripts for multiple investigators covering multiple topics in the health arena. For each manuscript, plans are referred to as a logically numbered Primary Analysis Request (PAR) and a detailed PAR is written, maintained, and updated as necessary for each of these 50 manuscripts. With these manuscripts all at varying stages of development from exploratory research to publication in well-known medical journals, one can imagine how an organized method of management would be critical to the success of the

project. The PAR provides several key points of information including:

- Identifying the team members
- Identifying what is being studied
- Defining all analysis variables (existing and derived)
- Identifying what, when, and why decisions were made
- Defining all analyses

Identifying the team members

Fast and Easy Tip #8: Don't introduce the programmer to other team members - you don't want to be embarrassed when everyone finds out that the programmer is actually the one doing all of the work.

One may wonder why it is so important to identify all of the team members when for a given project, contact may be limited for the programmer to merely working with the statistician. By knowing who is involved in the project in the early stages of the analysis, the programmer will have a better understanding of how to share and present output. If the statistician truly is the only other team member, then perhaps limited effort will be needed in manipulating and reshaping the data such that they are more understandable. However, if others will also be included on analysis updates, then the programmer will need to take a different approach. More emphasis should perhaps be put on creating more descriptive titles, footnotes, and labels, as well as integrating fancier techniques that may present only key results so as to not let selected information get lost in the vast amount of results that are provided as a SAS default. In Figure 1 below, a hypothetical team is presented for addressing the prevalence of Ilio-Tibial Band Tendinitis in runners. One can see that the project can bring together a wide range of talents in different areas of expertise.

PAR #87 (Prevalence of Ilio-Tibial Band Tendinitis)
Sample Team Members

- Analyst: Mickey Mouse, Ph.D.
- Statistician: Goofy, Ph.D.
- Programmer: Pluto, BS
- Study Nurse: Minnie Mouse, MSN
- Sports Medicine Physician: Winnie-the-Pooh, MD
- Radiation Technician: Eeyore, BS

Figure 1 - Analysis Plans, Team Members

Identifying what is being studied

Fast and Easy Tip #9: Don't waste time explaining the goals of the project. The programmer's job is to understand the SAS code and nothing else.

We have touched on this topic earlier in this paper when discussing the initial kick-off meeting between the statistician and the programmer, but this is a great way to put it in writing so that others may refer to it again later if necessary. Before a programmer begins taking on the statistician's requests, it would be helpful if the programmer knew what was being studied. For example, one may be studying the prevalence of stress fractures in female marathon runners. This will not only allow the programmer to feel that he or she is more of a part of the analysis, but it may give helpful insight as the programmer works through data problems down the road. If the programmer encountered a variable in the analysis such as the number of females who have run while they were pregnant and it was determined that more than half of these data were missing, then the programmer may determine early on that much of the data were not actually missing but not applicable since half of the overall population consisted of males. By not understanding this minor, but important fact about the data, the programmer could potentially include male subjects throughout the analyses leading to misinterpretations that might not even be discovered for several months and may result in entire analyses needing to be rerun.

Key components in describing the analysis within this section of the PAR might include the:

- Hypotheses
- Specific aims
- Background
- Methods
- Potential Bias

All of these components can help provide a clearer picture to all team members resulting in fewer problems going unnoticed and allowing everyone to contribute their maximum efforts to an analysis.

Defining all analysis variables (existing and derived)

Fast and Easy Tip #10: You've already stated the variables verbally, why waste paper?

One of the most useful components of the analysis plan is the documentation of variable definitions including both existing and derived variables. This documentation serves as an easy and quick reference as to the makeup of all variables involved in the analysis. Not a single variable is taken for granted regardless of its popularity in terms of being used frequently across multiple analyses for even several years. A significant advantage to having all variable definitions in a single location is that modifications only need to be made once and everyone will always be aware of the most current version of the analysis plan. Since team members will be aware of exactly where they may find this current version, endless hours of sifting through papers can be avoided.

In Figure 2 on the next page, one can see how analysis variable definitions can be presented so that both the programmer and the statistician may have a clear understanding of each variable. Components of the analysis plan variable definition section should as a minimum include the following:

- Location (SAS data set)
- Coding
- Description

PAR #87 (Prevalence of Ilio-Tibial Band Tendinitis)
Sample Variable Definitions

Variable	Location	Coding
RACE	DEM.SD7 (Demographics)	1=Caucasian 2=African American 3=Asian 4=Other 5=Unknown or missing
	Description: This variable describes the subject race and is captured on the Demographics form when first entering the study.	
RACE_C	Derived	0=Race code of 1 (Caucasian) 1=Combine race categories of 2, 3, and 4 .=Unknown or missing
	Description: This variable is a categorization of the RACE variable and has been derived specifically for this analysis where "Caucasian" will be the reference group during the modeling phase of the analysis.	

Figure 2 - Analysis Plans, Variable Definitions

Location. Using this format of documenting the analysis variables, the programmer can easily determine where to obtain each variable as the analysis proceeds. A clear statement of "derived" will not only prevent the programmer from searching exhaustively for a variable that is not yet there, but it may also serve as an alert as to what variables perhaps need to be looked at a little closer. Variables that have already been approved and utilized across other analyses most likely have successfully worked their way through some type of QA process in which a variety of checks have been performed. It can be difficult to determine how much thought has gone into the derived variables, so the implementation of additional checks may be necessary.

Coding. All possible codes for every variable in the analysis should be clearly stated in the coding portion of the variable definitions. A statistician should not have to ask the programmer to search through programs to determine a specific meaning of a code, and the programmer should not have to continuously ask the statistician how a specific variable should be coded for a given analysis. By defining all codes up front in the analysis, all team members will have a greater understanding through all subsequent phases of the analysis. Determining ahead of time how unknown or missing codes should be handled can also be of great benefit in the long run.

Description. If the analysis team is able to successfully maintain this section of the variable definitions, then this can serve as a wonderful place to note some type of useful description for each variable. This description can be limited to merely a few words describing the purpose of a variable, or it can provide even greater in-depth knowledge as to the reasons why a variable is being used, how the variable is being analyzed, and any other notes that may be of some interest to either the statistician or the programmer.

Identifying what, when, and why decisions were made

Fast and Easy Tip #11: Don't document modifications. Do you really want someone to know how many mistakes you've made?

An essential component of the analysis plan is not just to layout the road ahead, but just as importantly, to show the road on which the team has driven. It is probably very rare for any plan not to change even slightly from the start to the end of an analysis. This could be in part to the statistician not thinking out the entire analyses ahead of time, or it could be due to unforeseen results that may play a strong role in the evolvement of the analysis. Below in Figure 3, one can see how decisions regarding selected variables have been noted within the project's PAR.

PAR #87 (Prevalence of Ilio-Tibial Band Tendinitis)
Variable Modifications

Variable	Modification
xna_res	On January 21, 2000, it was determined the XNA lab results were tainted and therefore this variable has been dropped from the analysis (requested by JE through RM).
yna_c	On April 20, 2000, an additional level was added to the YNA categorical result variable to accommodate those values at the new lower limit of detection level.

Figure 3 - Analysis Plans, Variable Modifications

With these variable modifications written clearly in a single, easily accessible location, all team members will have quick reference to this information should it be needed at any time in the future. Programmers should strongly consider incorporating similar comments within their programs to serve as an additional source of documentation for these types of modifications. Such comments could come in handy when the programmer or others are reviewing the SAS log, or while the programmer is crafting some piece of algorithm which the variable modifications may affect.

Defining all analysis requests

Fast and Easy Tip #12: Avoid "showing your entire hand" to the programmer. This way, you'll always stay in control and no one will notice if you go off on tangents into nowhere.

The world is full of a wide range of statisticians, many of which may impress everyone by clearly thinking out every step of the analysis before any programming begins. However, some programmers may be lucky enough to work with a statistician who has absolutely no idea what he or she is even going to eat for dinner. While it can be

a difficult process in planning ahead, a plan that is thought out ahead of time in a clear and concise manner will not only save time and money in the long run, but it will also make for a more pleasurable work environment for all involved team members.

Even if it is not clear where an analysis is heading, most analyses should as a standard include a preliminary examination of the data. This process may include basic items such as frequency distributions, crosstabulations, and mean and univariate statistics. Only after these basics have been run should an analysis move into the more complicated phase of work where the programmer may be faced with regressions or other more commonly known or not so known statistical methods of analysis.

In Figure 4 below, a statistician has nicely laid out for the programmer exactly what runs are planned including specifically what SAS or SAS/STAT® procedure should be run along with a corresponding description. This description may offer as little information as just the variables that will be run with the procedure, or it may provide greater detail such as how the procedure can be run, what precautions should be taken, and what steps may be taken depending on the outcome of the runs.

PAR #87 (Prevalence of Ilio-Tibial Band Tendinitis) Sample Analyses

Procedure	Description
FREQ	Run for all demographic variables as noted in the variable definition component of the analysis plan. After the overall frequencies, run a crosstab of all demographic variables separately for male and female subjects. Chi-square statistics should be produced by using the CHISQ option in the FREQ procedure.
UNIVARIATE	Run for IgA, IgA log10, IgG, and IgG log 10 including stem and leaf, normal plot, and frequency distribution.
TTEST	Run for each outcome for each dichotomous predictor variable
REG	Run backward stepwise regression for each outcome variable using predictor variables of X, Y, and Z. Note that additional interaction terms will be added based on preliminary analyses. Include line listing of all observations including values for predictor variables included in model along with PID number, predicted value and upper and lower confidence limits for the mean and for the predicted value.

Figure 4 - Analysis Plans, Analysis Requests

If the statistician and other team members have the time and understanding to describe their planned analyses early on, it will be made clearer as to the direction of the analysis and how long one might expect the analysis to take. Without such a plan, it may be difficult to judge how long a team may need to vie for a programmer's time being that it's not clear whether the analyses will take a few hours, days, weeks, or even months.

CHANGING THE PLAN

Few programmers have escaped the joy of at least once in their careers needing to drastically change an entire analysis plan. For many programmers, changes in the analysis plan can be a routine part of their week (or day!). Factors leading to this annoying problem can be the result of poor planning, or it can be something that no one could have possibly avoided until getting extensively involved with the data. For whatever reason that an analysis plan must be changed, there are several factors that should be considered during all phases of the analytical programming process including:

- Putting excluded subjects on the back burner
- Different cohorts
- Code removal

Fast and Easy Tip #13: After a year long analysis is completed on the eligible cohort, ask the programmer to describe the ineligible cohort.

Putting excluded subjects on the back burner. From personal experience, the author can't stress enough the importance of the concept of having the ability to account for all study subjects at any given time within the analysis. This proved true after the author once worked for perhaps 6 months on a specific study population, and after 37 analysis plan changes later, the statistician requested an extensive descriptive analysis on the very group of subjects that had been excluded at the start of the analysis. As an analysis plan evolves, it can sometimes be difficult to determine exactly who will and who will not be included in the final analysis population. By allowing the programming code to accommodate varying cohorts, programmers can have easy access to whatever group they so desire. If a study has 500 subjects of which 488 were eligible for a given analysis, then the programmer should keep a dataset of the remaining 12 ineligible subjects readily available. This will help not only if information on the subjects are needed, but if the statistician changes the eligibility criteria and it turns out that some or all of these subjects will need to be added to the final analysis cohort.

Fast and Easy Tip #14: Rerun an entire analysis on a different cohort.

Different cohorts. As an analysis begins to take shape, changes to the study population should be expected and the programmer needs to plan ahead so that the SAS code is flexible enough to accommodate these changes. If the programmer introduces such detailed criteria that only one given cohort could possibly work with the program, then many hours could be wasted later on down the road for even the most minimal change.

Fast and Easy Tip #15: Tell the programmer to permanently remove code, then ask for it again later.

Code removal. While some code may change, or even become unnecessary, it is strongly recommended to keep all original pieces of a program. One method of dealing with this is to merely comment out unnecessary pieces of code while noting these exclusions appropriately within

the program. Even though you may think that it will never be used again, "never say never." This concept comes through personal experience where the author has reached the "final" destination on numerous occasions, many months down the road, only to have the statistician ask to reintegrate some algorithm that was thrown away a year before. Sample comments to include within the SAS program might look like the code as shown in Figure 5 below:

```
/*************************************************/
/*  Piglet asked me on 1/1/01 to remove this code    */
/*  that was supposed to address the male cohort,    */
/*  and we are now going to be looking at only       */
/*  female subjects in the analysis (RM).            */
/*************************************************/
/*
data old;
    set notneed;
run;
*/
/*************************************************/
/*  Dumbo said to remove the code for VAR5_C=2, but */
/*  he has absolutely no idea what he's doing, so   */
/*  I'll save it just in case.  (RM, 3/12/01)       */
/*************************************************/

data cats;
   set alldata;
   if var5=1 then var5_c=0;
   /*
   else if var5=2 then var5_c=2;
   */
   else var5_c=1;
```

Figure 5 - Analysis Plans, Documentation

VARIABLE STANDARDIZATION

When dealing with a project that includes multiple programmers or analyses, the concept of variable standardization is essential to maintaining the consistence, and the correctness in the reporting of data. In the short term, it may seem to the programmer that it would be very fast and easy to create any desired variable. However, by thinking more of the "big picture" of the project, much time and frustration can be saved by coordinating analyses such that variable names and their respective codings will be clear and usable to all, not only for current analyses, but for use across multiple, future analyses.

Standard Variable Names and Codes

Fast and Easy Tip #16: Change variable names and codes across analyses.

In Figure 6, one can see how different variable names are used across multiple analyses even though they represent the same information. Additionally, different codes are used to add even more confusion to the poor programmer who is lucky enough to work on more than one analysis.

	Analysis 1	Analysis 2	Analysis 3	Analysis 4
Variables	GENDER	SEX	SUBGEND	SUBSEX
Formats	1=Male 2=Female	2=Male 1=Female	0=Male 1=Female	52=Male 37=Female

Figure 6. Variable Specifications

Standard variable categorizations

Fast and Easy Tip #17: Create categorical variables representing all possible combinations.

While it can be helpful to a programmer to have several categorical versions of a variable, one must be cautious not to over do it. On one of the author's past projects, it was discovered that programmers had created over 12 different variables representing various ranges of CD4 counts. In Figure 7 below, three of these categorized variables are displayed.

CD4CNT_A	CD4_CNT_B2	CD4_CNT_C
0='0-199' 1='200+'	0='0-199' 1='200-499' 2='500+'	0='0-99' 1='100-199' 2='200-499' 3='500-999' 4='1000+'

Figure 7. Overlapping Variable Ranges

These variables were used across multiple analyses by multiple programmers and statisticians. With many of the ranges overlapping, it becomes apparent that one could have used a single variable to contain the absolute CD4 count, while then providing a variety of different formats. Attempting to decipher these multiple variables several months down the road and figure out what a "2" meant for a certain variable, became a logistical nightmare! True, for some types of analyses, it is going to matter what the actual code represents (e.g. performing a trend test where there is a difference between a 0 and 1 vs. a 1st and 2nd group). But, for most analyses, or at least in this particular case, this setup was totally unnecessary and downright confusing for all team members who needed to make sense of a much larger variable file to be used across multiple analyses with multiple programmers and statisticians.

REGRESSION BASICS

Fast and Easy Tip #18: Ask the programmer to "do some regression runs," but don't provide too much more information. Programmers won't understand this anyway.

SAS offers the analysis world a wealth of information intertwined within the many regression procedures that are offered. In fact, there are more than 15 of these regression procedures, all of which are tailored toward some group of data or some specific type of analysis. Many of these procedures, when applied to the same cohort and the same analysis objective, may provide exactly the same results. The procedures are sometimes chosen just on preference, or more often based on the advantages or disadvantages that each procedure offers. Therefore, when a programmer is requested to "run a regression," it may not be immediately clear what approach to take, especially for the beginning analytical programmer. It is for these reasons that the author sees

a need for the following questions to be discussed between the statistician and the SAS programmer before any runs are performed:

- What is the purpose of the regression run?
- Which procedure should be used?
- What is the outcome variable?
- What is/are the predictor variable(s)?
- What are the variable types?
- What are the key components of the output?

Below in Figure 8, a hypothetical data set is created that represents data for a group of 10 runners during the year, 2001. These data will be used to help answer some of the questions that should be posed when a programmer and a statistician begin working on their regression analyses. Feel free to read these data in on your end to help you follow along!

```
data rundata;
   input injury miles restdays stretchdays age beers;
   cards;
      1  1750   10   20  18  365
      1  1850    9   30  19  365
      0  1860   23  310  70    0
      0  1350   25  270  19   12
      0  1400   40  295  20   24
      0  1550   37  320  70   12
      0  1420   52  185  19  200
      1  1400    1   11  18  170
      1  1650   40   17  18    0
      0  1300   42  150  20   87
   ;
run;
```

Figure 8 - Sample Data

Below in Figure 9, a sample regression run is provided to allow us to discuss all of the relevant questions that both the statistician and the programmer should be able to answer before they proceed with the analysis.

```
MODEL INJURY = MILES RESTDAYS STRETCHDAYS AGE BEERS;

Label
  INJURY       = 'Runner injured (yes/no)'
  MILES        = 'Total miles run'
  RESTDAYS     = 'Number of rest days '
  STRETCHDAYS  = 'Number of days stretched before running'
  AGE          = 'Age of runner'
  BEERS        = 'Number of beers consumed in 2001';
```

Figure 9 - Sample Regression Run

What is the purpose of the regression run?

While a comprehensive understanding of the regression run are not necessary for the programmer to successfully meet the statistician's needs, it can be helpful if the programmer at least understands the general purpose. This knowledge can help the programmer mold the output, and it can also be beneficial in identifying any problems that arise before incorrect output is handed off to the statistician, or worse, to the client. Note that the purpose of the sample regression run above is to determine if the independent variables of MILES, RESTDAYS, STRETCHDAYS, AGE, and BEERS can predict the outcome variable, INJURY. Basically, do any of these factors play a role in a runner becoming injured during the course of a year?

Which procedure should be used?

The author is most likely not the only programmer who was ever asked to run some type of regression without knowing exactly what regression procedure to implement in SAS. While a more advanced programmer may understand right away the obvious procedure to use, it is almost certainly not evident to the beginning programmer. In the case of the sample regression run, since we are predicting the outcome of a yes/no categorical variable, PROC LOGISTIC would be an optimal procedure to use since it examines the relationship between discrete responses and some set of explanatory variables.

What is the "outcome" variable?

The outcome variable, also known as the dependent variable, represents what is being predicted based on some given set of potentially related variables. In the sample regression run, the variable INJURY is the outcome variable where we would like to determine if a runner will (INJURY=1, Yes) or will not (INJURY=0, No) get injured during the course of a year depending on the examination of other variables that may be related to running injuries. In Figure 10 below, one can see the default output (the Response Profile) that PROC LOGISTIC generates for the outcome variable where data are ordered from lowest to highest. Note that 4 out of the 10 runners were injured at some point during the year and are therefore coded as "1" while the remaining 6 runners were not injured resulting in a code of "0." Notice that the coding of the variables is important since PROC LOGISTIC orders these responses specially for the analysis, but we can save discussions of this topic for another day.

```
            Response Profile

Ordered                          Total
  Value        INJURY      Frequency

      1             0              6
      2             1              4
```

Figure 10 - Outcome Variable

What is/are the "predictor" variable(s)?

The predictor variable, also known as the independent variable, is chosen to determine if it has any effect in determining the outcome. This does not have to be limited to a single variable, but in fact can include many variables. In the sample regression run, we can see that there are 4 predictor variables including MILES, RESTDAYS, STRETCHDAYS, AGE, and BEERS. So, we would like to know if the number of miles that a runner ran in 2001 has any bearing on whether or not the runner will get injured. Additionally, we would like to know if the number of rest days that the runner took off, the number of days that the runner stretched before running, and age, have any effect on a runner getting injured. Lastly, just

for fun, the author chose to include a variable representing the number of beers that runners drank during 2001. While one might guess that the BEERS variable might just be related to one's performance (how fast one runs), it may be a good idea to keep this variable in the initial model and throw it out later if necessary since you may never know what interesting results you may produce. Note that in Figure 11 below, the Analysis of Maximum Likelihood Estimates produced by PROC LOGISTIC provides the parameter estimates and chi-square probability results for each of the predictor variables in the model (the default columns of standard errors and chi-square statistics are not shown).

```
        Analysis of Maximum Likelihood Estimates

 Parameter       DF    Estimate    …..  Pr > ChiSq
 Intercept        1     10.2606    …..      0.9730
 MILES            1     -0.0181    …..      0.9364
 RESTDAYS         1      0.1869    …..      0.9377
 STRETCHDAYS      1      0.0582    …..      0.9071
 AGE              1      0.1338    …..      0.9731
 BEERS            1      0.0190    …..      0.9583
```

Figure 11 - Predictor Variables

What are the variable types?

Fast and Easy Tip #19: Don't specify which variables should be included in the CLASS statement.

A common oversight among statisticians is that variable types are not also identified up front in preparation for the programmer's constructing of the models. It must be clear from the start whether variables are intended to be used as continuous or categorical variables, with some thought given to perhaps recategorizing or collapsing categories within a variable. Since we have chosen to utilize PROC LOGISTIC in the sample regression run, it's important to know that the outcome variable, INJURY, is a yes/no variable rather than some continuous variable that perhaps reflects the percentage of time that a runner was injured. If this were the case, then PROC REG would be a more optimal procedure, so one can see that by not understanding the data early on, incorrect approaches could be followed, perhaps leading to misinterpretations of the data.

Now for the variable types of the predictor variables in PROC LOGISTIC - continuous variables can be used, however, one needs to tell SAS that a variable should be examined categorically if that is the intent. In the olden days, variables with more than 2 category levels had to be broken up into multiple "dummy" variables where variable 1 might compare level B to level A, variable 2 might compare level C to level A, variable 3 might compare level D to level A, and so on. In version 8 of SAS, the programmer merely needs to identify this type of variable as a CLASS variable and all of this work is magically done by SAS behind the scenes. Note that in Figure 12 below, that appropriately no CLASS statement has been used and the variable AGE is analyzed as a continuous variable as shown in the Odds Ratio Estimates.

```
proc logistic data=rundata;
   model injury = age;
run;
```

```
                Odds Ratio Estimates

                 Point          95% Wald
Effect         Estimate      Confidence Limits

age             227.953       0.003     >999.999
```

Figure 12 - No CLASS Statement

Alternately, if one intended to use a subject's age as a continuous variable but accidentally included this variable in the CLASS statement, then a range of ages from 18 to 70 would be analyzed incorrectly as multiple levels of 18 vs. 70, 19 vs. 70, 20 vs. 70, and so on as shown in the Odds Ratio Estimates of Figure 13 below.

```
proc logistic data=rundata;
   class age;
   model injury = age;
run;
```

```
                Odds Ratio Estimates

                   Point          95% Wald
 Effect          Estimate      Confidence Limits

 age 18 vs 70      <0.001      <0.001    >999.999
 age 19 vs 70      <0.001      <0.001    >999.999
 age 20 vs 70       1.000      <0.001    >999.999
```

Figure 13 - Using a CLASS Statement

What are the key components of the output?

While little effort is needed to merely run regression models in SAS and produce the default output, it can often be better for the statistician, as well as other team members, if the team is able to focus on what the key components should be in the output. If only a few results are necessary out of 3 pages of output, then the SAS programmer can do some magic behind the scenes to only present these key results. The minimization of this output will make the results easier to read and easier to understand!

ILLOGICAL OR CONFUSING REQUESTS

All programmers at some time or another discuss a request with their statistician and then head back to begin the task of programming only to realize shortly thereafter that the request does not make any sense or is just plain wrong. In these cases, not only has the statistician not thought carefully about the request, but the communication has also broken down between the statistician and the programmer because the meeting was adjourned with all parties incorrectly thinking that they knew exactly what needed to be done. Before programming any request, the statistician and the programmer together must agree that:

- It is clear what is being requested
- It is clear what the final product should be
- The request is possible

Illogical Request Example

Fast and Easy Tip #20: Ask for a t-test to be run on a 3-level CLASS variable.

One such common example that the author has faced has been a request for a T-Test on variable X where the data are to be categorized by variable Y. Let us use the hypothetical data presented earlier in this paper for the group of 10 runners. If the statistician requested that a T-Test be performed on the analysis variable of MILES and to be categorized by the variable AGE, we would see the following results as shown in Figure 14 below:

```
Code:

proc ttest data=rundata;
   class age;
   var miles;
run;

SAS log error message:

ERROR: The CLASS variable has more than two levels.
```

Figure 14

Note that after running the requested code, the programmer will be shown a beautiful red message (Sorry, just black and white here!) in the SAS log that indicates that the CLASS variable, AGE, actually represents a 3-level variable (or higher) that the test just won't run. Depending on their level of experience, the programmer may often spend many, many hours trying to figure out the problem with the program itself rather than considering the fact that the statistician could have possibly made a mistake.

Confusing Request Example

Fast and Easy Tip #21: Include a variable whose values are 100% missing in a PROC LOGISTIC model.

Again, a programmer who is either inexperienced or lacks confidence in their code, may show great restraint in going directly to the statistician the minute that a model does not produce ideal results. For example, let us assume that an irrelevant variable such as NUMFOOT, this being the number of football games played during 2001 (let us assume that none of our 10 runners played football the entire year), was added to our existing hypothetical dataset. In a PROC LOGISTIC model, one can be assured that this variable that has 100% of its values as missing (or not applicable) will result in the failure of the model to generate any output at all. In Figure 15 below, one can see the resulting SAS log that is produced by incorporating NUMFOOT into the regression model.

```
Code:

data rundata2;
   set rundata;
   numfoot=.;
run;


proc logistic data=rundata2;
   model injury = miles restdays stretchdays beers
                  age numfoot;
run;

SAS log error message:

ERROR: There are no valid observations.
NOTE:  The SAS System stopped processing this step
       because of errors.
NOTE:  There were 10 observations read from the data
       set WORK.RUNDATA2.
```

Figure 15

Unfortunately, the SAS log is not going to specifically point out to the programmer what the exact problem is - in fact it will merely note an error message saying that "there are no valid observations." The log does not pinpoint exactly what variable is causing the problem, so an entire day or even week could consume the programmer's time in trying to determine why the model that the statistician requested is not running correctly.

UTILIZATION OF RESOURCES

Fast and Easy Tip #22: Request an elaborate macro to produce some rare statistic that SAS already provides.

As programmers gain more experience through their repeated exposure to basic types of analyses, as well as the opportunities of running more sophisticated analyses, it will become clearer when to ask for help and where to look if additional resources are needed. It is not uncommon for a statistician to make a request to the programmer without giving much guidance as to where to go from there. It may not be clear whether the statistician's plan involves components that are easily accessible, or if the plan has been rarely attempted and therefore not commonly supported throughout the SAS community.

The following items are excellent sources of information that may be obvious to the more advanced programmer, but maybe not to the beginning programmer:

- Books with SAS examples other than the typical manual
- Conference proceedings (e.g. SUGI)
- Internet

Non-SAS SAS books

Although SAS provides a wealth of information in its large library of manuals, a different perspective can often be helpful when trying to tackle a problem that wavers slightly from the norm. By identifying books written by

actual users, programmers are given a sometimes better step-by-step approach to solving their problem.

Conference Proceedings

Papers written by SAS users on very specific topics can often provide exactly what the programmer is looking for, resulting in saving weeks if not months of work trying to program some complicated approach. The most well known proceedings are of course from the annual SAS Users Group International (SUGI) conference, however, many of the local users groups such as the North East SAS Users Group (NESUG) also give just as good of a variety of examples. By looking carefully, the programmer is sure to pick up some nifty macro or programming approach that has been rarely heard of, but that matches exactly the type of crazy request the statistician so very much wants to pursue.

Internet

The Internet offers the SAS programmer endless pathways to problem solving solutions. In fact, it may even be easier for one to utilize fast search engines rather than paging through books and proceedings. By merely entering a few words to search on such as "Kappa" and "SAS," programmers can find the answers to their questions very quickly. Suggestions on useful Internet resources include:

- SAS Technical Support web page
- SAS-L
- University statistical departmental web pages

The author has found the SAS Technical Support page very useful when needing to check out rarely used options or statistics for which little documentation is provided in the typical manuals. This search method can also give you an idea of whether other programmers are also facing similar problems, or if they have already discovered solutions or alternative methods. Utilization of SAS-L, a mechanism for SAS users to present questions to the entire world and receive feedback from other users of varying expertise is another resource that should be strongly considered. Lastly, university statistical departmental web pages can also often helpful hints to those programmers needing a deeper understanding of the nuts and bolts behind a statistical method.

CONCLUSIONS

The timing of when information is transferred to the SAS programmer from the statistician, as well as the level of detail that is provided, can have a powerful effect on the outcome of a team's analysis. Strong communication, both verbal and written, is an essential factor that is needed to ensure a team's success. Well documented analysis plans that are clearly thought out can offer a solid foundation that allows the statistician and the programmer to better optimize their efforts and spend more time on the actual analyses rather than needing to perform rerun, after rerun, after rerun. With varying experiences and skill levels, a statistician and a SAS programmer both have the ability to help each other out with the many difficult issues that may be faced during the analytical process. Basically, the statistician can make the programmer look good, while the programmer can make the statistician look good - it's a team effort!

REFERENCES

- Mitchell, R.M. (2001). Fast and Easy Ways to Annoy a Statistician: The Sharing and Presentation of Data Between a SAS Programmer and a Statistician (Paper 243-26). *Proceedings of the 26th Annual SAS Users Group International (SUGI) Conference*.

ACKNOWLEDGEMENTS

The author wishes to thank Gary Elliot who sparked my initial interest in writing a sequel to the original “annoying” paper. Special thanks are also owed to Jonas Ellenberg, PhD, who was the "inventor" of the Programming Analysis Request (PAR) that has been successfully integrated in our statistical analysis process at Westat.

CONTACT INFORMATION

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