Paper SP09

Programming is Done by Human Beings

Ray McKinnis, PRA Health Sciences, Candler, NC, USA

ABSTRACT

In contemporary drug development, computer programming involves two independent individuals who use two different languages: a computer and a human being--one that does computations really fast and the other who plans what needs to be done and writes the code.

The SAS® system took programming into the 4th generation by offering an ingenious system maximizing the use of both the computer and the human programmer: the program was self-contained, code was self-documenting using human language; the computer did what it does best with the human programmers doing what they do best; error messages were in English, and much more.

My presentation reminds us of some of the original, brilliant features of SAS which have been widely replaced by using SAS a a purely 3rd generation programming language.

1. INTRODUCTION

Here we are in Raleigh, North Carolina—where the Statistical Analysis System (SAS) was conceived and given birth. It was one of the first 4th generation programming systems.

When I began using SAS many years ago, the story I was told was that 4 or 5 graduate students in the programming department at North Carolina State University had developed a new system for performing statistical analysis. They realized that statistical analysis using a computer involves two independent contributors which use two independent languages: a computer and a human being--one that does computations really fast and the other that understands what to do and writes the code that tells the computer what to do. They also realized that computers had developed to the point where the language of the computer and the language of the human programmer could (and should) both be included in making maximum use of the both the computer and the human programmer.

Many of the presentations here at PhUSE are efforts to help us make better use of the computer code. Most will suggest better ways of using computer language. Mine will suggest ways to make more efficient use of the human language dimension. I will be using SAS because the possibility of incorporating the human dimension was originally built into SAS.

The original developers of SAS, I was told, were guided by a few simple fundamental principles:

1. Firstly, the code was simple and 'self-documenting'—documenting refers to that aspect that a human being can understand.
2. Secondly, let the computer do what it could do best and let human beings do what they do best.
3. Thirdly, when the computer encounters code it doesn't understand, SAS told the programmer, precisely as it could determine, what was wrong. It gave a message in terms that the programmer understood—in English—they were not computer codes that had to be looked up.
4. On the other hand, when the computer came across code it did not understand, SAS was smart to realize that that a human being could learn something useful about the rest of the program by setting obs=0 and scanning it.
5. Finally, and most important for the human being, everything a human being needed to understand a program was contained within that program. Computers are good at searching lots of files and finding stuff but a human being is not.

Each program was complete in itself for a human being to understand what it did and how it did it.

I remember the first time I was introduced to SAS. I was taking a graduate course in statistics and the assignment was to run a correlation analysis of some data for the next week. I found the computer lab, found a SAS manual, typed in the code and out came the results! Simple enough for someone who had never worked with a computer before.

As the SAS system developed further functionality through the years, the development was primarily in improving the computer code but not the human code. In fact, none of the training I got in the use of SAS even mentioned the human language aspect. When you were learning about SAS, did your courses teach you the human dimension as well as the computer code? My purpose in this presentation is to raise awareness of that neglected potential so that the full functionality of SAS can once again be used making it more efficient and move it again from a 3rd generation programming language to the 4th generation system it was originally created to be.

2. COMPUTER CODE AND HUMAN LANGUAGE

Here is some simple SAS code that anyone should be able to figure out whether they know SAS coding or not:

```
libname sdtm '\\server124\fileabc\sdtm';
  data ae1;
     set sdtm.ae;
   keep usubjid aeterm aesae;
  run;
```

Both you and the computer know exactly where the data comes from, the kind of data it contains (thanks in large part to CDISC) and what variables are needed.

How about this code?

```
%includeit;
  data ae1;
    set sdtm.ae;
 keep usubjid aeterm aesae;
run;
```

The computer knows where to find the ae dataset. Do you? If you get an error message saying 'sdtm.ae' not found, do you know what to do? It has eliminated information important for a human being to understand the program. However, by adding one phrase, a human understanding returns and the program can set up the context:

```
%includeit;
libname sdtm '\\server124\fileabc\sdtm';
    data ae1;
     set sdtm.ae;
 keep usubjid aeterm aesae;
run;
```

Another practice which confuses human understanding found in much SAS code is the creation of a SAS dataset using PROC SORT instead of a data set.

```
proc sort(data=sdtm.ae out=ae);
 run;

data ae1;
     set ae;
 keep usubjid aeterm aesae;
run
```

To a human being, PROC SORT, means to sort a dataset so using in this way already confuses anyone trying to understand the logic of the program.
Using a SAS PROC in a way it was not meant to be used always invites problems.

Another brilliant example of use of the human language offered by SAS is:

```
if inae and insdtm;
```

This is feature of the SAS system can be understood by both the computer and the human being.
However, what happens with this:

```
if a and b;
```

The computer knows what this means but the human dimension has been removed.

SAS was built to allow humans to name things that would give a human being the information to know what was being done. Thus, this is good computer and human code

```
data ae;
```

The name of the dataset can be chosen to be understood by both the computer and the human being.
But change it to this:

```
data data1;
```

or this:

```
data &data1.;
```

the human understanding has been eliminated.

This example suggests one of the main reason the human dimension has practically vanished in contemporary SAS code. If the computer encounters code it does not understand, it gives an error message which you as a human being must fix. However, if you as a human being encounters code that is not understandable to a human being, nothing happens. If somehow every program could be reviewed by a searching tool for code that is incomprehensible to humans and issues error messages, I believe the human aspect of SAS could be brought back and much time and money saved in drug development. Pinacle 21 does this for CDISC how about for basic SAS code itself?

One really clever and useful feature that was built into SAS was the ability to include titles and footnotes in the program itself. Titles are the ultimate in self-documenting. It can tell anyone exactly the purpose of the program.

```
title1 'Listing 14.3.10';
title2 'Laboratory Results from the First Two Periods';
title3 'By Gender and Age';
 proc print data=lab;
   var usubjid param sex lbresult;
  run;
```

With the title in the program, much of the logic of the program can be understood and the titles and footnotes can be reviewed and updated very easily.

But what happens to the human understanding when one encounters code like this?

```
%titles;
proc print data=lab;
var usubjid param sex lbresult;
 run;
```

Anyone who has had to work with title and footnotes which have been put in another file knows the extra work it requires in setting it up and revising it.

Macro language is a powerful functionality of SAS. If the human understanding is included in its use, it can simplify the logic of a program if. But often macros do not include the human dimension in their construction and use It takes special skill to include human understanding in macro code.

This is especially true for those macros that are created to do more than the program needs. Take a macro. Call it %AETABLE. Program it to produce any kind of AE table that one might want to produce. But the program only needs 5 of them. The other functionality of the macro is excess baggage—often confusing things especially if it returns an error message which has nothing to do with the 5 tables the programmer wants to create.

Another brilliant feature included in SAS is the handle missing values accurately. That was a feature intentionally built into SAS—how to propagate missing values. Somewhere missing values got a bad reputation as being undesirable.

```
aedur = aeendt-aestdt;
```

This Is good SAS code that can be understood by the computer and gives humans important information about the data being processed.

Often however I see:

```
If aeenct>. and aestdt>. then aedur=aeendt-aestdt;
```

Which not only cluttered the code but also it prevents the programmer from knowing whether any of these variables have missing values.

The same is true about the transformation of character values to numeric values. This is good simple SAS code:

```
trtn = 1*trtc ;
```

SAS handles it much better than:

```
trtn = put(trtc,best.);
```

Let the computer do what it does best and let the human being do what he or she does best and with a 4th generation computer system like SAS, much efficiency can be gained.

Here is a final exercise to make my point about the genius of the SAS system:

How quickly can you identify what you would use each of these PROC's for:

```
PROC PRINT;
PROC SORT;
PROC MEANS;
```

```
PROC FREQ;
PROC SQL;
```

PROC SQL invites the very kind of 3rd generation programming that SAS was originally created to replace.

3. CONCLUSION

Finally, keep your SAS code simple enough for human understanding. It takes a genius like Steve Jobs or Bill Gates to produce simple computer code but with SAS you too can be a genius and write code that is understandable to the computer and to your fellow workers—if they happen to be human beings too.

The whole CDISC effort was to produce results that were easily understood by the FDA and other groups who were not computer programmers. Pinacle 21 was developed to scan CDISC documents to see if they followed the rules.

And in fact if you are required to use some code that is not clear to human beings, you can describe what it is doing by explaining what it is doing in terms a human being can understand and follow.

Perhaps someone could develop a comparable system that could scan a SAS program and detect any code that did not include the human language dimension. Using SAS only as a 3rd generation programming language could be compared to dancing using only one leg—no matter how clever it might be, it would still only be one leg.

SAS invites us to dance using both legs—it's faster and more fun.

CONTACT INFORMATION

Ray McKinnis
PRA Health Sciences
8 Bennett Road
Candler, North Carolina 28715
1-630-681-9456
dreamsampm@aol.com