

**Paper CS-061**

**The Impossible – An Organized Statistical Programmer**

Brian Spruell and Kevin McGowan, SRA Inc., Durham, NC

**ABSTRACT**

Organization is the key to every project. It provides a detailed history of what the client wanted and shows the means by which the programmer met client goals. It also allows for reproducibility. Finally, and perhaps most importantly, it reduces headache. This paper will demonstrate tools and techniques developed at Constella which were used for a large clinical trials project. Programs were standardized so that all changes would have to be made in one place. A master meta data file was developed and maintained throughout the project. The meta file itself was an excel spreadsheet. It contained columns of information which called certain programs, added titles, would subset the data based on certain variables, etc. This cut down on human error and significantly reduced the programming time.

Keywords: organization, macro, meta file, header

**INTRODUCTION**

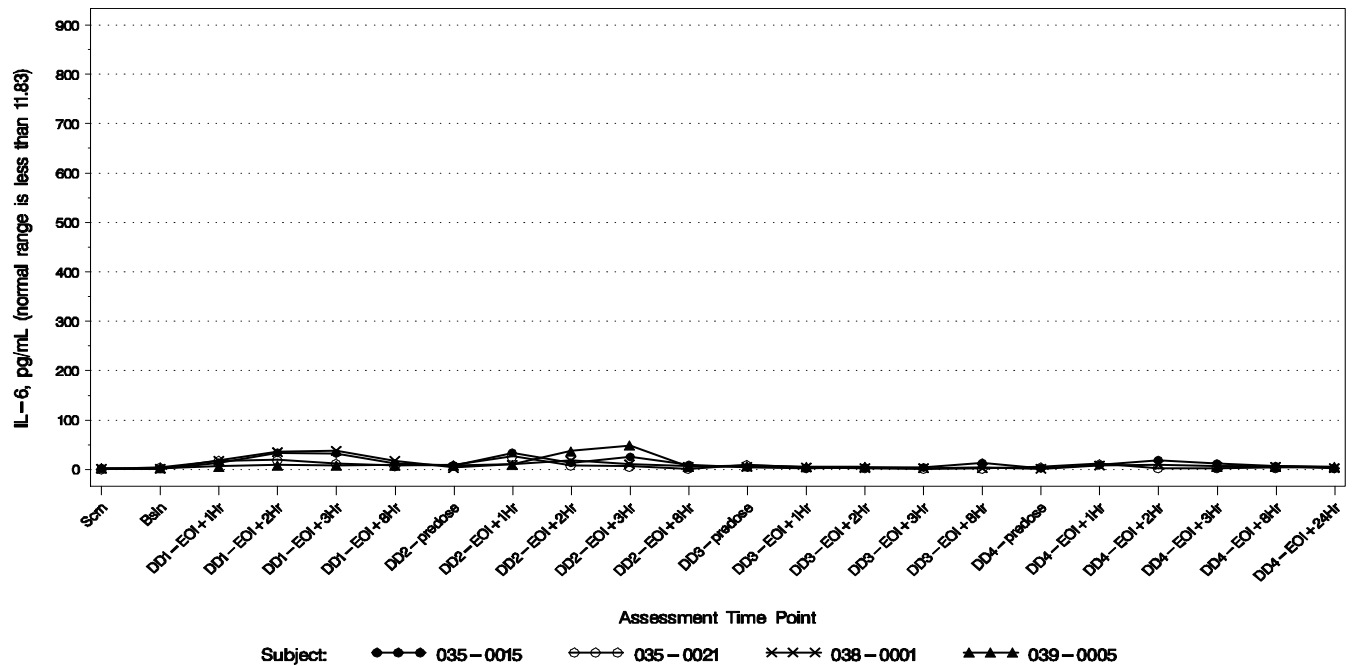
Recently SRA was working with a pharmaceutical company which was in the process of developing a new drug. For obvious reasons I can not mention the drug or the company. This particular company had a meeting scheduled with the Food and Drug Administration (FDA). Our role was to generate various outputs that the client could present to the FDA in order to show the progress they were making on the drug in question. This required SRA to generate hundreds of graphs, tables and listings.

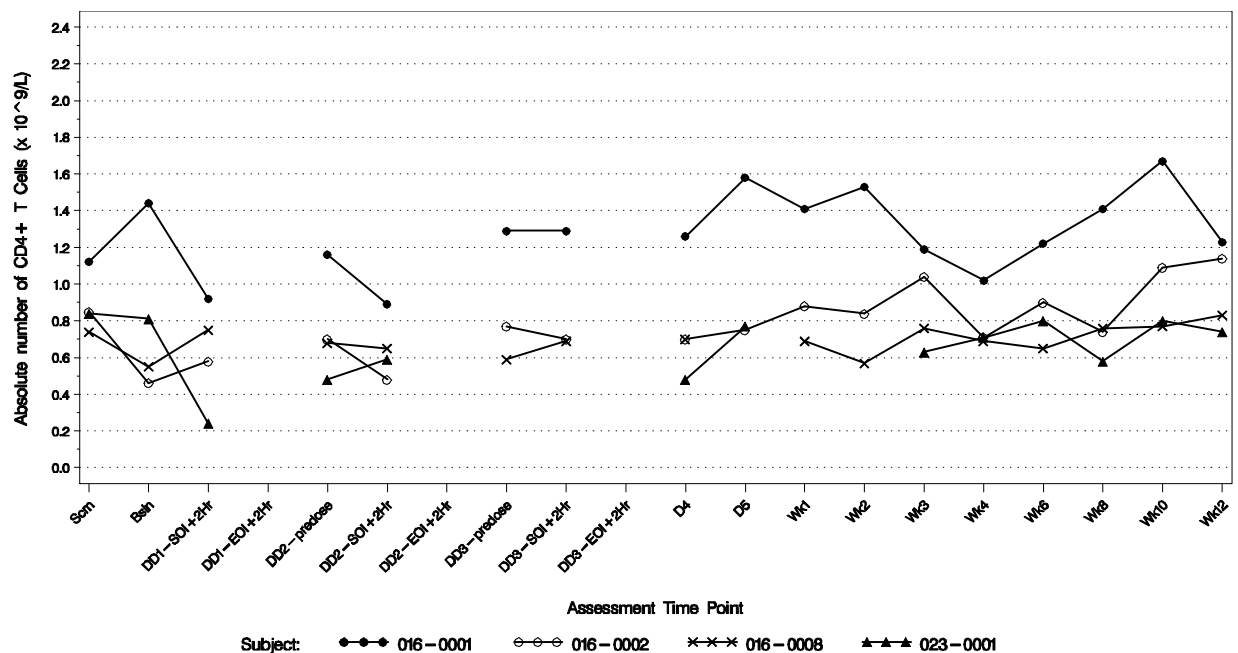
Fortunately many of those graphs and tables were very similar in structure/layouts. Differences came from the actual data, titles, footnotes, labels, etc. Rather than make a 100 different programs to generate each of these outputs, SRA decided to standardize a few programs (six) which were used to generate all the outputs needed. The six programs were written in such a way that they would not require modifications. It was decided that all modifications would be made in one central place. A master meta file was devised in excel which would serve this purpose.

In this paper I also hope to show a standard header used in SRA SAS® programs as well as some programming techniques involving macros and do loops which should make programs easier to maintain.

## THE OUTPUT

The following example will demonstrate the similarities and differences between the various outputs requested by the client. Both are very different endpoints but the graph outputs have similar components:





What changes are the y-label and the points on the x-axis. The actual data that is feed into the program changes as well. TitlesS (not displayed here) are also different. These same similarities and differences exist for the tables and listings as well. We had literally hundreds of graphs and tables to produce. So it was thought best to standardize the process both for ease of making changes and reproducibility. The same SAS program generated these two outputs.

## THE META FILE

The meta file which contained the information needed to generate all the output was built in excel. For the clinical drugs project thirty eight columns were populated. These columns varied in use. They detailed the name of the output file, told SAS which macro program to load, assigned footnotes and labels to graphs, indicated which person was assigned to qc the output, etc. Below is an example of the first four columns:

| PGM NAME   | TFL             | TFL Title                      | PGM Type |
|------------|-----------------|--------------------------------|----------|
| F_PIC_MEAN | Figure 14.2.3.1 | Composite PIC (Mean) Over Time | Figure   |

The program name is really the name of the file generated. It is assigned an appropriate number (TLF) and title which will appear in the output file. The program type indicates that this file is a figure.

| Class | foot1                                                        | foot2     | Input datasets |
|-------|--------------------------------------------------------------|-----------|----------------|
| PIC   | Composite PIC is the sum of TNF-alpha and IL-6 measurements. | 1,2,4,5,9 | Labs           |

These next four columns indicate the class designation (what type of data was being represented) along with footnotes and the input dataset. The dataset column tells the program which dataset to use. The foot2 indicates which set of standard footnotes to use (i.e. company name, label designations, etc.).

| Val    | Select1                  | Select2 | Orientation | Macro Used |
|--------|--------------------------|---------|-------------|------------|
| US_RES | tst_name='Composite PIC' |         | Landscape   | BAR        |

The value column indicates which variable from the labs dataset will be used to populate the figure. The 'Select1' column indicates which tests (in this instance the pic tests) will be displayed in the figure. The Macro used is 'Bar' which tells the SAS program to generate a bar graph for these end points.

The remaining columns are administrative (programmer, etc.). You can see how helpful this is when you have a very standardized process. Everything that changes takes place in the meta file and not in the actual program. The programs are static so you never have to worry about not being able to reproduce an output. Filters were added to the excel spreadsheet to make the navigation of the file easier.

The next section will demonstrate code written which reads in the above excel file.

## THE CODE

The code which reads into SAS the excel file is below:

```
PROC IMPORT OUT= WORK.metadata (where =(pgm_name^=""))
    DATAFILE= "M:\Clinical\Projects\Programming\Tools\
                meta_DATA_update.xls"
    DBMS=EXCEL REPLACE;
    SHEET=" 'newmeta$' ";
    GETNAMES=YES;
    MIXED=NO;
    SCANTEXT=YES;
    USEDATE=YES;
    SCANTIME=YES;
RUN;
```

We used the proc import procedure to input the excel file. We called the resulting SAS dataset metadata. We then specified the location of the meta file on the network. We then indicated which sheet we wanted, 'newmeta', to read in. This allowed us the flexibility of using the spreadsheet's other sheets for something else.

The resulting permanent SAS dataset is saved on the network in the project folder. It is then called by another SAS program which parses all the necessary information to produce output.

The next section will give an example of one of the macros used to generate the figures.

#### MACRO BAR

In our example above we were generating a bar graph for the PIC data. Let us now explore the bar macro program and see how it works.

```
%macro BAR(PGM_Name, Input_Datasets, Val, Select1, YLabel, Precision,
Title0, Title1, Title2, Title3, Foot1, Foot2, Foot3, Orientation);
  data tmp;
    keep Cohort usubjid Visit_TPT_Key Value;
    set adata.&Input_Datasets.;
    %if    (%nrbquote(&Select1) ^= %str()) %then %do;
      Where %unquote(&Select1);
    %end;
    Value = &Val.;
  run;
```

....

The above macro reads in several of the values which were originally put in the excel file. This allows us to keep the macro programs static. Inputs and outputs may change but the mechanism which turns input into output remains the same.

#### THE ALL IMPORTANT HEADER

There were instances where things, due to client requests, necessitated changing the actual macro programs. A mechanism was devised which enabled us to keep track of those changes and provided some accountability with regard to who made those changes.

That mechanism was the all important SAS program header. The following is a standard header used in our division of SRA Inc. It contains all the necessary background information at a convenient location within a SAS program: at the beginning!

```
/* *****
* File Name:
*
* Description:
*
* External References:
*
* Type          Name          Description of Reference
* -----
* INPUT
* OUTPUT
*
* Modification History:
*
* Num    Date    Programmer    Description of Modification
* ---
* 001    mm/dd/yy    Name        CREATED
***** */
```

This is a useful template regardless of what project it is used for. In an easy to find location a new programmer can determine what the program's purpose is. He/she can also determine what types of inputs it requires and what the program outputs.

The most important part of this header is the Modification History portion. In this section we can determine what changes have been made, when they were made and who made them. This is very helpful when you have to re-visit the program after a large amount of time has elapsed. It also provides much needed accountability. You can ask the programmer who made a modification why he or she made that particular change.

```

/*****
* File Name: Sesug2008.sas
*
* Description: Program written to demonstrate usefulness of SAS program header
*
* External References: Developed for the 2008 South Eastern SAS Users Group
*
* Type          Name          Description of Reference
* -----
* INPUT         meta_file.xls    1) Excel file
* OUTPUT        metafile        1) sas dataset
*
* Modification History:
*
* Num    Date    Programmer    Description of Modification
* ---
* 001    04/26/08    Brian Spruell    CREATED
* 002    05/11/08    Brian Spruell    Modified library name to reflect new network location
* 003    06/07/08    Kevin McGowan    Added new columns to import program. See new portion
*                                     labeled 'McGowan updates'
*****/

```

As the above example shows it is now very easy to determine the purpose of this program. We learn that the code that follows the header portion was written for the 2008 SESUG conference. We also learn that this particular program requires an excel file named 'meta\_file' and the resulting output is a sas dataset named 'metafile'. More importantly the modification history portion provides us with some accountability. We know who created the program and then we see changes made to this program and who made those changes.

## MACRO AND DO LOOPS

I recently worked on a project which required generating a large SAS dataset from several smaller SAS datasets. The datasets came from various NHANES (National Health and Nutrition Evaluation Survey) study years (1999-2000, 2001-2002, 2003-2004). The number of datasets needed varied based on which survey year was used. Also each year's datasets existed in separate subfolders. I tried my best to keep the code short and clean by using do loops and macros.

I first created a macro variable named Basedir which would contain the project base directory:

```

%let basedir=M:\CHR\Statistics\Projects\NIEHS-NTP_Task1\Work\Human
Studies\xenobiotic;

```

Second I assigned a libname statement to where I wanted to output the merged dataset:

```
libname outdata "&basedir.\SAS";
```

Next I built a macro which would read in the datasets from various locations and merge them together and finally output them to outdata:

```
%macro builddds;
  /*** SURVEY YEARS ***/
  %let years = %str(1999-2000 2001-2002 2003-2004);
  %let datasets = %str(lab28poc lab06 lab26pp sspfc_a, l28poc_b l06hm_b
    l26pp_b, l28dfp_c l06bmt_c l28ocp_c l24pfc_c);
  %let loops = %str(4 3 4);
  %let names = %str(dioxin mercury pest perflor);
  %local i j;
```

First I created several macro variables which would contain the survey years, dataset names, output dataset names and even the number of times I wanted my macro to loop based on the number of datasets within a survey (the survey 2001-2002 did not analyze certain forms of compounds which were analyzed in the other two years).

I then created my outer loop 'i' which would loop through each of the surveys:

```
%do i = 1 %to 3;
  %let year = %scan(&years., &i, ' ');
  %let loop = %scan(&loops., &i, ' ');
  %let sets = %scan(&datasets., &i, ',');

  libname nhanes&i. "&basedir.\Data\&year.";
```

I used the loop to assign macro variables year, loop and sets all of which are based on the plural macro variables. For instance in the first iteration year was assigned a value of 1999-2000, loop had a value of 4 and sets has a value of 'lab28poc lab06 lab26pp sspfc\_a').

I then had my macro perform another loop called j (j would have as many iterations as defined by the macro variable 'loop'):

```
%do j = 1 %to &loop;
  %let dataset = %scan(&sets., &j, ' ');
  %let name = %scan(&names., &j, ' ');
  libname XP xport "&basedir.\Data\&year.\&dataset..xpt";

  proc copy in=XP out=nhanes&i.;
  run;

  data &name.&i.;
  length nhanesyr $9;
  set nhanes&i.&dataset.;
  nhanesyr="&year.";
  run;

  data Xenobiotic&i;
```

```

        %if (&j = 1) %then %do;
        set &name.&i;
        %end;
        %else %do;
        merge xenobiotic&i &name.&i.;
        by SEQN;
        %end;
        run;
    %end;

```

For the first inner iteration dataset would take on a value of lab28poc and name would be 'dioxin'. Dataset dioxin1 would be created by setting the dataset lab28poc created by the proc copy statement from the 1999–2000 survey folder. For each survey year the dataset Xenobiotic would be created with the outer iteration number as an identifier. The first inner loop dataset would simply be set into this dataset. The subsequent inner loops would be merged by the participant id (SEQN). This Xenobiotic# would then be set into Xenobiotic which would be outputted to the network for further analysis.

```

        data Xenobiotic; %if (&i=1) %then set; %else set xenobiotic;
        xenobiotic&i; run;
    %end;

```

## CONCLUSION

In the beginning the time and effort it takes to ensure that we are an organized programmer may seem substantial. However, in terms of overall costs and reproducibility the long term costs (and potential headaches) are reduced significantly. Hopefully the header and the meta excel file idea will prove helpful in your future SAS programming endeavors.

## ACKNOWLEDGMENTS

Deepak Mav – senior statistician and primary programmer on the meta program.

Renee Jaramillo – statistical programmer on assisted with the meta program and bar macro

## CONTACT INFORMATION

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

Brian Spruell  
 SRA International  
 2605 Meridian Parkway Suite 200  
 Durham, NC 27713  
 Work Phone: (919) 313 – 7673  
 E-mail: [brian\\_spruell@sra.com](mailto:brian_spruell@sra.com)  
 Web: [www.sra.com](http://www.sra.com)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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