Paper 3187-2015

Structuring your SAS® Applications for Long-Term Survival: Reproducible Methods in Base SAS® Programming

Paul A. Thompson, Sanford Research

ABSTRACT

SAS® users organize their applications in a variety of ways. However, there are some approaches that are more successful, and some that are less successful. In particular, the need to process some of the code some of the time in a file is sometimes challenging. Reproducible research methods require that SAS® applications be understandable by the author and other staff members. Structuring your SAS® analyses can either help or hinder the processes of data access, data analysis, and data presentation. These tasks must be compartmentalized. This can be done using a well-defined program. The author presents his structuring algorithm, and discusses the characteristics of good structuring methods for SAS® applications. Reproducible research methods are becoming more centrally important, and SAS® users must keep up with the current developments.

INTRODUCTION

AN ANALYSIS FILE DISASTER

In my 40 years of experience with SAS® and other languages, I have had many opportunities to see the coding strategies that different analysts use. Some are good, while others are terrible.

A key experience involved “project cleanup” after “John Smith” resigned to take another position. I was asked to embrace and extend analyses which had been done by “John”. I found:

- **Formats:** Multiple copies of formats in different files, which were usually the same, but sometimes different although the names were the same. “John” copied the entire file to create a new analysis program and edited the new one.
- **Macros:** Multiple different copies of key analysis macros were found. These were often named the same, but had different functionality, in some cases at a very minor level. No version numbering or dating was used.
- **Key variables:** Key variables values (i.e., treatment condition) were modified.
- **File names:** Inappropriate names for files (i.e., `BOSSFIRSTANAL.SAS`) were used.
- **Documentation:** There was a lack of documentation about the manner in which the files were used. In particular, no documents about the connection between analysis program and output table were maintained.

GUIDANCE ON STRUCTURING ANALYSIS FILES

These sorts of bad practice are unfortunately common. Little guidance is available on good approaches, however. Methodology teams are tasked with the performance of internal and external consultation projects and grant service projects. They must manage the execution and tracking of small single-pass projects as well as larger, longer term projects which are on-going and require long-term maintenance. Managing the SAS® environment in these circumstances has several challenges:

- The code management process should not be onerous for the user.
- Different data analysts should be able to easily locate functional components of each project.
- The methods should make project handoff easy and straight-forward.
- The tools should enhance the use of SAS® components, including the SAS® macro language.

This process of code management is not often discussed by SAS® shops. Often it is not discussed in MPH (and such) programs, where the focus of the program is simply a matter of learning the basics and gaining a little mastery in both SAS® and other languages.

Software creation is sometimes seen as a craft where each person is encouraged to “roll their own” approach to management of files, storage of output, and long-term maintenance of figures and tables. While this approach of “individual empowerment” can be good, it can also mask dysfunctional and poor coding practices which are not likely to lead to long-term success. In addition, the lack of standard methods for organization means that the departure of key staff members may produce a short-term difficulty in performing analysis and continuity of projects.

REPRODUCIBLE RESEARCH

Reproducible research is a current interest of many scientists (Claerbout and Karrenback, 1992; Fomel and Hennenfent, 2007; Peng, Diggle, and Zeger, 2009; Thompson & Burnett, 2011). An analysis process is “reproducible” if the process:

- can be repeated without great effort;
- is well-documented;
- has every step clearly defined; and
- has a clear line from data to final result.

Reproducible methods for analysis are an important part of the entire scientific process.

RR AND WORKFLOW MANAGERS

Reproducible research is a key component of applied statistics work, since it is essential that all work products (tables, figures, statistically oriented text) can be able reliably produced by re-execution of the stored code. The statistician is responsible for the scientific integrity of the summary statistics, both on the interim basis for trial and study monitoring purposes, and for documents produced out of studies. Thus, the data analyst must ensure that a repeatable, dependable process exists to go, step by step, from data to result. A standard approach to the reproducible research requirements in applied statistics is the use of a workflow manager.

SAS® AS A WORKFLOW MANAGER

SAS® can be used as a "workflow manager", which is the process of running components of a task in an ordered manner, while making decisions. A stored SAS® program should be able to run from start to finish to produce all interim and final work products. The components of data processing include some or all of the following:

- **Data Import:** Data are input (i.e., Excel spreadsheets, raw data, external databases).
- **Data Cleaning:** Data are cleaned by:
    - checking variables for errors
    - converting values (i.e., height in cm from height in in)
    - creating new variables (i.e., BMI from weight and height)
    - creating range values from continuous variables
    - creating indicator variables from continuous variables
    - categorizing free text strings
    - labeling variables and values
- **Management:** Data are managed by combining datasets together and by changing their form. This may include the use of the MERGE, SET, TRANSPOSE, and SORT functions. These operations create analysis datasets.
- **Analysis:** Appropriate statistical procedures are used to analyze the data.
- **Reporting:** Statistical results are formed into tables, figures, and "incidental values" (placed in text).

The notion of the workflow manager is a common one in reproducible research. The notion here is that SAS® system structures the order and, in many cases, uses a decision-making process to choose which of the scripted components to actually run.

AN EXAMPLE PROGRAM

Here is a simple program:

```
LIBNAME PROJECT "sasloc/currentsubdir";
/* Read in the data from seta.xlsx ***************************** */
PROC IMPORT OUT=PROJECT.SETA REPLACE DATAFILE= "seta.xlsx" DBMS=EXCELCS;
   RANGE="Database"; SCANTEXT=YES; USEDATE=YES; SCANTIME=YES; RUN;
/* Read in the data from setb.xlsx ***************************** */
PROC IMPORT OUT=PROJECT.SETB REPLACE DATAFILE= "setb.xlsx" DBMS=EXCELCS;
   RANGE="Database"; SCANTEXT=YES; USEDATE=YES; SCANTIME=YES; RUN;
/* Sort the seta data ****************************************** */
PROC SORT DATA=PROJECT.SETA;BY GROUP ID;RUN;
/* Sort the setb data ****************************************** */
PROC SORT DATA=PROJECT.SETB;BY GROUP ID;RUN;
/* Create the PROJECT.MAINSET dataset ************************** */
DATA PROJECT.MAINSET; MERGE PROJECT.SETA PROJECT.SETB; BY GROUP ID; RUN;
/* Compute summary statistics ********************************** */
PROC MEAN DATA=PROJECT.MAINSET; VAR A B C; BY GROUP; RUN;
```

If the user simply runs the project from start to finish, the process of data input (reading in the data from the datasources) and data management (sorting and merging the two datasets) are performed repeatedly, even after they are successfully performed. Once code sections have run correctly, they need not be run again. There are several approaches to run the project in separate sections to control code execution.

METHODS FOR SEPARATED ANALYSIS

There are a number of methods for separated or controlled analysis of code. Each has advantages and disadvantages.

- **Running highlighted sections:** First, each section to be run can be highlighted with the mouse, and executed in that manner. This is simple, but is not reproducible, as there no records of the manner in which the code was executed. This approach does not reproducible results, and should only be used for code testing.
- **Using “comment-out methods:** When code has been run successfully, it can be commented out by using the "/*” and “*/" comment strings to “inactivate” code. This is a very traditional approach, and is commonly used by many SAS® programmers. However, if there are actual comments already in the section to be commented out, this does not work. So, it is not a general solution, and can be hard to work with. This approach does not reproducible results, and should only be used for code testing.
- **Separate files:** Each section can be put into separate files. In our example, three files are needed:

File readin.sas:

```
LIBNAME PROJECT "sasloc/currentsubdir";
/* Read in the data from seta.xlsx ***************************** */
PROC IMPORT OUT=PROJECT.SETA REPLACE DATAFILE= "seta.xlsx" DBMS=EXCELCS;
   RANGE="Database"; SCANTEXT=YES; USEDATE=YES; SCANTIME=YES; RUN;
/* Read in the data from setb.xlsx ***************************** */
PROC IMPORT OUT=PROJECT.SETB REPLACE DATAFILE= "setb.xlsx" DBMS=EXCELCS;
   RANGE="Database"; SCANTEXT=YES; USEDATE=YES; SCANTIME=YES; RUN;
```

File manage.sas:

```
LIBNAME PROJECT "sasloc/currentsubdir";
/* Sort the seta data ****************************************** */
PROC SORT DATA=PROJECT.SETA;BY GROUP ID;RUN;
/* Sort the setb data ****************************************** */
PROC SORT DATA=PROJECT.SETB;BY GROUP ID;RUN;
```

```
/* Create the PROJECT.MAINSET dataset ************************* */
DATA PROJECT.MAINSET; MERGE PROJECT.SETA PROJECT.SETB; BY GROUP ID; RUN;
```

File analyze.sas:

```
LIBNAME PROJECT "sasloc/currentsubdir";
/* Compute summary statistics ******************************** */
PROC MEAN DATA=PROJECT.MAINSET; VAR A B C; BY GROUP; RUN;
```

The SAS® `LIBNAME` statement is needed in all files. Once these are defined, each may be run, by processing each file. Once finished, they need not be run again. This is a relatively simple approach, as each file is run until they work correctly. The repeated `LIBNAME` is a problem, as it needs to be changed in all files if the files are moved. In addition, each file must read in data from a permanent SAS® dataset, and write data to a permanent SAS® dataset.

- **Simple workflow manager:** The separate files defined above can be inserted into a workflow manager via the `%include "file";` mechanism. This program looks like this:

```
LIBNAME PROJECT "sasloc/currentsubdir";
/* Read in the data using readin.sas ************************** */
%include "readin.sas";
/* Perform necessary management tasks ************************* */
%include "manage.sas";
/* Compute summary statistics ******************************** */
%include "analyze.sas";
```

Each of the files would be modified by removing the `LIBNAME` statement. The file above, the workflow manager, would be a fourth file. This approach is simple, as the `%include` statements can be commented out. However, the code in each file is hard to keep in mind, and a number of source files might need to be open at once.

- **Macro-controlled workflow manager:** A sophisticated and flexible approach lies in using a macro-controlled workflow manager (MCWM). This approach looks like this:

```
%macro workflow(setup=0, readin=0, manage=0, analyze=0);
%if (&setup) %then %do; /* Setup libraries and prepare ********* */
LIBNAME PROJECT "sasloc/currentsubdir";
/* Read in the data from seta.xlsx *************************** */
PROC IMPORT OUT=PROJECT.SETA REPLACE DATAFILE= "seta.xlsx" DBMS=EXCELCS;
   RANGE="Database"; SCANTEXT=YES; USEDATE=YES; SCANTIME=YES; RUN;
/* Read in the data from setb.xlsx *************************** */
PROC IMPORT OUT=PROJECT.SETB REPLACE DATAFILE= "setb.xlsx" DBMS=EXCELCS;
   RANGE="Database"; SCANTEXT=YES; USEDATE=YES; SCANTIME=YES; RUN;
%end;
%if (&manage) %then %do; /* Manage data for project ************ */
/* Sort the seta data **************************************** */
PROC SORT DATA=PROJECT.SETA;BY GROUP ID;RUN;
/* Sort the setb data **************************************** */
PROC SORT DATA=PROJECT.SETB;BY GROUP ID;RUN;
/* Create the PROJECT.MAINSET dataset ************************* */
DATA PROJECT.MAINSET; MERGE PROJECT.SETA PROJECT.SETB; BY GROUP ID; RUN;
%end;
%if (&analyze) %then %do; /* Perform analysis ****************** */
/* Compute summary statistics ******************************** */
PROC MEAN DATA=PROJECT.MAINSET; VAR A B C; BY GROUP; RUN;
%end;
%mend workflow;
/* Workflow for Jones Project, 2013 *************************** */
%workflow(setup=0,readin=1,manage=1,anal=0)
```

The macro variables in the macro invocation are used to "turn a section on" or "off". Each macro variable is associated with a `%do-%end` group, which form a code block. The SAS® macro statement processor converts each macro variable into its value. If 0, the section is not executed, and if 1, the section is executed. This makes for a very easy approach to running sections.

The entire project is laid out in full complexity. The sections, delimited by the macro do-groups, define functional sections that are coherently stored, easy to understand, and simple to work with. Since the entire process is run inside a macro environment, the use of additional macro code to run multiple variables through specific analysis sections is very simple and natural. The comment in each section explain the process of the section. Additional analysis sections (`analysisa`) can be added, as can report-writing sections (`reporta`). New sections can be added. Some users do find this approach confusing, but it is not hard to use.

MANAGING PROJECTS

COMMON REQUIREMENTS FOR PROJECTS

Data analysis projects have common requirements, found in many projects.

Locations: Locations are needed for input files, input data, and output. The locations should not be hard-coded, but need to be identified in a flexible manner. To put an .rtf table somewhere, you can use:

```
ODS RTF FILE="c:\users\owner\Documents\sas\projecta\tab1.rtf";
```

If the location is changed for the output, the location will no longer be correct. It needs to be fixed in all location. In addition, if you have one table, you usually have 6, and possibly some figures as well. Thus, it is better to use code like this:

```
%let tabloc=c:\users\owner\Documents\sas\projecta;
ODS RTF FILE="&tabloc.\tab1.rtf";
```

The macro variable `tabloc` can be changed very easily if needed, and, if used consistently, all table locations will be updated immediately. Generally speaking, hard-coded directory locations should never be used, in any project.

Data sources: Data must be read into SAS® from different kinds of sources. The names and locations of data sources must be identified. Locations should be consistent from project to project.

Formats for categorical labels: Variables with discrete levels must be formatted for proper presentation, using the FORMAT procedure to properly identify values. Formats need to be defined in one and only one location.

Labels for variables: Variables must have a descriptive label. SAS® allows for long names currently, but labels have more flexibility, and every variable should have a proper label.

Output: Output from the analysis must be placed in a consistent location from project to project. Different forms of output need to go in defined locations. The ODS system in SAS® produces a variety of graphs automatically, which should go in a project-specific location.

Support for repetitive processes: The SAS® macro system is a good way to define code to perform complex repetitive processes. If a specific procedure is being used more than one time for an analysis, the macro system can define a flexible and simple approach to enhance processing and increase flexibility. Macros have considerable power, and can be used with control structures to process one variable in a manner slightly differently than another variable. Approaches for analysis need to facilitate and support the use of macro coding.

IMPLEMENTING REQUIREMENTS

These ideas can be implemented as follows:

Data sources: Standard approaches for data storage should be used. Each project should use a parallel method. Excel spreadsheets, raw data, and SAS® data should be stored in specific locations.

Formats for categorical labels: Formats should be placed in a central location. It is often good to alphabetically list formats in the PROC FORMAT specification. Additionally, for each separate format, an ordered listing of the options should be used. These practices, if maintained carefully, can make long-term maintainance of a project much easier. One good approach is to put the formats into an external file which is inserted into the main program via the %include mechanism:

```
%let fmtloc=w:\project\StatRes;
%include "&fmtloc\fmts.sas";
```

As discussed above, it is best to use a flexible location, defined using macro variables, because this can be changed simply if the location of the tree changes.

Labels for variables: Variable labels are an essential component of a well-defined SAS® program. They are also likely to be long, filled with plenty of opportunities for unmatched quotation marks, and essentially interfere with the readability of the program.

Output locations: Output for projects should be directed to parallel locations relative to the subdirectory root. This ensures that the tables and figures can be easily found, either by the original data analyst or by someone else after the author has become unavailable. Again, this location should be defined by a macro variable, for flexibility and maintainance simplicity.

Repetitive processes: With macros, repetitive processing can be set up in several ways, by having the iteration located in different places.

Projects structures: Projects need to have parallel structures. This ensures that, rather than devising a structure for a new project, each project simply uses a parallel structure. This simplifies the process of project setup, and ensures that similar things are found from one project to the next.

THE MCWM SYSTEM FOR PROJECT MANAGEMENT

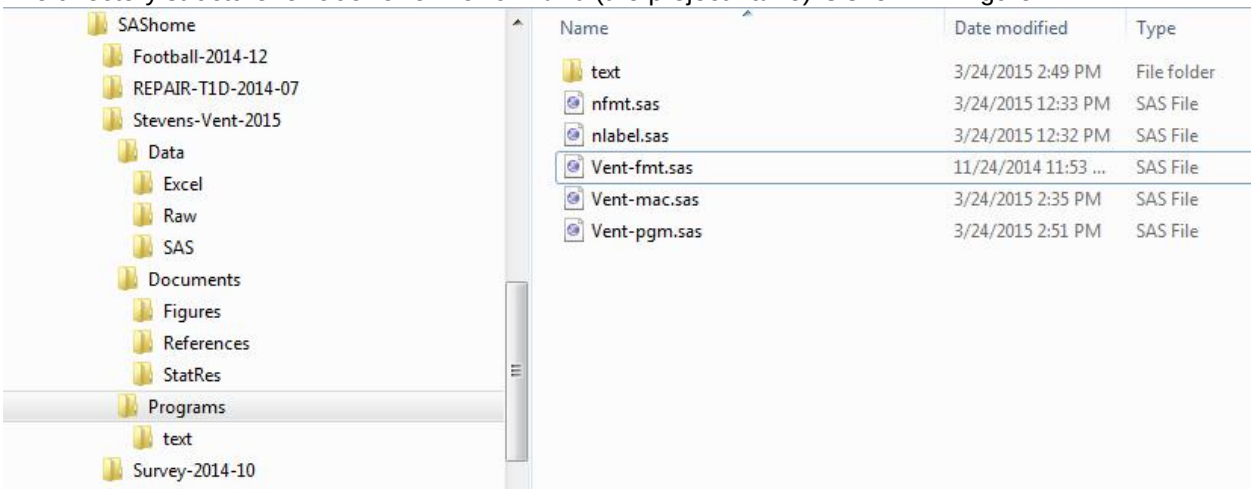
In the MCWM system, projects are each placed in a subdirectory structure. The main SAS® program is a macro shell which contains:

- a pre-defined set of links to all locations;
- macro-controlled library links;
- pre-set library designations for stored macros and for output templates;
- separate macro-controlled sections for import, cleaning, analysis, and reporting.

Current projects are placed under the `mainloc\current` directory. Previous projects are placed under the `mainloc\doneYYYY` location (where `YYYY` refers to the year). As a project is finished, it is moved from `mainloc\current` to `mainloc\doneYYYY`.

DIRECTORY STRUCTURE

The directory structure for `Stevens-Vent-2015` (the project name) is shown in Figure 1:



Error! Reference source not found.: Directory structure

Datasets are stored in the `Data` subdirectory, output goes to the `Documents` subdirectory, and programs are stored in the `Programs` subdirectory. The MCWM shell is `Vent-pgm.sas`, formats are stored in `Vent-fmt.sas`, and macros are stored in `Vent-mac.sas`. All projects have the same subdirectory structure.

Each project begins by copying z`Masterpattern, shown in Figure 2:`

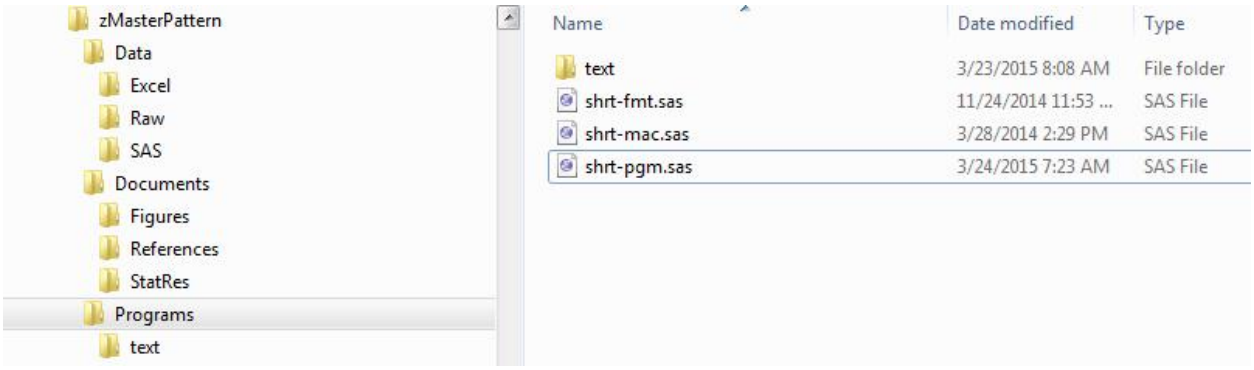


Figure 2: Master pattern for duplication

To set up a new project:

- Copy `zMasterpattern` using operating system utility
- Paste into `mainloc\current`. This will produce `zMasterpattern - Copy`
- Rename `zMasterpattern - Copy` to `Stevens-Vent-2015`
- Navigate to |Programs| and change `shrt-pgm.sas` to `Vent-pgm.sas` (same for `-mac` and `-fmt`)

THE MCWM SHELL PROGRAM FILE SHRT-PGM.SAS

The MCWM file contains the macro shell. The macro shell is somewhat long and involved. The basic structure is:

```
dm 'clear output;clear log;';options nosource mprint;
ods html close;ods html;

%macro mcwm(setup=0,readina=0,cleana=1,processa=0,reporta=0);
```

```
%if (&setup) %then %do;
The SETUP SECTION
%end;
The REMAINING SECTIONS
%mend mcwm;
/* ********************************************************** */
/* Overall analysis description ***************************** */
/* setup:    Setup for project ****************************** */
/* readina:  Defines Method A to read data ****************** */
/* cleana:   Clean and reprocess data *********************** */
/* processa: Process data - Anal.A: ************************* */
/* reporta:  Reporting mechanism - Anal.A ******************* */
/* ********************************************************** */
%mcwm(setup=0,readina=0,cleana=0,processa=0,reporta=0);
```

The shell is executed by running the macro program. If the macro is called as:

```
%mcwm(setup=1,readina=0,cleana=0,processa=0,reporta=0);
```

only the section controlled by the `setup` macro variable is run.

If the macro is called as:

```
%mcwm(setup=1,readina=1,cleana=1,processa=1,reporta=1);
```

the sections controlled by the `setup, readina, cleana, processa, and reporta` macro variable are run. Thus, selective execution of specific sections are easy to execute, and do not have the problems that other systems have.

THE SETUP SECTION

The `setup` macro controls the setup of code for the MCWM system. The code for this section is:

```
%if (&setup) %then %do;
/* Customize by setting the basedir value ******************* */
%global basev exceldata sasdata rawdata sasprog figdocs tabdocs;
/* Change Masterpattern to name of project subdirectory ****** */
%let basev=mainloc/current/MasterPattern;
%let sharesub=mainloc/Shared/Macros/Prod;
/* Change shortname to the shortname in the Programs subdir ** */
%let shrt=shortname;
/* Change yoursaslibnamehere to the preferred sas libname **** */
%let sasname=yoursaslibnamehere;

/* These are data locations ********************************** */
/* Change sasdir to an appropriate name ********************** */
%let exceldata=&basev/Data/Excel;
%let sasdata=&basev/Data/SAS;
%let rawdata=&basev/Data/Raw;
libname lib "&sasdata";

/* These are output locations ******************************** */
%let figdocs=&basev/Documents/Figures;
%let tabdocs=&basev/Documents/StatRes;
%let tmplt=systemroot/SASGTStore;

/* These are SAS code inclusions - include macro and format file */
%let SASprog=&basev/Programs;
```

```
%include "&SASprog/&shrt.fmt.sas";
%include "&SASprog/&shrt.mac.sas";

/* Bring in a stored macro file ****************************** */
LIBNAME macrlib "&sharesub";
options mstored sasmstore=macrlib;

/* Bring in group template store ***************************** */
ods path reset;
ods path (prepend)Group.templates(read);
ods path show;libname group "&tmplt";

/* Route statistical graphics to &figdocs ******************** */
ods html body="&shrt.log.html"
   path="&figdocs" gpath="&figdocs" image_dpi=600 style=statistical;
ods graphics on / reset=index imagefmt=png width=10cm height=7cm;
%end;
```

Some notes about this section:

- To fully customize the entire system for this project
  - Change `Masterpattern` to `Stevens-Vent-2015` in Line 5
  - Change `shortname` to `Vent-` in Line 8
  - Change `yoursaslibnamehere` to `vent` in Line 9
- All other changes are performed by the macro variables, and all immediately are done by the three changes noted above.
- Locations for data are stored in `exceldata, sasdata,` and `rawdata`. These macro variables contain code for the appropriate subdirectories.
- Location for figures is `figdocs`, tables is tabdocs. These point to the right place, and can be used in the same manner, but always point to the correct place in the subdirectory structure.
- The formats and macros are included using:
  ```
  %let SASprog=&basev/Programs;
  %include "&SASprog/&shrt.fmt.sas";
  %include "&SASprog/&shrt.mac.sas";
  ```
- A stored macro catalog is referenced using `options mstored sasmstore=macrlib;`
- A location for a template store for custom styles is given with:
  ```
  ods path reset;
  ods path (prepend)Group.templates(read);
  ods path show;libname group "&tmplt";
  ```
- All statistical graphics for the project are routed to figdocs. Thus, each project has its own location for statistical graphics defined by the shell.

THE REMAINING SECTIONS

The remaining parts of the MCWM file are:

```
/* Read in data ********************************************** */
%if (&readina) %then %do;
PROC IMPORT OUT=&sasname..d1 DATAFILE= "&exceldata/fnm.xlsx"
     DBMS=EXCELCS REPLACE;
     RANGE="Sheet1$A1:Z20000"; SCANTEXT=YES;
     USEDATE=YES; SCANTIME=YES;
RUN;
%end;
/* Clean data ************************************************ */
%if (&cleana) %then %do;
```

```
%end;
/* Perform analysis ******************************************* */
%if (&processa) %then %do;
%end;
/* Produce reports ******************************************** */
%if (&reporta) %then %do;
ODS RTF FILE="&tabdocs/output.rtf";
proc report data=ds nowd;
  column ;
  define var /
    order order=internal "" format=fmt.;
  define var /
    display "" format=fmt.;
run;quit;
ODS RTF CLOSE;
%end;
```

Some notes about this section:

- The IMPORT procedure code is stored in the `masterpattern` structure, and is available for easy customization. The name of the spreadsheet replaces `fnm`. The SAS® dataset name chosen by the user replaces `d1`. The Excel spreadsheet name replaces `Sheet1`. The range (if necessary) replaces the `A1:Z20000` specification, which may also be omitted. Once these modifications are made, PROC IMPORT will read the spreadsheet.
- The data management code is placed in the `%do` group associated with `cleana`.
- The data analysis code is place in the `%do` group associated with `processa`.
- The reporting code is placed in the `%do` group associated with `reporta`. In using the MCWM, the reports which are associated with `processa` should be placed in `reporta`. In that way, it is simple to find the code. The code block contains the shell of a REPORT procedure. This can be modified and altered. Since the `&tabdocs` macro reference is used, the PROC REPORT code is automatically placed into the correct location in the subdirectory structure.

USING AND CUSTOMIZING THE MCWM SHELL

The MCWM shell is designed to simply and easily set up projects, and provide a systematic way for an analyst to structure up, manage, and close out a project. There are several other points.

MODIFYING THE SHELL

Adding additional sections: It is simple to add additional `processx` and `reportx` components to the MCWM shell. They must be added to the macro definition statement, be incorporated into the body of the macro at an appropriate place, and be added to the macro invocation statement.

Reporting sections: It is good programming practice to prepare a permanent SAS® dataset to be the source of each table and figure. That way, the tables can be refreshed easily by running the `processx` and `reportx` sections. Alteratively, the `reportx` alone will rebuild the table.

Project close-out: Close-out of a project involves leaving on all items that are needed to correctly complete all tables, and adding comments to correctly indicate what each section does. In addition, the analyst should run the process one final time to ensure that all tables, figures, and incidental statistics are correctly defined, and that the MCWM is running properly.

Move to storage: Long-term storage of projects is done by moving the project to `mainloc\doneYYYY`. Once the project is moved to |past|, the various directory strings within the program dataset should be converted to the correct name, and the project again run to ensure that the correct tables are produced. Code in the SETUP SECTION must be modified. Previously, for the example project, we had:

```
%let basev=mainloc/current/Stevens-Vent-2015;
```

To fully set up for possible later runs of the program, this line is modified to:

```
%let basev=mainloc/done2015/Stevens-Vent-2015;
```

This fully changes the entire structure of linked macro variables, and the program can be run immediately in the long-term storage location.

MACROS IN THE MCWM SYSTEM

The SAS® macro system allows a variety of coding structures to be incorporated into SAS® programs. The advantage of the MCWM is that the macro environment is used for all processing. That means that specific actions are simpler, and require less preparation. In standard SAS® coding, the use of macro control structures such as the `%do-%end` structure require that a macro program be defined. Rather than setting up a macro to perform iterative processing of a list of items, the iterative process can be simply be used, as all processing is already in a macro program.This is a simplification of the macro system, and allows the user to incorporate macro control structures in a natural and almost automatic manner. It

The MCWM system also includes a file for macro storage. As the project is developed, code will be written. Once it is completed, it can be converted into a macro subprogram by adding `%macro macname;` and `%mend macname;`. It can then be placed into the file `&shrt.mac.sas`, and the call `%macname` placed in the MCWM shell file. This allows code under development to remain in the MCWM shell file, while finished code is safely stored in the macro file, available for use, but less able to be accidentally and inappropriately altered.

FORMATS IN THE MCWM SYSTEM

The MCWM approach provides a convenient location to store formats generated by the FORMAT procedure. Rather than formats being scattered around in a workflow program, the MCWM approach provides the `shrt-fmt.sas` file for formats. Placing formats here makes it easy to locate formats, manage formats, and share formats between projects. All formats are to be stored here. It is often the case the format code is placed in the MCWM shell file during development, but once development is done, the PROC FORMAT code is moved to the correct location.

IMPLEMENTATION OF THE MCWM SHELL AT SANFORD RESEARCH

The MCWM shell is stored with a variety of pre-set macros at Sanford Research. These include:

- Macro `descrtable`: Creates a table of summary statistics for continuous and discrete variables
- Macro `itemcnt`: Counts items in a macro list
- Macro `itemarray`: Creates a macro array from a macro list
- Macro `fmtrewr`: Rewrites a PROC FORMAT specification to list format names and values in an ordered list (alphabetically and numerically)
- Macro `buildfmt`: Creates a starting LABEL structure (which can be modified) and starting variable name format specification

There are a number of other variables which are stored in the macro catalog. Since the MCWM shell is copied to start a new project, these tools are available immediately for the user. It is unlikely that all SAS® users would be interested in the same macros that are of use in our situation. The advantage of the MCWM system is that each user can add the macro programs to the macro catalog at their site.

CONCLUSION

The MCWM (macro-controlled workflow manager) system is a flexible, easy-to-use SAS® environment which can be extensively customized, yet provides a basic tool which can provide standardized structures and locations for files in projects. It is easy to learn, easy to use, and provides a well-designed environment for reproducible research using SAS®. In addition, the MCWM environment can have local

modifications and local additions in the master source environment which enable such additions and modifications to be routinely present in all new projects. Local macros which are commonly used can be placed in the stored macro catalog. Local templates can be placed in the template store. Local macros can be added to the stored MCWM shell file, so that they are immediately available when the subdirectory copy process starts a new project. Using the MCWM system, read the main Excel file, and perform several custom operations in a very short time (20 minutes in some cases). The consistent structure of the MCWM makes it simple to find information in different projects, since all information is stored, relative to the subdirectory root, in the same location.

The importance of the MCWM system lies in the structure of the project environment. By providing a project environment with a very standardized parallel structure, all users who understand the system can easily find data, output, and code which created them. This is the essence of reproducibility. Computing users, data analysts, statisticians, and other SAS® users can benefit strongly from a system similar to, or identical to, the MCWM system.

REFERENCES

Thompson, P A and Burnett, A C. Reproducible Research. In: CORE Issues in Professional and Research Ethics. URL: http://goo.gl/hs7eb.

Peng, R D, Diggle, P J, and Zeger, S L. Reproducible research and Biostatistics. In: Biostatistics 10.3 (2009), pp. 405-408.

Claerbout, J F and Karrenbach, M. Electronic documents give reproducible research a new meaning. In: Proceedings of the 62nd Annual International Meeting of the Society of Exploration Geophysics (1992), pp. 601-604.

Fomel, S and Hennenfent, G. Reproducible computational experiments using SCons. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing 4 (2007), pp. 1257-1260.

CONTACT INFORMATION

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

Paul A. Thompson, Ph.D.
Sanford Research
605-312-6462
paul.thompson@sanfordhealth.org