SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

Be More Productive!
Tips and Tricks to Improve your SAS® Environment

Marje Fecht, Prowerk Consulting

#SASGF

Topics for Today's Presentation

Productivity tips and tricks to help you create a

standard and efficient environment for your SAS® work so you can minimize effort and maximize results.

- Setting up your programming environment
- Sharing easily with your team
- Dynamically managing files and results.

How do you start a new programming task?

Do you…

- Open up the editor and start typing – from scratch
- Find a similar task, copy the code, and start changing

- Write a game plan / outline of steps needed

- Open your standard template and provide the details of the new task

WORK SMART – Maintain Focus

What's the benefit of using a program template?

What do you gain?

- Improve Productivity – since less 'reinventing the wheel'
- Decrease maintenance – since commonalities

What does your team gain?

- Job-sharing / shadowing is easier ← *standards*
- Inherit *well-planned* and *well-documented processes*

What would you include in a program template?

- Comment blocks and Code Outline (game plan)
- Environment cleanup
- Easy movement between test and production
- Calls to “Tool Box” – Special purpose code modules
  - Required titles, footnotes, security
  - File handling, including permissions
  - Email notifications
  - Shared Formats and Macros
  - Report Templates
  - Date Handling (including Min and Max for data extracts)

Program Template: an example

```
/*************************************************************************************/
    RequestID - Purpose of Program - SKELETON CODE TO INCLUDE AS STARTING POINT
Requester:    Mary NeedsAnswers
Programmer:   Marje Fecht, Prowerk Consulting, www.prowerk.com
Request Date: February 23, 2016
Initial Code: March 1, 2016
Goals:        Provide strategy success metrics, including ……
Inputs:       REQUIRED
                - macro variables:
                    RequestID        - identifier of request (use for all files)
                    Automate (Y / N) - Use production ID or test ID
                    Filepath_Out     - location for output: LOG, Results, DATA
                    SASData_In       - location for SAS Data INput
                    SASData_Out      - location for SAS Data OUTput
                    Metadata_In      - location of metadata
                    DW_Passwd        - password strings for DW access

Outputs:      Reports:
              SAS Data:
              Data Files:
**************************************************************************************/
/******** M O D I F I C A T I O N    H I S T O R Y ****************************
When       Who       Changes

**************************************************************************************/

/** STEP 1:   DEFINE manual INPUTS **/
%let Request_ID   = MJFxxxyy;
%let filepath_out = /sas/&Request_ID./;     /*location for output: LOG, RESULTS, DATA*/

/** STEP 2:   CALL STARTUP PROCESSES **/
%include "insert call to startup program here" ;

. . . etc . . .

/** STEP FINAL:   CALL FINAL PROCESSES **/
%include "insert call to FINAL program here" ;
```

Program Template: Purpose and Contact Info

```
/*************************************************************************
RequestID - Purpose of Program

Requester:     Mary NeedsAnswers          (xxx)yyy-zzzz
Programmer:    Marje Fecht, Prowerk Consulting, www.prowerk.com
Request Date:  February 23, 2016
Initial Code:  March 1, 2016
Goals:         Provide strategy success metrics, including ……


Etc . . .
*************************************************************************/
```

Program Template: INPUTS

```
/*************************************************************
Inputs:        REQUIRED
- macro variables:
   RequestID          - identifier of request (for all files)
   Automate (Y / N)   - Use production ID or test ID
   Filepath_Out       - location for output: LOG, Results, DATA
   SASData_In         - location for SAS Data INput
   SASData_Out        - location for SAS Data OUTput
   Metadata_In        - location of metadata
   DW_Passwd          - password strings for DW access

Inputs:          REQUEST SPECIFIC

Etc . . .
*************************************************************/
```

Program Template: PLANNED OUTPUTS

```
/*************************************************************
Outputs:        Reports:     what?     Format?
                SAS Data:    Format?     Location?
                Data Files:
                  Format:         Location:
                Email Alerts:
                  Report Completion:
                  Error / Issues:
                  Data Availability:
Etc . . .
*************************************************************/
```

Program Template: Modification History

```
/***** M O D I F I C A T I O N    H I S T O R Y *****
When        Who        Changes
16Mar2016   Marje      Initial Development
13Apr2016   Marje      Added  etc etc
*****************************************************/
```

Program Template: Program Parameters

There are many approaches for input parameters.

- Macro variables, defined local to the program

```
%let Request_ID   = MJFxxxyy;
    /*location for output: LOG, RESULTS, DATA*/
%let filepath_out = /sas/&Request_ID./;
```

- External metadata that drives the program
  - » Normally you would use a single parameter (macro variable) that would read all of the appropriate metadata to customize the rest of your processes.

Program Template: Start-up Processing (Call)

What do you typically do at the beginning of every program?

- LOG routing including version control
- Clean-up (from any processes that ran prior)
- Userid and passwords depending on TEST / PROD environments
- Access permanent format and macro libraries.

Modularize your code by putting the above into a “Start-Up” module and use **%INCLUDE** to run this SAS program module.

```
/** STEP 2:   CALL STARTUP PROCESSES **/

%include "/&codeloc/standard_startup.sas";
```

Start-up Module: Example

```
** Create a macro variable with the current date and time
    to use for version control in log and output files **;

%let DateTime = %sysfunc(
                  compress(%sysfunc(today(),ymmddN8.)_
                  %sysfunc(time(),hhmm6.), ': ')
                        );

** route Log to permanent location and version control**;
proc printto  log =
  "&Filepath_Out.\logs\&stage.\&Strategy_ID._&datetime..log"
   ;
run; ** REMINDER: Turn PRINTTO OFF at end of process ! **;
```

Clean-up from previous work

If you are running a sequence of programs that are unrelated, you need to make sure that there are no carryover results that would impact the next program.

- Could macro variable values from the first program impact how later programs work?
  - If yes – DELETE the macro variables before submitting the next program.

- Could temporary datasets from the first program impact results from later programs?
  - Just to be safe, delete those too!

Code Module: Clean-up – Part 1

```
*************************************************;
*** clean up work datasets;
*************************************************;

*** clean up the work directory ***;
proc datasets lib=work
              nolist nowarn nodetails kill;
quit;
```

Code Module: Clean-up – Part 2

```
*** build list of all macro variables to clean up ***;
data _null_;
    length cmd $200;
    set sashelp.vmacro; /*Use SAS macro dictionary table*/
       /*Select all GLOBAL Macro variables w/ offset of zero*/
    where
          scope = 'GLOBAL'    and offset = 0
       and    /* exclude some macro variables from deletion */
         (name ne: 'SYSDB'  and  name ne: 'SYSODS'
            and  name ne: 'SYS_'     )  ;

       /*Cycle through list of macro variables and delete */
     cmd  = '%nrstr( %symdel '||trim(name)||' / nowarn ) ; ' ;
     call execute(cmd);
run;
```

Other Code Modules to Consider

- Standard titles / logos / footnotes
- File handling
- Format handling (libnames, fmtsearch…)
- Macro libraries or AUTOCALL Macros
- FINISH – file permissions

Code Module: FINISH – set file permissions

When your job completes, don't forget to handle file permissions, so that others in your team can access the files you have created!

```
   /** Unix  Example:  **/
x "chmod 777
   "&Filepath_Out.\logs\&stage.\&Strategy_ID._&datetime..log";

/** NOTE:
    Beware of CASE in your path and file name – unix cares **/
```

My Top 5 Suggestions

- Always write your code as **generically** as possible
- Use code **comments** liberally, including starting with a code **plan** (in the form of comments) and a **standard code template**
- Use a **phased** approach, which includes delivering important results quickly (for buy-in)
- **Participate** in meetings – do not become an *order taker* **-** work directly with the information consumer
- **THINK** before you type!

Conclusion

As the requests for results increase and the demand on our time also increases, we have to find ways to streamline our processes, and reduce start-up time.

WORK SMART – Easily Share and Standardize

This presentation has shared a variety of suggestions to improve your programming environment by standardizing and modularizing the tasks you typically need.

About the Presenter

Marje Fecht is a Senior Partner with Prowerk Consulting, and has enjoyed using and teaching SAS software since 1979.

She is passionate about developing efficient, reusable methodologies so that businesses can focus on insight instead of processes.

Her role as SAS Global Forum 2014 Conference Chair provided a wonderful opportunity to collaborate with SAS and the SAS community.

http://www.sas.com/en_us/customers/customer-recognition/30-years/marje-fecht.html