



Are Your SAS® Programs Running You?



Marje Fecht

Senior Partner

Prowerk Consulting

Larry Stewart

Vice President, Education

SAS Institute

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are registered trademarks or Trademarks of their respective companies.

Copyright © 2010 . Prowerk Consulting Ltd. All rights reserved.

Overview

- Have you ever QUICKLY written some code assuming it will **never** be used again??
- Is it now 5 years later and the SPAGHETTI CODE is **still** in production?
- Worse still – does the code require YOUR time to tweak every time you run it?



This tutorial focuses on maintenance – free, efficient coding techniques so that you can spend your work time being more productive.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Topics

- Part 1: Reduce your coding time and run time!
-Remove Inefficient and "Wordy" Coding
- Part 2: Reduce Programmer Intervention
-Use Macro Variables to avoid repetitive changes
-Write Code that "thinks for itself"
- Part 3: Speed up Delivery Time
*-Focus on **reusable code modules** so you don't play the "copy and paste" game*
- Part 4: Testing Tip

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Part 1: Reduce your coding time and run time!

Business Task:

You need to generate a report that requires

- subsetting a SAS data set
- sorting the resulting data set.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Does this look familiar?

Subset the data prior to sorting

```
data compare;  
  set report;  
  if ProductCode in ('XER', 'REF', 'CRS')  
    and Date gt '01MAY2010'd;  
  keep ProductCode Usage Date Location;  
run;  
proc sort data=compare;  
  by ProductCode Date;  
run;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Remove Inefficient Coding

Problems

- All observations in the REPORT data set are read but only observations with specific ProductCode and Date values are required
 - Use WHERE instead
- Data set is read and written at least twice
 - Use one step instead of two

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Remove Inefficient Coding

Subset and sort the data in one step

```
proc sort data=report
  (keep=ProductCode Usage Date Location)
  out=Compare
  ;
  where ProductCode in ('XER','REF','CRS')
    and Date gt '01MAY2010'd ;
  by ProductCode Date;
run;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Business Task

You need to

- compute and compare revenue figures by day of the week.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Does this look familiar?

```
data work.revenue;  
  set sug.sales;  
  DayOfWeek=weekday(date);  
run;  
proc format;  
  value DayWk 1='Sunday'  
              2='Monday'  
              3='Tuesday'  
              4='Wednesday'  
              5='Thursday'  
              6='Friday'  
              7='Saturday';  
run;  
proc tabulate data=work.revenue;  
  class DayOfweek;  
  var Revenue;  
  tables DayOfWeek , Revenue;  
  format DayOfWeek DayWk.;  
run;
```

Compute
total
revenue by
Day of Week

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Does this look familiar?

Step 1: Create a variable that contains the day of the week.

```
data work.revenue;  
  set sug.sales;  
  DayOfWeek=weekday(Date);  
run;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Does this look familiar?

Step 2: Create a format that substitutes the name of the day for the day number.

```
proc format;  
  value DayWk 1='Sunday'  
              2='Monday'  
              3='Tuesday'  
              4='Wednesday'  
              5='Thursday'  
              6='Friday'  
              7='Saturday';  
run;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Does this look familiar?

Step 3: Use the format when computing the totals.

```
proc tabulate data=work.revenue;  
  class DayOfweek;  
  var Revenue;  
  tables DayOfWeek , Revenue;  
  format DayOfWeek DayWk.;  
run;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Reduce Inefficient Coding

Problems

- Multiple steps when one is enough
- Creating a format when one already exists

➤ Use **WEEKDATE9.** format

Original Date	: 14MAY2007
SAS Date	: 17300
WEEKDATE Format	: Monday, May 14, 2007
WEEKDATE9 Format	: Monday
EURDFWKX9 Format:	Lundi (dflang=French)

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Reduce Inefficient Coding

Using

```
options dflang=french;
```

Original Date	: 11Nov2008
SAS Date	: 17847
EURDFWDX Format	: 11 novembre 2008
EURDFWDX9 Format	: novembre
EURDFWKX Format	: Mardi 11 novembre 2008
EURDFWKX9 Format:	Mardi

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Reduce Inefficient Coding

Solution: Use WEEKDATE9. format

```
proc tabulate data=work.revenue;  
  class Date;  
  var Revenue;  
  tables Date , Revenue;  
  format Date weekdate9. ;  
  
run;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Part 2: Reduce Programmer Intervention

Some programs are written **too specifically** to the task, and require constant intervention to specify

- dates / timeframes
- titles
- inclusion criteria.

In extreme cases, this means scanning 1000's of lines of code.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Part 2: Reduce Programmer Intervention

Business Task:

You need to

- run a daily report that compares current month to date revenue with prior month revenue.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Does this look familiar?

Solution 1:

```
data currentmonth;
  set sug.sales;
  if month(Date)=5 and year(Date)=2010
  then do;
    MonthYear = " 5/2010";
    output;
  end;
  else if month(Date)=4 and year(Date)=2010
  then do;
    MonthYear = " 4/2010";
    output;
  end;
run;
proc means data=comparemonths;
  class MonthYear;
  var Revenue;
  title "MTD Revenue vs Last Month";
run;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Reduce Intervention and Remove Inefficient Coding

Problems

- All Observations are read when only current and most recent months needed
 - Use WHERE for selecting data

- Monthly intervention required to change values
 - Generalize the program so it is data driven by using macro variables and SAS functions

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Remove Intervention

Solution 2:

Use SAS functions to make date decisions

```
data comparemonths;
  set sug.sales;
  if month(date) = month(today()) and
     year(date) = year(today()) then do;
    MonthYear = put(today() , mmyyS7.);
    output;
  end;
  else do;
    lastmonth=intnx('MONTH' , today() , -1);
    if month(date) = month(lastmonth) and
       year(date) = year(lastmonth) then do;
      MonthYear = put(lastmonth , mmyyS7.);
      output;
    end;
  end;
run; *** proc means step unchanged;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Reduce Intervention and Remove Inefficient Coding

Problems

- Repeated FUNCTION calls (Today(), Month, Year) when one call would suffice

➤ Use `_N_ = 1` or Macro Variables or Both

- If program runs across *Midnight*, three *partial* months of data will result

➤ Compute Today() one time!

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Solution 3: FOCUS on Key components

```
** ONE TIME computation of Today's Date  
   at time program starts  
   - at top of program before DATA step  
;
```

```
%let DateToday = %sysfunc(today());  
   /* Macro Variable contains a  
   /* sas date value  
   */
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Solution 3: FOCUS on Key components

```
< SNIP - PART OF DATA STEP >
  if _n_ = 1 then do;
    *** current month;
    mon_today = month(&DateToday);
    yr_today = year(&DateToday);
    MonthYear_today
      = put(&DateToday , mmyyS7.);

    *** last month;
    lastmonth
      =intnx('MONTH',&DateToday, -1);
    mon_lastmonth = month(lastmonth);
    yr_lastmonth = year(lastmonth);
    MonthYear_lastmonth
      =put(lastmonth,mmyyS7.);
  end;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Solution 3: Reduce function calls and handle "over midnight" runtime

```
*** PART ONE OF SOLUTION 3 *** ;
** ONE TIME computation of Date at time program starts;
%let DateToday = %sysfunc(today()); /* sas date value */

data comparemonths;
  set sug.sales;
  *** create static values - first time through the DATA step;
  retain mon_today yr_today MonthYear_today lastmonth
         mon_lastmonth yr_lastmonth MonthYear_lastmonth ;
  if _n_ = 1 then do;
    *** current month;
    mon_today = month(&DateToday);
    yr_today = year(&DateToday);
    MonthYear_today = put(&DateToday , mmyyS7.);
    *** last month;
    lastmonth=intnx('MONTH',&DateToday, -1);
    mon_lastmonth = month(lastmonth);
    yr_lastmonth = year(lastmonth);
    MonthYear_lastmonth =put(lastmonth,mmyyS7.);
  end;
  *** CONTINUED ***;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Solution 3: continued . . .

```
*** PART TWO OF SOLUTION 3 *** ;

*** DATA STEP CONTINUED ***;

*** Process Current Month Revenue;
if month(date)= mon_today and year(date)= yr_today then do;
    MonthYear= MonthYear_today;
    output;
end;

*** Process Last Month Revenue;
else do;
    if month(date)= mon_lastmonth and year(date)= yr_lastmonth
then do;
    MonthYear = MonthYear_lastmonth;
    output;
    end;
end;
run;

*** proc means step unchanged;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Two final suggestions

Consider two final improvements:

- Since only two months of data are required, consider a **WHERE** in the data step to reduce the amount of data processed
- Consider creating the 7 variables (at the top of the data step) as **macro variables** – so that you can use them throughout the program.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Part 3: Speed up Delivery Time

Business Task: You need to use the **same query logic** and **reporting formats** on a regular basis for each new product / campaign / region.

The program is **always** the same, except for changes to

- date ranges
- inclusion codes
- titles and footnotes
- search logic.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Does this look familiar?

- You make a copy of the **most recent version you can find** of the query / reporting program
- You step through the program and make all the changes that apply for your current use
- You run the program and... OOPS... *you must have over-typed some quotes and semicolons*
- You apply corrections, and try again.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Reduce Intervention

Problems

- It is difficult to locate the most current version of the query/reporting program
 - Store a single dated copy of the main program / Modules
- Updating values in the program often leads to errors, or missed parameters
 - Use a driver program with macro variables as input and call the *Modules*

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Reduce Intervention

Solution: A driver program specifies parameters and runs the standard source programs.

```
*** Driver Program - specify correct parameters;  
  
%let prestart = 01OCT2009;  
%let prestop = 31DEC2009;  
%let poststart = 01JAN2010;  
%let poststop = 30JUN2010;  
%let title = "January 2010 Introduction of Low  
Interest Rates";  
%let products = ('VRS', 'GQL', 'BGO', 'DLA');  
%let codes = ('015', '119', '214');  
  
*** run standard reporting programs;  
%include 'CampaignReportExtract_20100110.sas';  
%include 'CampaignReportOutput_20100113.sas';
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Focus on Production Code!

Only one problem remains...

when the source code changes, you have to find ALL the drivers that call it to change the version.

Solution:

Create a program that calls the latest source!

CampaignReportExtract_ **CurrentPgm**.sas

```
*** call latest version of source program;  
%include 'CampaignReportExtract_20100110.sas';
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Focus on Production Code!

When source changes, your drivers always call the **most current version** 😊 .

```
*** Driver Program - specify correct parameters;  
  
%let prestart = 01OCT2009;  
%let prestop = 31DEC2009;  
%let poststart = 01JAN2010;  
%let poststop = 30JUN2010;  
%let title = "January 2010 Introduction of Low  
Interest Rates";  
%let products = ('VRS', 'GQL', 'BGO', 'DLA');  
%let codes = ('015', '119', '214');  
  
*** run standard reporting programs;  
%include 'CampaignReportExtract_CurrentPgm.sas';  
%include 'CampaignReportOutput_CurrentPgm.sas';
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Reduce Intervention – Success!

The Old Way – Hours to locate / QA / etc

The New Way – Campaign reporting is ready with less than ½ hour of effort

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Part 4: Testing Tip

Remember: COMPARE the *before and after* data sets to confirm comparability of solutions!

```
*** Useful to compare 2 solutions;  
proc compare data=sales compare=sales2;  
run;
```

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Conclusion

As you learn more about SAS, you will find that there are numerous features that enable you to accomplish tasks easily and efficiently.

In this presentation, you have seen

- tips to reduce how many times data is processed
- code reduction techniques
- suggestions to reduce or remove manual intervention to enable “maintenance-free” jobs.

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.

Thank You!



TM
Marje Fecht
marje.fecht@prowerk.com

Senior Partner
Prowerk Consulting

Copyright © 2008, Prowerk Consulting LLC. All rights reserved.