

/*****

A SAS-Based Six-Sigma Assessment Routine

Bryan K. Beverly,
BAE Systems Information Technology

This program determines if an application or code segment is 'six-sigma compliant'. Six-sigma is a statistical concept that assesses the quality of a process by counting the number of defects. Achieving a six-sigma rating means your processes are producing no more than 3.4 errors for every one million opportunities. A six-sigma rating for a process suggests that the process almost never fails.

Please Note:

- o This program is called by an external batch program because executing it directly inside of a SAS session would overflow the LOG window with processing messages. Therefore one must create a batch program (using the following code "**C:\Program Files\SAS Institute\SAS\V8\sas.exe**" **-sysin c:\six_sigma.sas -nosplash -icon -nolog**) in addition to the main testing program.
- o This application should be executed on a stand-alone computer because performing one million iterations is time-consuming and resource-intensive.
- o It should only be used for applications where a process could potentially produce a different outcome with each iteration (such as testing a SAS/Connect session, a random number application or any data or proc step where the inputs would always be different).

The example below took 26 hours to execute one million times. For this demonstration, the inputs were static; hence the outcome was the same for each iteration. The &SYSERR values were used to increment the counter. When implementing this code, one may choose to allow &SYSERR values other than '0' to determine whether the error counter is incremented.

The take-away from this example is that one can easily create a SAS implementation of industry-accepted quality standards to determine the consistency of SAS programs or code segments.

*****/

```
%macro six_sigma;
```

```
  /*****  
  /* STEP 1 - Initialize the environment and the counter */  
  *****/
```

```
options nosource noxwait noxsync xmin ;
```

```
filename _all_ clear;  
libname _all_ clear;
```

```
proc datasets kill nolist;  
run;  
quit;
```

```
%let counter=0;
```

```

/*****/
/* STEP 2 - Embed or include the test code, trap for */
/*          errors after the data or proc steps, and */
/*          increment the counter with each iteration */
/*****/

%macro run_app;
  data one;
    x=1;
  run;

  %if &syserr^=0 %then %do;
    %let counter=%eval(&counter + 1);
  %end;

  data two;
    y=2;
  run cancel;

  %if &syserr^=0 %then %do;
    %let counter=%eval(&counter + 1);
  %end;
%mend run_app;

/*****/
/* STEP 3 - Execute the code one million times */
/*****/

%macro calc_ss;
  %do i=1 %to 1000000;
    %run_app;
  %end;
%mend calc_ss;
%calc_ss;

/*****/
/* STEP 4 - Calculate the error percentage */
/*****/

data ss_eval;
  length status $ 30;
  errorcnt=input("&counter",12.0);
  errorpct=(errorcnt/1000000);

  if errorpct<=.0000034 then do;
    status='Six-Sigma Compliant';
  end;

  else do;
    status='Not Six-Sigma Compliant';
  end;
  format errorpct 12.9;
run;

/*****/
/* STEP 5 - Produce a status report */
/*****/

```

```
proc print data=ss_eval;  
  title 'Six-Sigma Assessment Report';  
run;  
%mend six_sigma;  
  
%six_sigma;
```