

A Modular SAS Programming to Support ABC™ Simulation Study

Sejong Bae, University of Alabama at Birmingham, Birmingham, AL
Michael Weaver, University of Alabama at Birmingham, Birmingham, AL

ABSTRACT

A huge program composed of one long line can be divided into a number of sub-programs. A modular programming technique has been adapted to study the properties of ABC™ under various simulated conditions. A modular programming technique has several advantages over one large program. It provides easier coding, debugging, and plug-in use to various situations, ABC™ is a tool to facilitate the measurement, comparison and dissemination of high levels of performance derived from the process of care practices already being achieved by “best-in-class” providers.

INTRODUCTION

A long line of program is hard to debug and program. Using several macros makes the programming and the debugging a lot easier. The modular programming approach was taken to support the ABC™ simulation study.

Benchmarking is the process for identifying medical industry leaders in order to emulate their process. The traditional Continuous Quality Improvement approaches do not include tools that provide a uniform, data-driven definition of benchmark performance. The ABC (Achievable Benchmarks of Care) is designed to utilize the tools for use with process of care indicators.

ALGORITHM FOR ABC SIMULATION STUDY

Main Module :
Calls the sub-modules.

Sub-module :
Creates the data set for given simulation conditions, calls the ABC module, and outputs the Benchmark and actual performance to a file.

ABC Module:
For a given data set,
Calculates the Bayes adjusted performance, where the Bayes adjusted performance is calculated as

(Treatment given + 1) over (Treatment eligible + 2)
 $= (n_i + 1) / (d_i + 2)$.

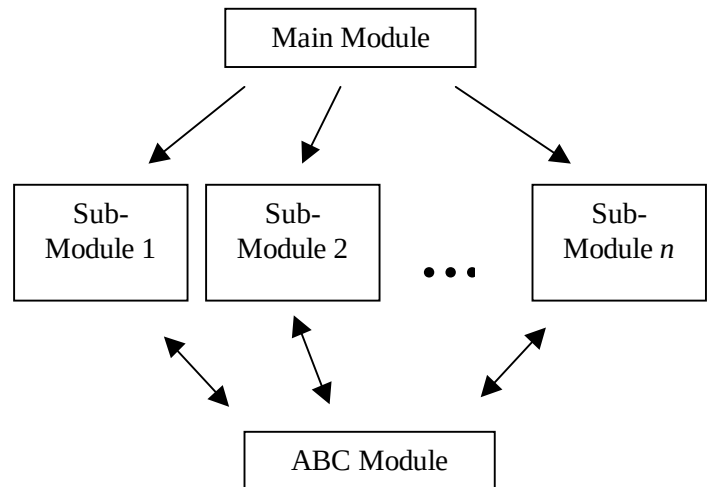
Finds the top 10% Bayes adjusted performance + any tied performance.

Based on the above observations, benchmark is calculated as

$$\frac{\sum_i n_i}{\sum_i d_i}$$

Where n_i is the numerator, which is the number of eligible cases of treatment, and d_i is the denominator, which is the number of treatments given.

The above algorithm can be shown as the following diagram of $n + 2$ separate programs.



Without modular approach, the program could have been in a very long one line of coding that is hard to debug and read.

From the main module, the sub-modules have been called

```
%include 'e:\ABCSIM\RSMDS101.SAS';  
%include 'e:\ABCSIM\RSMDS102.SAS';  
.  
.  
.  
%include 'e:\ABCSIM\RSMDS10n.SAS';
```

From each sub-module, the following ABC module has been called several times under various conditions of the data set.

ABC MACRO MODULE

```
%macro ABC(datain, dataout1, dataout2, dataout3);
data a;
set &datain;
if adenom=0 then aperf=0;
else aperf=anumer/adenom;
prov=hsp + 0;
temp = 1;
bdenom = 2 + adenom;
bnum = 1 + anumer;
bperf = bnum / bdenom;
```

```
proc means data=a noprint sum;
var anumer adenom;
output out=temp sum=tanumer tadenom;
by temp;
run;
```

```
data a;
merge a temp;
by temp;
label
tanumer = 'total actual numerator'
tadenom = 'total actual denominator';
```

```
proc rank data = a out=b ties = mean descending;
var bperf;
ranks rbperf;
run;
```

```
proc sort data = b; by rbperf tadenom; run;
data b; set b;
retain sumden sumnum 0;
sumden = sumden + adenom;
sumnum= sumnum+anumer;
targer = .1*tadenom;
label
sumden = 'sum actual denom'
sumnum= 'sum actual num'
targer = 'targer 10% actual denom';
```

*The following IML selects the data based on 10% (target) and further includes all that had the same rank as the target;

```
proc iml;
```

```
use b;
read all var {anumer adenom rbperf sumden target
prov} into bm;
flag1=0; flag2=0;
i=1;
do until (flag1=1 & flag2=1);
if bm[i,4] >= bm[i,5] then
do; flag=1;
if bm[i,3] ^= bm[i+1,3] then do;
flag2=1;
do j=i+1 to nrow(bm);
bm[j,5]=0;
end;
end;
end;
i=i+1;
end;
```

```
varnames={ 'anumer' 'adenom' 'rbperf' 'sumden'
'target' 'hosptl' };
create c from bm{colname=varnames};
append from bm;
close c;
quit;
```

```
data c;
set c;
if target = 0 then delete;
```

```
*output the Above benchmark hospitals;
data & dataout3;
set c;
*the following IML calculates the ABC from the
selected partial data;
proc iml;
use c;
read all var {anumer adenom} into abcmx;
```

```
benchnum=abcmx[,1];
benchden=abcmx[,2];
bnum=sum(benchnum);
bden=sum(benchden);
bmark=bnum/bden;
```

```
varnames={ 'bmark' };
create &dataout1 from bmark[colname=varnames];
append from bmark;
close &dataout1;
quit;
```

```
data &dataout2;
```

```
set temp;  
aperf=tanumer/tadenom;  
keep aperf;  
%mend ABC;
```

SUMMARY

Having a separate ABC module, that is well programmed and debugged, the module can be accessed from any part of the simulation study as long as the data set is generated. Using this approach, we did avoid one single long line of coding and provided more efficient way of programming.

Author Information

Sejong Bae
505 Medical Towers, 1717 11th Ave South.
Birmingham, AL 35294

205.934.0775 Fax: 205.934.0777

E-mail: bsejong@uab.edu

Michael Weaver
School of Nursing, Graduate Studies
University of Alabama at Birmingham
Birmingham, AL 35294

205.934.6913 Fax 205.975.6142

E-mail weaverm@uab.edu

ABCTM is the University of Alabama at
Birmingham's Achievable Benchmark of Care
SAS[®] is a registered trademark of SAS institute