PharmaSUG2011 - Paper PO13

Array of hope: Using SAS Arrays to produce data-driven analysis and reports.

Aida Likaj, PPD Inc., Austin, TX

ABSTRACT

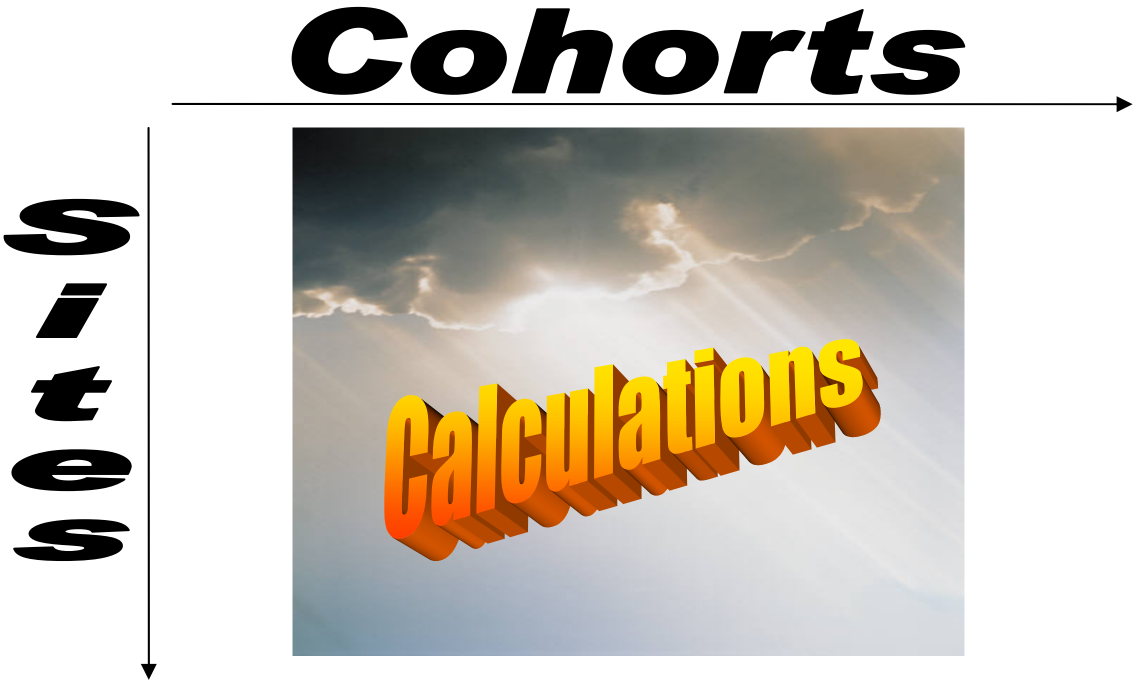
Programmers sometimes need to program reports when the complete range of data values is unknown at the time of the request. For example, the number of treatment cohorts may increase throughout the life of the study. This paper will show how to use arrays to produce results dynamically, minimizing the need for updates in statistical calculation and reporting regardless of the number of unique cohorts.

INTRODUCTION

The purpose of this paper is to show SAS® users some programming techniques that can minimize their work, regardless of the state of the data. If data is static, any calculation needing to be made on finalized data is simple. But what if data is changing day to day? In a clinical trial study, is very possible that as a study progresses the number of locations where patients are enrolled and the quantity of the drug they are receiving is changing at the same time. This can continue until the number of patients forecasted in the protocol for a particular drug cohort, reaches the desired statistical requirement. How can we write code early in the study to be robust and with no need for update when the study is ended? For this presentation two keywords are going to be used that we define as:

Site – a number assigned to a particular geographical location

Cohort - a group of persons sharing a particular statistical or demographic characteristic, in our case being on the same drug amount for the same time period.



GETTING MAX OF COHORTS

To meet the challenge, SAS provides multiple tools that make magic when used together. The most useful for the scenario described above are going to be arrays and macro variables. These are going to allow the dynamic calculation of sites and cohorts. First the maximum number of cohorts at the time program is run is needed. As this number is going to be referred to during the whole process, it is easier to assign it in a macro variable. This needs to be implemented only on the data where both site and cohort are present.

The following code reads in the data and through the use of “into:” in proc sql produces macro variables that hold the number of cohorts, the number of patients per cohort, and the overall number of patients The values in these macro variables will change throughout the life of the study and will be called later in the program in arrays that set the dimensions of the outputs.

```
%global n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 max_cohort;
data p_&progname.1;
     set derive.header(keep =pt site cohort);
     where (~missing(cohort) and ~missing(site));
     format _all_;
run;

proc sql noprint;
     select count(distinct cohort) into: max_cohort
       from p_&progname.1;

     select count(distinct pt) into: n1-:n&max_cohort
          from p_&progname.1
          group by cohort;

     select count(distinct pt) into: n%eval(&max_cohort+1)
          from p_&progname.1;

     create table cohort_SITE as
          select site, count(distinct pt) as cnt, cohort
          from p_&progname.1
          group by site, cohort;

    create table cohort_total as
          select count(distinct pt) as cnt, cohort
          from p_&progname.1
          group by cohort;

    create table SITE_TOTAL as
          select site, count(distinct pt) as total
          from p_&progname.1
          group by site;
quit;
```

PREPARING THE CONTAINER

After transforming and merging tables cohort_site and cohort_total by site, the new table provides the information used to create the table of site x cohorts populated with the number of patients for each cell of the table. Figure 1 shows the result:

INVSITE	_1	_2	_3	_4	_5	_6	_7	_8	_9
01	2	2	1	4	2	1	2	4	18
02	1	2	2	2	1	2	1	4	15
Total	3	4	3	6	3	3	3	8	33

Figure 1.

If the requirement was just to know the number of patients for every combination than we are done. Let's assume that the actual calculation that we need to perform on each cell is also the percentage that this number represents for that particular cohort. To perform this calculation the arrays are very useful.

PERFORMING THE CALCULATION

There are going to be 3 arrays that will perform the required calculations. All of them will have the same dimension; defining this dimension will be easy using the code provided above.

Array a will hold the number of patients for each cohort for the site where the calculation is being performed. If you see figure 1, think of the array as a pointer for every individual site.

Array b will hold the total number of patients for each individual cohort. It should be noted that information on array b will be the same for all the records. In order to assign this information dynamically, the data step will be included in a macro call to facilitate the do loop. This assignment will require a double && reference, as this information is assigned in individual macro variables. Assignment of these macros is already performed in the code above. This is the only part of the code that programmer has to make an estimate in order to have all the needed macro variables. More is better so create a couple more than what a statistician might estimate for the study.

Array c will hold the final result, concatenation of patient number and percentage that this represents.

```
%let dim_cohort=%eval(&max_cohort+1);
%Macro Calculate;
data &progname(keep=col1 - col&max_cohort TOTAL);
     retain col1 - col&dim_cohort TOTAL;
     set &progname.02;

     array a(*)  _1 - _&dim_cohort;
     array b(&dim_cohort);
     array c(*) $25 col1 - col&dim_cohort;

     %do i = 1 %to &dim_cohort;
          b(&i)= &&N&i.;
     %end;

     do i = 1 to dim(a);
          if a(i)=. then a(i)=0;
          c(i)=trim(put(a(i),9.));
          if  a(i) > 0 then  c(i)= a(i)*100/b(i) ;
      end;

     TOTAL=col&dim_cohort;
run;
%mend;
%Calculate;
```

In conclusion, in order to have dynamic code, a combination of macro variables and arrays is very useful. This approach makes it possible to create code that will not require any updates regardless of the status of the data.

The complete code and output created from it is included in files Array of hope - Code.doc and Array of hope - Output.doc.

ACKNOWLEDGMENTS

A big thank you to Jeanina Worden and John Gorden for helping me out with my first poster.

CONTACT INFORMATION (HEADER 1)

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

Name: Aida Likaj
Enterprise: PPD
Address: 7551 Metro Center Dr. Suite 101.01,
Austin, Texas, 78744

Work Phone: 512-747-5839
Fax: 512-747-9914
E-mail: Aida.Likaj@ppdi.com
Web: