Paper 200-2010

Facilitating Genetic Analysis: SAS®, the NHLBI and CARe

Taylor Young for the CARe Project, The Broad Institute, Cambridge, MA

ABSTRACT

Two common tasks for a SAS programmer are creating summary statistics and distributing data. Often, these are relatively benign tasks requiring only a handful of SAS statements but on the occasions when a large volume of data must be processed, writing too many of these statements by hand becomes inefficient and ultimately error prone. The purpose of this paper is to describe several techniques for summarizing and distributing a large number of SAS data sets and variables. The methods are discussed within the context of the National Heart Lung Blood and Sleep Institute's (NHLBI) Candidate Gene Association Resource (CARe) project which assembled genotype and phenotype data from more than 40,000 individuals across 9 longitudinal, epidemiological cohorts (Musunuru et al. 2010). To date, 862 SAS data sets encompassing 53,428 distinct variables have been made available to a community of several hundred CARe investigators.

INTRODUCTION

Two common tasks for a SAS programmer are creating summary statistics and distributing data. Often, these are relatively benign tasks requiring only a handful of SAS statements but on the occasions when a large volume of data must be processed, writing too many of these statements by hand becomes inefficient and ultimately error prone. The purpose of this paper is to describe several techniques for summarizing and distributing a large number of SAS data sets and variables. The methods are discussed within the context of the National Heart Lung Blood and Sleep Institute's (NHLBI) Candidate Gene Association Resource (CARe) project which assembled genotype and phenotype data from more than 40,000 individuals across 9 longitudinal, epidemiological cohorts (Musunuru et al. 2010). To date, 862 SAS data sets encompassing 53,428 distinct variables have been made available to a community of several hundred CARe investigators. The SAS methods used for the summarization, automation, and management of these genetic data follow.

SUMMARIZING DATA

Due to the sheer number of variables, summary statistics must be generated by the use of a decision method base on the following criteria. Although the FREQ and MEANS procedures are standard tools in this process, and SAS maintains a dichotomy of numeric and character variables, it is not advisable to simply run PROC MEANS on all numeric variables and PROC FREQ on all character variables because clinical data is often a mix of at least 4 different variable types:

1. **Numeric and continuous** – Measurements like height, weight, age, and lab values are almost always stored as continuous numeric values meaning that any numeric response is acceptable. PROC MEANS is the best method for summarizing these types of variables.
2. **Numeric and discrete** – Often times, categorical data such as gender, race, ethnicity, and yes/no/unknown responses are stored as discretely coded numeric values where the number is only a representation of the actual response. Acceptable responses for these types of variables are usually limited to a small range of integers and PROC FREQ is the best method for summarizing numeric and discrete variables. An exception would be ICD9, MedDRA, or WHO Drug Dictionary codes, which may appear to be continuous because of the number of distinct values but are in fact discrete.
3. **Character and discrete** – These are similar to numeric and discrete variables except that the actual response is stored as text rather than a numeric code. Likewise, PROC FREQ is the best method to summarize these types of variables.
4. **Character and continuous** – These types of variables are essentially a free text response and acceptable responses include any string of characters or numbers. It is best not to summarize these types of variables since the number of distinct responses will likely be approaching the number of observations making it difficult to generate useful information.

Now that we have an idea of what kinds of data we're summarizing, a decision method can be used to ascertain the most appropriate method for summarizing each variable. Before discussing the decision method in depth, let's outline some code for evaluating all variables in a library one at a time.

The first step is to create a list of all data sets in a given library and count their number. For this we'll use the SQL procedure to query the automatic table DICTIONARY.TABLES and use SELECT INTO and COUNT() to pull the MEMNAME column into sequential macro variables and to store the number of data sets in a separate macro variable. This will enable us to reference each data set individually from within a DO loop:

```
libname sumstat 'directory/path/of/datasets';

proc sql noprint;
    select distinct memname, count(distinct memname) into :mem1 - :mem999, :memnum
    from dictionary.tables where libname='SUMSTAT';
    quit;
```

Next, we'll use a DO loop to step through the data set names stored in macro variables to get a list of variables in the current data set and count their number. Again, we'll use PROC SQL but we'll query the DICTIONARY.COLUMNS table and pull the NAME column into sequential macro variables and the number of variables into a separate macro variable:

```
%do i=1 %to &memnum;

    proc sql noprint;
           select name, count(name) into :nam1 - :nam999, :namnum
           from dictionary.columns where libname='SUMSTAT' and memname="&&mem&i";
           quit;
%end;
```

Finally, a DO loop is nested within the existing DO loop to step through the variable names stored in macro variables and evaluate each variable in the decision method:

```
%do j=1 %to &namnum;

    /* Decision Method Code Here */

%end;
```

For each variable encountered, the decision method uses the number of distinct values, a threshold value, and the variable type to assign it to one of the 4 types. The threshold value, set at 25, indicates that any variable with more the 25 distinct values should be treated as continuous and any variable with less than or equal to 25 distinct values should be treated as discrete. Here is how the 4 types of variables are assigned:

1. **Numeric and continuous** – number of distinct values > 25 and type = num
2. **Numeric and discrete** – number of distinct values <= 25 and type = num
3. **Character and discrete** – number of distinct values <= 25 and type = char
4. **Character and continuous** – number of distinct values > 25 and type = char

The first step in executing the decision method is to SELECT the number of distinct values and the type of the current variable INTO macro variables. This is done with PROC SQL querying both DICTIONARY.COLUMNS and the current data set:

```
proc sql noprint;
    select count(distinct &&nam&j) into :disval from sumstat.&&mem&i;
    select type into :namtyp from dictionary.columns where libname='SUMSTAT'
           and memname="&&mem&i" and name="&&nam&j";
    quit;
```

The next step in the decision method is to assign the category of the current variable based on the above criteria. As an example of the decision method, we'll print the category to the SAS log rather than create summary statistics. This is accomplished with IF/THEN/ELSE and PUT statements:

```
%if &disval > &threshold and &namtyp=num %then %do;
    %put &&nam&j is a continuous numeric variable;
    %end;
    %else %if &disval <= &threshold and &namtyp=num %then %do;
           %put &&nam&j is a discrete numeric variable;
           %end;
```

```
    %else %if &disval <= &threshold and &namtyp=char %then %do;
           %put &&nam&j is a discrete character variable;
           %end;
    %else %if &disval > &threshold and &namtyp=char %then %do;
           %put &&nam&j is a continuous character variable;
           %end;
```

Once the proper class is assigned to each variable, PROC FREQ and PROC MEANS or any other combination of procedures and statements can be executed accordingly. In CARe, the variable lists and summary statistics generated are made available to CARe investigators as Excel Spreadsheets and also uploaded to the CARe Portal Phenotype Trait Catalog where investigators can search variable names and labels, view summary statistics, and select variables to include in their research.

The complete decision method macro outlined above is as follows:

```
/*****************************************/
/***** Sample code for decision method *****/
/*****************************************/

%macro decision_method;

/* Set distinct value threshold for discrete/continuous decision */
%let threshold=25;

libname sumstat 'directory/path/of/data_sets';

/* Count datasets and grab memnames as macro variables */
proc sql noprint;
    select distinct memname, count(distinct memname) into :mem1 - :mem999, :memnum
    from dictionary.tables where libname='SUMSTAT';
    quit;

/* (1) Do loop for each memname */
%do i=1 %to &memnum;

/* Count variables and grab names as macro variables */
proc sql noprint;
    select name, count(name) into :nam1 - :nam999, :namnum
    from dictionary.columns where libname='SUMSTAT' and memname="&&mem&i";
    quit;

/* (2) Do loop for each name */
%do j=1 %to &namnum;

/* Decision Method */
proc sql noprint;
    select count(distinct &&nam&j) into :disval from sumstat.&&mem&i;
    select type into :namtyp from dictionary.columns where libname='SUMSTAT'
    and memname="&&mem&i" and name="&&nam&j";
    quit;

%if &disval > &threshold and &namtyp=num %then %do;
    %put &&nam&j is a continuous numeric variable;
    %end;
    %else %if &disval <= &threshold and &namtyp=num %then %do;
           %put &&nam&j is a discrete numeric variable;
           %end;
    %else %if &disval <= &threshold and &namtyp=char %then %do;
           %put &&nam&j is a discrete character variable;
           %end;
    %else %if &disval > &threshold and &namtyp=char %then %do;
           %put &&nam&j is a continuous character variable;
           %end;
```

```
%end; *(2) End do loop;

%end; *(1) End do loop;

%mend decision_method;

%decision_method;
```

AUTOMATED DATA DISTRIBUTION

Since CARe is intended to be a collaborative resource, all investigators have access to all of the data but are only able to receive data related to their manuscript proposals, which require approval by the CARe Publications Committee. Similar to the CARe summary statistics, the volume of data available is a factor to consider since writing hundreds of unique DATA steps for each data request would be an error prone, inefficient process. To facilitate automation of data distribution, a Cohort Specific Variable Request Form (CSVR) is required with each data request. The CSVR allows investigators to make precise, specific variable requests and it allows us to track which variables are requested by each investigator and for what purpose. It is comprised of 5 columns:

1. **Requested Variable** – The intent or research purpose of the variable
2. **Study** – The name of the cohort that submitted the variable
3. **Visit** – The visit when the variable was collected
4. **Memname** – The SAS data set name in which the variable resides
5. **Name** – The SAS name of the variable

Our first task in data distribution is to import and sort the CSVR by the STUDY and MEMNAME variables. For this we'll use the IMPORT and SORT procedures:

```
proc import datafile='CSVR.txt' out=csvr replace;
    run;

proc sort data=csvr;
    by study memname;
    run;
```

Now that the data set is sorted, we'll use PROC SQL, SELECT INTO, and COUNT() to store each distinct cohort name in sequential macro variables, and the total number of distinct cohorts in a separate macro variable. These will be referenced by a subsequent DO loop:

```
proc sql noprint;
    select distinct study, count(distinct study) into :stdy1 - :stdy99, :stdynum
    from csvr;
    quit;
```

From within a DO loop, we can step through each cohort stored in the macro variables and use PROC SQL to pull the MEMNAME column for the current cohort into sequential macro variables. The total number of data sets will also be pulled into a separate macro variable:

```
%do i=1 %to &stdynum;

libname distro 'directory/path/of/cohort/data_sets';

proc sql noprint;
    select distinct memname, count(distinct memname) into :mem1 - :mem999, :memnum
    from csvr;
    quit;

%end;
```

By nesting a DO loop within the existing DO loop and stepping through each data set we can again use PROC SQL to SELECT the NAME column INTO a variable list specific to the current data set. This variable list will be used to create a subset of the original data set for distribution:

```
%do j=1 %to &memnum;
```

```
proc sql noprint;
    select name into :varlist separated by ' ' from csvr where memname="&&mem&j";
    quit;

%end;
```

The final step is to subset the original data set to the specific list of variables requested by the investigator. Since that variable list is currently stored in a macro variable, this is a breeze using a DATA step and the KEEP= option:

```
libname newdata 'directory/path/for/data_distribution';

data newdata.&&mem&j;
    set distro.&&mem&j (keep=&varlist);
    run;
```

UNIQUE USER IDS

One of the mandates for distributing CARe data is that each CARe investigator is given data encoded with a random set of IDs unique to that investigator. While SAS isn't used to create the random IDs, an ID file for each investigator is imported and stored in a permanent data set so that any data sent out can quickly and easily be encoded.

This ID swap is most often performed from within the same macro used to distribute data requested in a CSVR and only requires the addition of two PROC SORT statements and a DATA step to merge the investigator's IDs with the new data set:

```
proc sort data=user_ids;
    by IdVar;
    run;

data newdata.&&mem&j;
    set distro.&&mem&j (keep=IdVar &varlist);
    run;

proc sort data=newdata.&&mem&j;
    by IdVar;
    run;

data newdata.&&mem&j (drop=IdVar);
    merge newdata.&&mem&j (in=A) user_ids;
    by IdVar;
    if A;
    run;
```

Note: The USER_IDS data set should be sorted at the beginning of the macro because it only needs to be sorted once.

Here is the complete data distribution macro as outlined above:

```
/*****************************************/
/***** Sample code for data distribution *****/
/*****************************************/

%macro distribute_data;

/* Assign library where distributed data will exist */
libname newdata 'directory/path/for/data_distribution';

/* Import and sort CSVR */
proc import datafile='CSVR.txt' out=csvr replace;
    run;

proc sort data=csvr;
    by study memname;
    run;
```

```
/* Count studies and grab them as macro variables */
proc sql noprint;
    select distinct study, count(distinct study) into :stdy1 - :stdy99, :stdynum
from csvr;
    quit;

/* (1) Do loop for each study */
%do i=1 %to &stdynum;

libname distro 'directory/path/of/cohort/data_sets';

/* Count memnames and grab them as macro variables */
proc sql noprint;
    select distinct memname, count(distinct memname) into :mem1 - :mem999, :memnum
    from csvr where study="&&stdy&i";
    quit;

/* (2) Do loop for each memname */
%do j=1 %to &memnum;

/* Select the requested variables into a macro variable */
proc sql noprint;
    select name into :varlist separated by ' ' from csvr where memname="&&mem&j";
    quit;

/* Make a subset of the requested dataset */
data newdata.&&mem&j;
    set distro.&&mem&j (keep=&varlist);
    run;

%end; *(2) End do loop;

%end; *(1) End do loop;

%mend distribute_data;

%distribute_data;
```

MANAGING FILES

CARe investigators also have access to a standardized analysis pipeline for association analysis between genotype and phenotype data. Once an investigator has created a model for a phenotypic trait using the data they requested, they can submit the modeled trait and covariates to the Broad for analysis. The preferred data format for the analysis pipeline is tab-delimited text and since each investigator submits data encoded with their unique IDs and the analysis pipeline uses another set of internal IDs, the IDs need to be swapped.

SAS is great tool for this but several obstacles such as a large number of files from each investigator, long file names, long column headers, no column headers, or nonconforming missing values are often encountered which require several unconventional techniques when importing and exporting each file. Most importantly, the exported file has to be identical to the imported file with the exception of different IDs. These techniques have been discussed at length in another SAS 2010 Global Forum paper 066-2010, *Taming the Herd: Wrangling Unruly Text Files in SAS®*.

CONCLUSION

In data-intensive projects similar in scope to CARe, leveraging nested DO loops, PROC SQL, and sequential macro variables enables a SAS programmer to efficiently and accurately complete common tasks that may otherwise become overwhelming.

REFERENCES

Musunuru K, Lettre G, Young T, Farlow DN, Pirruccello JP, Ejebe KG, Keating BJ, Yang Q, Chen MH, Lapchyk N, Crenshaw A, Ziaugra L, Rachupka A, Benjamin EJ, Cupples LA, Fornage M, Fox ER, Heckbert SR, Hirschhorn JN,

Newton-Cheh CH, Nizzari MM, Paltoo DN, Papanicolaou GJ, Patel SR, Psaty BM, Rader DJ, Redline S, Rich SS, Rotter JI, Taylor HA Jr, Tracy RP, Vasan RS, Wilson JG, Kathiresan S, Fabsitz RR, Boerwinkle E, Gabriel SB. Candidiate Gene Association Resource (CARe): design, methods, and proof of concept. Circ Cardiovasc Genet, in the press.

Taylor Young. "066-2010: Taming the Herd: Wrangling Unruly Text Files in SAS®" The proceedings of the the SAS Global Forum 2010.

RECOMMENDED READING

SAS Institute Inc. 2010. **SAS OnlineDoc® 9.2.** Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Taylor Young
The Broad Institute
7 Cambridge Center
Cambridge, MA 02142
tyoung@broadinstitute.org
www.broadinstitute.org