PharmaSUG 2023 - Paper QT-239

Taming the Large SAS® Dataset by Splitting it Up

David Franklin, TheProgrammersCabin.com

ABSTRACT

Large SAS® datasets tie up computer resources and are often not useful to have as stand-alone entities. You may also want to split a dataset set up into pieces either for the practicalities of sending a dataset in small chucks to a single recipient, or send subsets of that data to different recipients.

This paper will show the development of two macros that are useful in splitting the large dataset, the first being specifying the number of observations that should go into an output dataset, and the second being split by values.

INTRODUCTION

SAS datasets can be very large and cumbersome for computer resources. Often it will help to split that dataset up into two or more pieces before processing.

Two macros in this paper are presented that can do this task, the first by specifying the number of observations that can go into the output dataset, and the second by specifying a variable that can be used to split the datasets by.

SPLITING A DATASET BY SPECIFYING THE NUMBER OF OBSERVATIONS

Specifying the maximum number of observations that an output dataset can have is one of the ways a dataset can be split. This is useful if you want the datasets to have the same number of observations in each dataset, except for the last dataset which could have less than the number specified. The dataset name of the outgoing dataset will the same name as the original dataset but with a number character followed by a number.

To do this type of split we have to know three things – the dataset name (macro variable &dataset) and number of output datasets, and the maximum number of observations that each output data dataset should have (macro variable &numindset).

The dataset name where the data is coming from, and the maximum number of observations that each output data dataset should have are both user definable, but calculating the number output datasets needed is something that the program needs to know, as shown in the following code:

```
data _null_;
   if 0 then set <any_dataset_name> nobs=nobs;
   call symput('numoutdset',strip(put(ceil(nobs/&numindset),8.)));
   stop;
run;
```

It the code above, the number of observations is read off the first pass-though of the data and used to create a macro variable called NUMOUTDSET is created which counts the number of datasets that are going to be created.

To get the observations for each dataset, the FIRSTOBS and OBS SET options are deployed, as shown below:

```
data &dataset._&i;
   set &dataset (firstobs=%eval((&numindset.*(&i-1))+1)
                 obs=%eval(&numindset.*&i));
run;
```

In this code the first observation in the outgoing dataset is calculated as well as the last observation, so in effect a slice of the original dataset is going into the outgoing dataset.
Bringing this altogether, the macro that would do this type of dataset split is given below:

```
%macro breakDataset(dataset=_last_   /*Input dataset*/
                   ,numindset=100   /*Maximum num of obs output datasets*/
                   );

   *Count the number of records and calculate the number of datasets needed;
   data _null_;
      if 0 then set &dataset nobs=nobs;
      call symput('nobs',strip(put(nobs,8.)));
      call symput('numoutdset',strip(put(ceil(nobs/&numindset),8.)));
      stop;
   run;

   *Split dataset -- output dataset name is OriginalDatasetName_number;
   %do i=1 %to &numoutdset;
      data &dataset._&i;
         set &dataset (firstobs=%eval((&numindset.*(&i-1))+1)
                       obs=%eval(&numindset.*&i));
      run;
   %end;

%mend breakDataset;
```

The following example creates a dataset X with 2500 records, and when the macro is run three datasets are created (X_1, X_2 and X_3) with 1000 observations for the first two and 500 observations in the last dataset.

```
data x;
   do k=1 to 2500;
      output;
   end;
run;
%breakDataset(dataset=x
             ,numindset=1000
             );
run;
```

SPLITTING DATASETS BY A VARIABLE

Now lets look at the case where we have a SAS dataset that we split by a variable. For this purpose the following data will be used:

```
data _AllPets;
   length pet $3 name $9;
   infile cards;
   input pet $  name $ @@;
cards;
Cat Pacha Cat Muschi Dog Mia Dog Dixie Dog Princess Cat Archie Dog Bella
Cat Princess Cat Boots Dog Gus Cat Milly Dog Bailey Cat Max Dog Roxie
Dog Cody Dog Boomer Cat Sooty Cat Shadow Dog Tucker Cat Faraday
;
run;
```

In this example we have a dataset with a name (variable NAME) and whether they are a Cat or a Dog -- what we want to do is take this dataset and create two datasets, one called CAT and the other DOG. It is easy to see from the data that this could be done in one datastep as shown below:

```
data cat dog;
   set _AllPets;
   if pet='Cat' then output cat;
   else if pet='Dog' then output dog;
run;
```

But what if this was a very large dataset containing a hundred PET values with a thousand NAME values -- not so easy. So lets look at something a little more general. Lets first create a macro variable with the distinct PET values:

```
proc sql noprint;
   select distinct pet
      into :petval separated by " "
      from _AllPets;
   quit;
run;
```

Running this code creates a macro variable with the distinct PET values, separated by a space. Now we are going to create a macro that will use each value of PET in this macro variable PETVAL and create a dataset with the same name:

```
%macro breakup;

   *Initialize macro counter;
   %let i=1;

   *A DO loop to actually do the breakup -- read each value
    of PET contained in the macro variable PETVAL until no more;
   %do %while(%scan(&petval,&i) ne );

      *For each value of PET create a dataset with that VALUE,
       dropping PET from the outgoing dataset;
      data %scan(&petval,&i);
         set _AllPets;
         where pet="%scan(&petval,&i)";
         drop pet;
      run;

      %let i=%eval(&i+1);

   %end;
%mend breakup;
```

Running this macro will read each value from the macro variable PETVAL, create a dataset with that value, one called *Cat* and the second called *Dog*, and use a WHERE statement to do the selection of the records from _ALLPETS. Using a PRINT procedure call on each the two datasets created, the following data is present:

```
List of Cat Names

 Obs    name

   1     Pacha
   2     Muschi
   3     Archie
   4     Princess
   5     Boots
   6     Milly
   7     Max
   8     Sooty
   9     Shadow
  10     Faraday


List of Dog Names

  Obs    name

   1     Mia
   2     Dixie
   3     Princess
   4     Bella
   5     Gus
   6     Bailey
   7     Roxie
   8     Cody
   9     Boomer
  10     Tucker
```

Now this would be okay and it certainly would work if edited every time, but there are few more enhancements that will help make this a more general macro, which is what we want, namely:

- Dataset name to be a macro variable
- The variable to split by be a macro variable
- Handle the cases where a split variable may have a space as part of a valid value.

The first two requirements are easily handled, but the third is a little tricky. If we were to change the INTO line to read

```
into :petval separated by "~"
```

we could select the values by scanning the line for “~” instead of spaces.

Putting all this together, we get a more robust macro of

```
%macro breakup(dsn=         /*Dataset name*/
              ,dsv=         /*Variable in dataset to use split by value*/
              ,vrs=%str( )  /*Values separated in list by
                             char - default space*/
              );

   *Get a unique list of values;
   proc sql noprint;
      select distinct &dsv
         into :petval separated by "&vrs"
         from &dsn;
      quit;
   run;

   *Initialize macro counter;
   %let i=1;

   *A DO loop to actually do the breakup -- read each value
    inside in the macro variable PETVAL until no more;
   %do %while(%scan(&petval,&i,&vrs) ne );

      data %scan(&petval,&i,"&vrs");
         set &dsn;
         where &dsv="%scan(&petval,&i,"&vrs")";
         drop &dsv;
      run;

      *Increment the DO LOOP counter;
      %let i=%eval(&i+1);

   %end;

%mend breakup;
```

Now taking this macro and the original _ALLPETS data, the call to split up the dataset by PET would be called using:

```
%breakup(dsn=_ALLPETS,dsv=PET);
```

CONCLUSION

This paper has taken us first though creating a macro that will split a large dataset into smaller pieces by specifying the maximum number of observations in the outgoing datasets, and then went on a journey or creating a general macro that would take any dataset and split it by a specified variable value.

CONTACT INFORMATION

Name: David Franklin
Enterprise: TheProgrammersCabin.com
E-mail: dfranklinnh@gmail.com