So Much Data, So Little Time: Splitting Datasets For More Efficient Run Times and Meeting FDA Submission Guidelines

Anthony Harris, PPD, Wilmington, NC
Robby Diseker, PPD, Wilmington, NC

ABSTRACT

When clinical trials increase in duration, enrollment population, or both, the amount of data collected grows proportionally. The result is larger datasets that must be split for regulatory submissions to meet the current FDA requirement of being less than 100 megabytes in size. This paper describes a macro that splits datasets while keeping patient data together. It can reduce any sized dataset to fit the regulatory requirement. Since these larger datasets also require longer run times in producing analysis datasets, it will be shown how using the macro in programming of analysis datasets can reduce run time. A comparison of run times on large datasets with and without this approach is included.

INTRODUCTION

With lengthy clinical trials often collecting data for thousands of patients, datasets frequently exceed the current FDA requirement of 100 megabytes (MB) in size. This is not only a problem at submission time, but can also slow down run times, especially on resource-intensive procedures like PROC SORT. While it is often necessary to produce a final dataset sorted in a particular order, other variable sort orders are often required at intermediate points in the program. Because PROC SORT is a heavy resource user, both in terms of space and processing time, as datasets increase, the processing time increases not linearly, but geometrically. This can severely limit how fast a program runs and, in a deadline-driven business like the pharmaceutical industry, put projects at risk of meeting deadlines. By splitting datasets into smaller sets, the split_data macro can help reduce run-times by 60% or more. The purpose of this paper is to describe the functionality of a macro designed to split a large dataset into multiple datasets, all of which meet FDA required size restraints, as well as give some examples of how this macro can be implemented to achieve improved program performance.

DESCRIPTION OF HOW %SPLIT_DATA WORKS

MACRO PARAMETERS

INDATA – the input dataset that will be split.
INLIB – libname for location of input dataset
OUTLIB – libname for location to write the output dataset(s)
PT – ID variable for dataset, in most clinical trials, this will be a patient number. The split_data macro will keep all data for the specified variable in the same dataset. Note that for character variables, the macro assumes the data is already sorted by this variable.
ROOT – the root name of the created datasets, the new names will be the specified root concatenated with 01-N (the total number of datasets required for each of them to meet 100MB size limit).
PCTSIZE – The percentage of the input dataset size for which each new dataset will be created. If left blank, %split_data will calculate the largest percentage necessary to create the least amount of datasets possible to meet FDA submission requirements.

The first step of %split_data determines whether the patient identifier variable (PT parameter) is a numeric or character variable. This determination will provide for which of two ways the macro will split the data while keeping all of each patient’s data in the same dataset. The next step of the macro determines the number of records in the dataset as well as the cumulative percentage and count of records for each PT variable. Once these values are determined, the macro uses the parameter PCTSIZE, if given, to determine the number of datasets to be created. For example, if you want 20 datasets to be created, set PCTSIZE to 5. A general computation to compute the PCTSIZE parameter is 100 divided by the number of datasets desired. If it is not given, %split_data uses a computation from a macro called %dssize(1) to determine the total size of the INDATA dataset and calculates the number of 100MB or smaller datasets that should be created. At this point, the macro will act according to the type of variable that PT is assigned.
If the PT parameter is a numeric variable, %split_data uses actual patient numbers based on cumulative percentages to keep all observations for each patient together.

If the PT parameter is a character variable, %split_data will use the cumulative frequency to keep all observations for each patient together. The macro assumes the dataset is already sorted by the character variable to avoid unnecessary sorts of the input dataset.

**USING %SPLIT_DATA TO ACHIEVE BETTER RUN-TIMES**

**TESTING OF MACRO**

Extensive testing was done on %split_data to try to determine how much time can be saved by using this macro. All of the testing was done in an open VMS environment, so there may be some discrepancies in the findings due to queue size, server space, etc. For simplicity in testing, a single PROC SORT was performed on laboratory datasets with 159 variables of multiple sizes: 100, 200, 300, 500, and 3209 MB. We performed the test using one, three, and six BY variables to see how results may vary by the complexity of the sort. The control test was the time in seconds that it took to do a single proc sort on the largest version of the dataset. The comparison test included the time to split the dataset using %split_data, perform the same proc sort on each of the smaller, created datasets, and append them back into one larger dataset. The coded for testing %split_data are below and the comparison results can be found in Table 1.

```latex
%let start = %sysfunc(datetime()); **Set Start Time for Run Time Calc at end; %split_data(indata=&ds., inlib=work, outlib=work, pt=pt, root=newset);

option mprint spool;
%macro sorter(dataset=);
proc sort data=work.&dataset out=&dataset; by &SORTBY.; run;
proc append base = NewOUT data = &dataset.; run;
%mend sorter;
proc contents data = work._all_ out = allds noprint ; run ;
data allds (keep = memname) ;
set allds ;
by memname ;
if first.memname and upcase(substr(memname,1,6)) = 'NEWSET'; run ;
filename datasets "allds.txt" new ;
data _null_ ;
file datasets print new notitle ;
set allds ;
output = '%sorter(dataset=' || compress(memname) || ")" ;
put output ; run ;
%include "allds.txt" ;
x "delete/noconfirm allds.txt;"*

******************************************************************************;
* output to log the run time *;
******************************************************************************;
%let seconds = %sysfunc(round(%sysfunc(sum(%sysfunc(datetime())), -&start)), .01));
%let minutes = %sysfunc(round(%sysevalf(&seconds/60), .01));
%put Note (RunTime): Real Run Time: &seconds seconds -OR- &minutes minutes;

```
TABLE 1. RUN-TIME EXAMPLES

<table>
<thead>
<tr>
<th>Size (MB)</th>
<th>1 Variable Sort</th>
<th>3 Variable Sort</th>
<th>6 Variable Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sorting Whole Dataset</td>
<td>Split, Sort and Append Subsets</td>
<td>Time Saved</td>
</tr>
<tr>
<td>100</td>
<td>10.4</td>
<td>14.0</td>
<td>-3.6</td>
</tr>
<tr>
<td>200</td>
<td>8.62</td>
<td>15.94</td>
<td>-7.3</td>
</tr>
<tr>
<td>300</td>
<td>27.2</td>
<td>24.6</td>
<td>2.6</td>
</tr>
<tr>
<td>500</td>
<td>169.3</td>
<td>82.1</td>
<td>87.2</td>
</tr>
<tr>
<td>3209</td>
<td>924.0</td>
<td>387.9</td>
<td>536.1</td>
</tr>
</tbody>
</table>

Time is in seconds

For the 100mb dataset, 10% was used as the pctsize. For all others, 100mb subsets were used.

FINDINGS FROM TESTING

Initial testing on small datasets proved that %split_data is only advantageous to run-times on large datasets. Splitting a dataset smaller than 300 MB and performing a proc sort on the smaller datasets actually took longer than just sorting the original dataset. As larger datasets were used, it was found that run times grew exponentially when sorting the entire dataset by 1 variable, and %split_data saved a higher percentage of time as the dataset grew. By adding variables to the sort, and thus creating a more realistic sort for laboratory type data, it was found that %split_data continued to save valuable run-time. The largest advantage was found on the biggest dataset which saved between 50% and 60% of runtime on all three sort types. While the seconds saved on a single proc sort may seem insignificant, these percentages begin to show just how much time could be saved in a real world environment throughout an entire analysis database program.

LOG REVIEW

Upon completion, %split_data writes some helpful hints and information regarding the outcome of the macro run to your log file. When using the macro to generate 100MB sized files automatically, the first note to look for is the size of your dataset; both kilobytes(KB) and megabytes(MB) from %dssize.

(SPLIT_DATA) kb = 3209501
(SPLIT_DATA) mb = 3134.3

If your total dataset size is less than 100 MB, %split_data will let you know this and recommend supplying a value to the PCTSIZE parameter if you want to split your dataset.

(SPLIT_DATA) Note: No Data Split Necessary, labdata = 94 <= 100 MB Size Limit
(SPLIT_DATA) Note: To override, use PCTSIZE parameter

The next important note to look for is the following,

(SPLIT_DATA) NOTE: PT is Character type. Macro assumes data is sorted by PT.

If this note appears in the log, the PT parameter is of character type, %split_data assumes that the dataset is already sorted by PT, and the cumulative frequency should be used to keep all observations for each PT together. If this note is not in the log, PT is numeric, and %split_data used the actual patient numbers based on cumulative percentage to keep all of a PT’s observations together.

Once %split_data has created new datasets, the following note can be found in the log to let you know how many datasets you are now working with. A global macro variable, MAXDEX2 is created that can be used in further programming.
The last note to review in your log allows for %split_data to use the law of averages to assume all datasets created will be smaller than 100MB. Throughout testing, we found that occasionally %split_data will create a dataset slightly larger than 100MB prior to compression due to a slight variance between calculated size and actual dataset size. For example, when testing on a 3209mb dataset, the %dssize macro calculated 3134mb which is 97.6% of the actual size. To accommodate this, %split_data uses 96MB instead of 100MB as a target size. This does not, however, completely remedy the situation, so the following note will appear in the log.

(SPLIT_DATA) Note: If Some Datasets Are Still > 100MB, Reduce PCTSIZE By 1 Decimal From 3.1

CONCLUSION

With FDA requirements and ever tightening timelines, splitting datasets is becoming an essential tool in the pharmaceutical research industry. Using %split_data is an effective way to output datasets that not only meet federal requirements, but also give the programmer a much more manageable dataset for analysis database building as well as TLF programming.

REFERENCES


CONTACT INFORMATION

Anthony Harris
PPD, Inc.
929 North Front Street
Wilmington, NC 28401
Work Phone: 910-558-8648
Fax: 919-654-0430
Email: Anthony.Harris@wilm.ppdi.com

An electronic version of this program can be obtained via e-mail from the author. SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. Other brand and product names are trademarks of their respective companies.

SOURCE CODE

**********************************************************************
* PROGRAM NAME:    split_data.sas
* SAS VERSION:   8.2
* PURPOSE:         Macro to split a dataset based by
*                  patient # and a given
*                  percent size (decimals OK) of original data;
*                  
* USAGE NOTES:     Generally used to split data for FDA;
*                  submissions. Will calculate size of input;
*                  dataset and create 100 MB sized datasets;
*                  To Determine size of dataset manually:
*                  Percent= (1/(Filesize in MB/100))*100;
*                  or Percent = 10000/Filesize in MB ;
*                  Recommended to use at least 1 decimal if < 10;
%macro split_data (
  indata  = labdata,   /**Designate input data (indata)*/
  inlib   = work,      /*input library*/
  outlib  = work,      /*output library*/
  pt      = pt,        /*designate Patient ID (pt);*/
  root    = new,       /*root name of new dataset, i.e., new will
  make new01-newN*/
  pctsize =            /*Default = blank. If blank, will calculate
  size of dataset and
  create 100 MB sized datasets. Designate percent size (pctsize) of indata to
  make each new dataset;*/
);  
  ***check type of var;
  %local VarType;
  %let VarType = ; %**Resets value to mis-s-ing if used before;
  proc contents data = &inlib..&indata out=contents (keep = name
  type) noprint;
  run;
  data _null_;  
  set contents;
  if upcase(name) = "%upcase(&PT)" then do;
    call symput('VarType',
               compress(put(type,8.)));**(1=Numeric, 2 = character);
    end;
  run;
  **clean up work area;
  proc datasets library = work;
  delete contents;
  quit;
  ***Get Frequency of Patient ID;
  proc freq data = &inlib..&indata noprint;
  tables &pt/list missing out=count;
  run;
  ***Count number of obs in the dataset and put into numobs macro
  variable;
  data _null_;  
  if 0 then set &inlib..&indata. nobs=nobs;
  call symsput('numobs',trim(left(put(nobs,8.))));
  stop;
  run;
  %put (Split_data) numobs = &numobs;
***Assign Deciles to the row counts;***

%if %length(&pctsize) = 0 %then %do;
   /*
   Code from Xingshu Zhu and Shuping Zhang.
   A New Method to Estimate the Size of a SAS Data Set.
   SUGI 27, February 16, 2006 to calculate size of dataset;
   */
   proc printto print="dssize.txt" new;
   proc contents data=&inlib..&indata;
   proc printto;
   run;

   data _dssize_(keep=pagesize totpages);
      length string $200;
      infile 'dssize.txt' length=lg;
      retain pagesize totpages;
      input @1 string $varying. lg;
      if index(upcase(string),"DATA SET PAGE SIZE") then
         pagesize=input(compress(scan(string,2,".")),8.);
      if index(upcase(string),"NUMBER OF DATA SET PAGE") then
         totpages=input(compress(scan(string,2,".")),8.);
      if index(upcase(string),"OBS IN FIRST DATA PAGE") then output;
   run;

   %local kb mb;
   data _null_;
      set _dssize_;
      size_bt=256+pagesize*totpages;
      kb=ceil(size_bt/1024);
      mb=round(size_bt/1048576,0.1);
      call symput("kb", trim(left(put(kb,8.))));
      call symput("mb", trim(left(put(mb,8.))));
   run;
   %put (Split_data) kb = &kb.;
   %put (Split_data) mb = &mb.;

   proc datasets nolist;
      delete _dssize_; quit;
   %let mb1 = %scan(&mb.,1,'.');
   %let mb2 = %scan(&mb.,2,'.'),
   %let top = 100; *Max size of datasets;
   %if (&mb1. < &top.) or ( &mb1. = &top. and &mb2. <= 0) %then %do;
      %put (SPLIT_DATA) Note: No Data Split Necessary, &indata=
         &MB <= 100 MB Size Limit;
      %goto exit; %***data with 100mb limit;
   %end;
   %else %put (SPLIT_DATA) Note: &indata size = &MB MB > than 100
      MB. Therefore, Macro Will Split Data;
   %local rc;
   filename TARGET 'dssize.txt';
   %let rc=%sysfunc(fdelete(TARGET));

   data count;
      set count end=eof;
      retain cumpct cumfreq 0 ;
      cumpct = cumpct +percent;
      cumfreq=cumfreq+count;
      if cumfreq = &numobs. then cumpct = 99.999;
   run;
```plaintext
tempvar = (96/\&mb.) *100; **used 96 vs. 100 to ensure datasets <100 
   mb = \&mb.; 
   if tempvar < 1 then dex = 100; 
   else dex = int(cumpct/tempvar); 
   if eof then do; 
      call symput('PCTSIZE', trim(left(put(tempvar,best.)))); 
   end; 
run;

%end;
%else %do;
***Base number of datasets off of given percent size;
data count;
   set count;
   retain cumpct cumfreq 0 ; 
   cumpct = cumpct +percent; 
   cumfreq=cumfreq+count; 
   if cumfreq = \&numobs then cumpct = 99.999; 
   dex = int(cumpct /\&pctsize); 
run;
%end;

proc sort data = count;
   by dex cumfreq;
run;
data count;
   set count;
   by dex cumfreq;
   if first.dex and dex >0 then output;
run;

proc print data = count;
   title 'Splits based on CumPct';
run;
%if \&vartype = 1 %then %do;
***USE values of macro var PT (numeric) to split datasets;
*** put PT values from Count dataset (from Frequency) into macro variables;
data _null_; 
   set count end=eof;
   call symput( 'DIV'||trim(left(put(dex,8.))), put(&pt,8.));
   if eof then do;
      call symput( 'maxdex', put(dex,8.));
   end;
run;

***Output Datasets based on the Numeric Values of Patients;
%do i = 1 %to \&maxdex;
   %if %length(&i) < 2 %then %let ix = 0&i;
   %else %let ix = &i;
   %if &i = 1 %then %do;   %***FIRST SET OF OBS; 
      %put this is obs***&i***;
      proc sql;
         create table \outlib..&root.\ix as
            select a.*
               from \inlib..\indata as a 
               where &pt <= %superq(div&i.);
      run;
   %end;
   %else %do;  %***2nd to Next to last sets;
      proc sql;
         create table \outlib..&root.\ix as
```
select a.*  
from &inlib..&indata as a  
where &minobs < &pt <= %superq(div&i.);  
run;  
%end;  
%let minobs = %superq(div&i.);  
%end;  
***End of 1 to maxdex;  

%%%The last set of obs is new output;  
%global maxdex2;  
%let maxdex2 = %eval(&maxdex+1);  
%put (Split_data) Total # datasets created: parameter MAXDEX2 =  
&maxdex2;  
%if %length(&maxdex2) < 2 %then %let ix = 0&maxdex2;  
%else %let ix = &maxdex2;  
proc sql;  
create table &outlib..&root.&ix as  
select a.*  
from &inlib..&indata as a  
where &pt > &minobs;  
quit;  
%end;  
%%%End of splitting based in numeric PT;  

%else %if &vartype = 2 %then %do;  
%%USE values of CUMFREQ to split datasets since PT is character;  
%% put observation number values from Count dataset (from  
Frequency) into macro variables;  
%put (SPLIT_DATA) NOTE: %upcase(&PT) is Character type. Macro  
assumes data is sorted by %upcase(&PT);  
data _null_;  
set count end=eof;  
call symput( 'DIV'||trim(left(put(dex,8.))),  
trim(left(put(cumfreq,8.))));  
if eof then do;  
call symput( 'maxdex', trim(left(put(dex,8.))));  
end;  
run;  
%let minobs = 0;  
***options nomprint nosymbolgen;  
***Output Datasets based on the CumFreq to keep Patients together;  

%do i = 1 %to &maxdex;  
%if %length(&i) < 2 %then %let ix = 0&i;  
%else %let ix = &i;  
%if &i = 1 %then %do;  
%%FIRST SET OF OBS;  
%put this is obs***&i***;  
data &outlib..&root.&ix;  
set &inlib..&indata (obs = %superq(div&i.));  
run;  
%end;  
%else %do;  
%%2nd to Next to last sets;  
data &outlib..&root.&ix;  
set &inlib..&indata (firstobs = &minobs obs =  
%superq(div&i.));  
run;  
%end;  
%let minobs = %eval(%superq(div&i.)+1);  
%end;  
%%%The last set of obs is now output;  
%global maxdex2;  
%let maxdex2 = %eval(&maxdex+1);  
%if %length(&maxdex2) < 2 %then %let ix = 0&maxdex2;
%else %let ix = &maxdex2;

data &outlib..&root.&ix;
set &inlib..&indata (firstobs = &minobs);
run;

%put (Split_data) Total # datasets created: (parameter MAXDEX2) = &maxdex2;

%end; /**end of Do loop;

**Log Note when dataset created;
%if %length(pctsize)^=0 %then %do;
%put************************************************************************
***************************;
%put (Split_Data) Note: There were &maxdex2 files created named
&root.01 - &root.&maxdex2. ;
%put (Split_Data) Note: If Some Datasets Are Still > 100MB,
Reduce PCTSIZE By 1 Decimal From &PCTSIZE *;
%put************************************************************************
***************************;
%end;

**exit: /**Exit here for <100MB datasets */
**clean up work area;
proc datasets library = work nolist;
delete count;
quit;

%mend split_data;