

Creating Variable Numbers Of Output Files Based On Dataset Values

Rob Boostrom, Crate & Barrel, Northbrook, IL

ABSTRACT

This paper will describe a technique, using macro variables, for dividing a large set of data into a variable number of smaller files and saving each file with a name, and in a directory, based on values within the data. In other words, if there is a data set that can have sales from 25 offices on one run and 782 offices on the next, the program uses macro variables and do loops to create the number of output records equivalent to the number of unique offices that are within the source data set. Tools utilized include CALL SYMPUT, macros with do loops and SYSTASK.

INTRODUCTION

Consider a scenario where, organization wide, 200,000 customers need to be contacted by an unspecified number of your 500 regional offices. Each regional office wants a file that they can easily open in a common spreadsheet program and easily identify just the folks that they need to contact – not the folks that need to be contacted by the other offices.

Though the specific scenario above may not be your own, it is quite possible for you to have a similar situation – in which you need to create a large number of output files for a large number of end users. One way to do it is to write SAS code that includes a data step naming each output file and defining how records would get split. Another way, however, is to use macro variables, macros and other techniques to dynamically handle every output file that you wish to create.

The program, in its entirety, is as follows:

```
/******
```

```
Goal: Generate output files based on a list of undetermined variable values from a source file.
```

```
*****/
```

```
***** Assign the library of the file that needs to be split;
```

```
libname source      "C:\Program Files\SAS Institute\SAS\V8\core\sashelp\";
%let splitfile=shoes;
%let splitvar=subsidiary;
```

```
***** Create a sorted output that is a list to represent all of the splitvar values;
```

```
proc freq data = source.&splitfile (keep=&splitvar) noprint;
    table &splitvar / out = splitvar_list;
run;
```

```
***** Get the total number of distinct splitvar values;
```

```
data splitvar_list;
    set splitvar_list end=final;
    splitvar_num = _n_;
    if final then call symput('lastvar',_n_);
run;
```

```
***** Sort source file to merge against new splitvar list file;
```

```
proc sort data = source.&splitfile out = all_recs;
    by &splitvar;
run;
```

```
***** Run macro loop once for each splitvar, creating the ouptut
    directory and output file;
```

```

%macro split;
%do i=1 %to &lastvar %by 1;

data splitvar&i (drop=count percent splitvar_num);
    merge
    all_recs (in=a)
    splitvar_list (in=b)
    ;
    by &splitvar;
    if &i = splitvar_num then do;
        call symput('newname',compress(&splitvar,'/- &%().'));
        output;
    end;
    if a;
run;

systask command "md C:\sas_paper_out\&newname";

proc export data=splitvar&i
    outfile="C:\sas_paper_out\&newname\&newname..csv"
    replace;
run;

%end;
%mend;
%split;

```

This demonstration code above is designed to run on any Windows machine with version 8 on it. (It uses the SHOES data set from SAS HELP.) You will notice, in reviewing this code, that the key to this is loading and utilizing macro variables. By dynamically setting values that are then used to extract data and name output files, there is no need to actually know how many different output files there will be. The number of distinct values in a single variable drives everything.

GETTING THE NUMBER OF DISTINCT VAUES

The first step is to get a file that contains all distinct values in the order that SAS will sort them. In my example, I use a proc freq against the file that I want to split. It's a quick way to get SAS to provide all distinct values in a sorted list. The output of the proc freq is "splitvar_list". The next step is to get the count of distinct values. I do that by using a call symput to create a variable called "lastvar". The example is below:

```

data splitvar_list;
    set splitvar_list end=final;
    splitvar_num = _n_;
    if final then call symput('lastvar',_n_);
run;

```

This data step does two things: It creates the variable splitvar_num, which will be used later to select which value is used to create output files in the split macro, and it creates "lastvar" so that the split macro will know how many iterations are required. Once this is done, I sort the file that I will be splitting to prepare it for the split macro.

THE SPLIT MACRO

The split macro is the meat of the whole program. Since you've already figured out how many iterations are required, this macro runs once for each unique value and creates an output directory and file for each run.

The macro starts off as follows:

```

%macro split;
%do i=1 %to &lastvar %by 1;

```

Note that this employs the "lastvar" variable created in the previously mentioned data step. After this, the split macro has three components in this program: The data step that does the split, the SYSTASK to create the output directory, and the export to create the output.

THE SPLIT DATA STEP

This data step is a merge that merges only one record from the split variable list, dependent on the number of the iteration, with the full list. Along the way, the one value of the split variable is used to create a macro variable by again employing a call symput command:

```
data splitvar&i (drop=count percent splitvar_num);
  merge
    all_recs (in=a)
    splitvar_list (in=b)
  ;
  by &splitvar;
  if &i = splitvar_num then do;
    call symput('newname',compress(&splitvar,'/- &%().'));
    output;
  end;
  if a;
run;
```

I put a compress in here because I have had situations where the split variable contained slashes or hyphens (like "WASHINGTON DC/BALTIMORE") and dropping them out has become a "better safe than sorry" strategy.

SYSTASK CREATION OF OUTPUT DIRECTORIES

Because a different output file is required for each distinct value, it may also be helpful for each output file to be in its own directory. The advantage to this is even greater when there are multiple files that will be split out. If each regional office will get several reports specific to them, this method allows you to ensure that there is a directory whenever there will be a file.

The following line of code handles this work for you:

```
sytask command "md C:\sas_paper_out\&newname";
```

There are a couple of things to note about this. First, there are probably some fans of "x command" who wonder why I'm using SYSTASK for this. I found that SYSTASK will close out more regularly, whereas x command will often get hung up on a C: prompt window. Second, I'd like to note that if the directory already exists, the SYSTASK will just open and close, moving on to the next task. If there are already files in the directory, the SYSTASK won't delete them and recreate the directory. In other words, it's safe to use repeatedly.

OUTPUT EXPORT

All of the steps in this program might be changed for specific environments, but this particular portion of the program is probably the most likely one for a programmer to want to modify dependent on the end users of the output. For my demonstration, I have chosen to put the files in .csv format because it's an export option that is part of the SAS Base system and because it's easily handled by the more popular spreadsheet programs:

```
proc export data=splitvar&i
  outfile="C:\sas_paper_out\&newname\&newname..csv"
  replace;
run;
```

This export uses the directory freshly created by the SYSTASK command and uses the same name for the actual output file. Keep in mind that another option would be to hard code something before or after the "newname" variable so that if many split reports are run users can distinguish between the different output files.

CONCLUSION

My coworkers and I have utilized this code in one form or another within several projects. It can be a simpler way of handling files that need to be divided more ways than you would want to try and hard code.

ACKNOWLEDGMENTS

I'd like to thank past coworker David Hussong and present coworker Ryan Snedden, as their projects helped to drive the data manipulation questions that I have tried to answer.

CONTACT INFORMATION

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

Rob Boostrom
Crate & Barrel
1250 Techny Road
Northbrook, IL 60062
Work Phone: 847-239-6521
Fax: 847-562-9626
Email: rboostrom@crateandbarrel.com

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.