PharmaSUG 2018 - Paper BB-15

Get Your Sort On: Macro Driven Method for Automated Unique Sorting

Kevin Miller, Gilead Sciences

ABSTRACT

With an ever growing desire to automate processes for increasingly more tasks, pressure falls to programmers to make their programming more adaptable and efficient. One such task is uniquely sorting data; this paper will present a method, a SAS macro, to perform this task using minimal oversight.

INTRODUCTION

It's late Friday afternoon - a stained, empty mug sits on your desk. Most of your coworkers have already started their weekend. Your data vendor just sent over the latest version of lab data for your study. You process the data set, send it to production and get ready to head out the door. Just when you're ready for the weekend to call; your phone rings. But it's not Saturday on the phone, it's your data manager. They are wondering why this transfer has fewer records than the last lab data transfer, and by the way, "can you tell me which observations are missing from the latest transfer?" You begin to panic that you will be trapped in the office forever, but then it hits you. You remember your programming team developed a tool to get this job done easier than it is to remember the PROC SORT syntax. So go ahead and let your brain turn on the proverbial Out of Office. Because it's time to Get Your Sort On.

In addition to the above circumstance, there are many situations where determining what makes your data sort unique is necessary. Whether you're merging, comparing, or transposing; the need to know the variables in your data that make its rows unique exists. This paper will provide insight into a macro method to allow SAS to do the heavy lifting for you. You will be exposed to the iterative %DO loop and some basic SAS Macro functions.

BREAKING DOWN THE METHOD

GET CONTENTS OF YOUR DATA

In order to determine which variables make a data set unique, the tool will need to first find out information about the input data set. First, it will retrieve some metadata about the input data set, grabbing all of the variable names and a total count of variables.

We can easily grab all of this info using a quick SAS SQL procedure statement:

```
PROC SQL noprint;
  select strip(put(COUNT(*),best.)),name
     into :varcnt,:varnam separated by ' '
     from dictionary.columns
      where upcase(libname)=upcase(your libname) AND
      upcase(memname)=upcase(your data set);
 quit;
```

The table dictionary.columns is an automatically created data set that stores metadata information about available data sets. The tool will access this table to grab the info to create the two macro variables, "varcnt" and "varnam" that are used in subsequent steps of the process.

YOUR TIME TO LOOP

The tool needs to iteratively loop through the input data set at least once, but up to as many times as it takes to identify a sort that doesn't create duplicates. We know our first loop count is one and our last possible loop would be the total count of all variables, or the macro variable varcnt from our above SQL statement. Using this number, the tool will add variables one by one to our sort by statement using this code:

```
%do _dup_ = 1 %to &varcnt;
  data _null_;
   call symput('keys', strip("&keys")||" "||scan("&varnam",&_dup_));
 run;
```

This will scan through the macro variable containing all of our variable names "varnam", adding an additional variable to the new macro variable, "keys", each loop.

SORT IS NEXT

Next is the most important step, actually determining the by variables that will make the input unique. The PROC SORT procedure offers many options and settings to generate exactly the sort the user desires, below are the highlights of the two that are utilized, nodupkey and dupout:

```
proc sort data = your data set out = outsort nodupkey dupout = dups;
 by &keys;
run;
```

The option NODUPKEY tells the sort to delete any observations that are duplicate based on the by values. Option DUPOUT creates a data set with the observations deleted using the NODUPKEY option. We can count the number of observations in the data set specified after the DUPOUT function; in this case the data set named "dups" to know if the data set was sorted without any duplicate values. If we have observations in the data set then we will continue looping, adding an additional variable to our SORT statement. If our DUPOUT data set contains no observations, we can end the loop and look at our results. In the rare case that we loop through all of our variables and still have observations in our DUPOUT data set then the tool will provide an error and alert the user.

ON TO THE RESULTS

Once the sort has found a set of by values that created a DUPOUT data set with 0 observations, or we have added all variables in our input data set to the by statement; the tool can exit the loop. Upon exiting it's necessary to create a way to communicate the results of the process back to the user, where the information can be used accordingly. The tool ends with the creation of four macro variables.

Table 1 shows the created macro variables.

SAS Macro Variable	**Description**
Keys	This macro variable contains the variables that create a unique by statement
sorted	Populated either Yes or No. Yes if a unique by statement is found, no if it isn't.
Varnam	List of all variables in inputted data set
Sortcnt	Numeric count of number of by variables

Table 1 Generated SAS Macro Variables

The user can employ these variables to sort the data set with the confidence that the by statement outputs a data set sorted uniquely. In the full version of the macro the user has the option to sort inside of the macro by specifying in the parameter that it's desired to do so.

CONCLUSION

Working in a manner that lets SAS do the heavy lifting allows you to work less on items that rely on automated processes and instead focus on more involved and demanding tasks. This method works well when dealing with a multitude of different and changing data across many different studies. However, if you are extremely familiar with your data you may be able to accomplish this task in a more efficient manner.

As the number of observations and variables grow processing time will suffer. Some options to potentially reduce processing time involve using such options like OVERWRITE and compressing the input data set. Also setting the order by which the variables are looped through, and placing items in specified, deliberate order; could resolve the by keys earlier, resulting in efficiencies. The primary motivation behind development was heavily tied to a need for a completely hands off, versatile, and successful method for uniquely sorting. My goal was to show insight into this particular method, but also, to prove how simple repetitive loops can be utilized to solve complex problems.

CONTACT INFORMATION

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

Kevin Miller
Gilead Sciences
Kevin.Miller@gilead.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.