PhUSE 2007

Paper CC02

Merging By Formats

James McGiffen, GlaxoSmithKline, Greenford, UK

ABSTRACT:

Merging multiple small datasets to one large one can have performance issues and merging datasets where there is a set of ranges involves can lead to inaccurate merging.

This paper presents and discusses one possible solution to both these issues by using SAS formats. This paper will detail the advantages and disadvantages of using this method and provides examples of where it can be useful

INTRODUCTION:

For the vast majority of time where a programmer needs to merge two datasets together the basic sort and merge techniques and SQL joins provide all the functionality required.

There are though some situations where both these techniques either have performance issues or they can provide inaccurate and/or results that are difficult to debug.

One solution to these problems is to use SAS formats that will act as a look up table which allows a neat efficient solution to these issues

SO HOW DO YOU MERGE BY FORMATS THEN?

The first thing that needs to be understood is that PROC FORMAT can take a dataset with certain variables and produce a SAS format. The CNTLIN option allows you to define the dataset that SAS will use to create such a format.

Details of the variables required on the dataset can be found in the SAS documentation but in summary you need at least the following variables:

FMTNAME: This is a variable containing the name of the SAS format that you are creating.
START: This should contain the value of the variable that SAS will use to return the value in the Label variable, or in this case the by variable
LABEL: This is the variable that holds the value you want to return or in this case the value of the variable you want to merge on.
TYPE: This is the type of format that you want to produce, details of which can be found in the SAS documentation

So as an example to create a format that holds a dataset of adverse event severity codes then the dataset should have the following structure:

FMTNAME	START	LABEL	TYPE
AESEV	1	Mild	C
AESEV	2	Moderate	C
AESEV	3	Severe	C
AESEV	X	Not Applicable	C
AESEV	other		C

If the following code is run then a format will be created in the work directory (the dataset above has been called aesev). It is important when you are creating datasets for use within proc format that you create an observation that contains the value of 'other' which has a blank label. This will set the value of the variable you are trying to merge to missing if the start variable is not found

```
PROC FORMAT cntlin = aesev library=work; run;
```

SO I HAVE CREATED A FORMAT NOW WHAT DO I DO?

So once the dataset with the values that you want to merge on has been converted to a format you have to use either the input or put functions to retrieve the value:

```
AESEVDECODE = put(AESEV,$aesev.);
```

This will create a variable called AESEVDECODE with the value in the label of the format that corresponds to the start value.

If you wish to create a numeric variable from a format then the input function should be used and again both these functions are well documented.

ADVANTAGES OF THIS METHOD WHEN WORKING WITH LARGE DATASETS:

Although there is extra work and coding within using this method there are obvious advantages when you are working with very large datasets (over 1 million observations) and you wish to add information from numerous small datasets to one large one.

This situation is quite common when creating data warehouses where you have a 'fact' table and wish to add the details of numerous 'dimension' or lookup tables. Typically in this situation the fact table is very large and there is a common key between it and each of the dimension tables. Using a basic sort and merge technique would require the large table to be sorted and merged numerous times. This can be very inefficient and cause issues with runtimes.

Using the format approach you would create formats for all the small dimension tables and then with one pass through the large dataset create all the variables using put and input statements. The author has used this technique to reduce the running time of a SAS program from 5 hours to 40 minutes.

ADVANTAGES WHEN MERGING WITH RANGES:

Although the volume and amount of data that is being captured in clinical trials is increasing the datasets produced are still relatively small compared to other industries, therefore the use of this method detailed above may not be of a great deal of interest to a clinical programming community.

There is though a situation that is very common within the clinical trial world where this technique has real benefits and that is when there is the requirement to merge ranges of values.
This typically happens when dealing with lab, vital signs or ECG data and can have the added complexity that ranges can change depending on the lab taking the measurement, the time the measurement was taken, the sex of the patient and the disease state of the patient.

This type of data can be difficult to handle using proc sort and merge, where typically you will have to delete observations where the date range has not been met and in proc SQL where complex SQL code has to be created and Cartesian products may be produced.

In order to use formats to merge this data it the programmer has to build up the start and end variables with concatenations of the common by variables (e.g. lab test , sex, lab code) and then use the start of the range on the start variable and the end of the range on the end. e.g;:

Lab Parameter	Start date	End Date	High	Low
AST	01Jan01	01Jan02	20	10
AST	02Jan02	01Jan03	30	10
AST	02Jan03		40	30

The table above should be reformatted in order to produce a format with the correct ranges:

FMTNAME	START	END	LABEL	TYPE
ASTRG	AST01Jan01	AST01Jan02	10-20	C
ASTRG	AST02Jan02	AST01Jan03	10-30	C
ASTRG	AST02Jan03	AST*today*	30-40	c
ASTRG	Other	other	.-.	

The variables should be created as follows:
FMTNAME: This should be a unique name for this range of formats
START: This is a concatenation of the common by variables and the start of the range. In this example the lab

parameter code and the start of the range, but other variables (e.g. sex) could be added after the lab code.
END: The same as the start variable except with the end of the range.
LABEL: This is a concatenation of the low and high ranges. Any number of values can be concatenated together in the label variable that can then be separated out on the dataset.

In the code that produces the dataset that you want these values to be merged it is important that you build up the sting to you use in your put statement in the same way as both the start and end dates. E.g.

PARAM	LABDATE	VALUE	DECODE	RANGE	HIGH	LOW
AST	01Jun01	15	AST01Jun01	10-20	10	20
AST	02Jun02	14	AST02Jun02	10-30	10	30
AST	02Jun03	24	AST02Jun03	30-40	30	40
AST	03Jun04	45	AST03Jun04	30-40	30	40

The PARAM LABDATE and VALUE variables will come from a dataset of lab values e.g. the dataset that you want to merge the range onto.

The other variables are created by:
DECODE: is a concatenation of the parameter and the date the value was taken. This value is used in the put statement and is used to interrogate the format and return the range
RANGE: This is the value returned from the format using the statement range = put(decode,$ASTRG);
HIGH and LOW: These variables are created from the range values

Merging ranges using the method above has two main advantages:

The first is that the processing is relatively quick, lab datasets can be relatively large and processing them through numerous sort and merges or a complex SQL can be time consuming.

The second is accuracy, because you are creating a format SAS checks that the ranges that are being produced are distinct. The format will not be created if there are duplicate ranges the programmer therefore knows that the merging is going to be accurate.

CONCLUSION:

Initially using formats to merge data can seem overly complex for simple data merges and this is probably correct , but there are certain situation (dealing with large datasets and merging ranges) where this technique is more efficient and accurate that either proc sort and merge statements and SQL statements..

AUTHOR CAN BE CONTACTED:

James McGiffen
Principal Analyst
GlaxoSmithKline

Phone (+44) 020 8966 2146
Email James.x.Mcgiffen@gsk.com

Brand and product names are trademarks of their respective companies.