PharmaSUG 2018 - Paper QT-15

SAS®sy Data Checks

Amie Bissonett, Syneos Health

ABSTRACT

Every study has data issues and those issues often make their way to the data sets that the statistical programmers use. It is a part of our job to do data checks, and those same types of tools are invaluable when validating data sets. Following are a few useful tips and tricks to make data checking a breeze.

INTRODUCTION

SAS® has created stored code to make it easy to bring up code that you use often, but for those that still prefer coding and not clicking, these tips will bring that code up with a few quick keystrokes. The examples given in this paper show the creation and use of abbreviation macros, and using them to bring up quick and easy data checking code to use on the fly.

ABBREVIATION MACROS

Abbreviation macros in SAS® store code that can be quickly brought up in the program editor with just a few quick keystrokes as shown in Figure 1. The code can be set up with placeholders for instance specific information, like data set name, variable names, and more. Note that the code added to the abbreviation macro does not have to be a macro, it can be any free text.

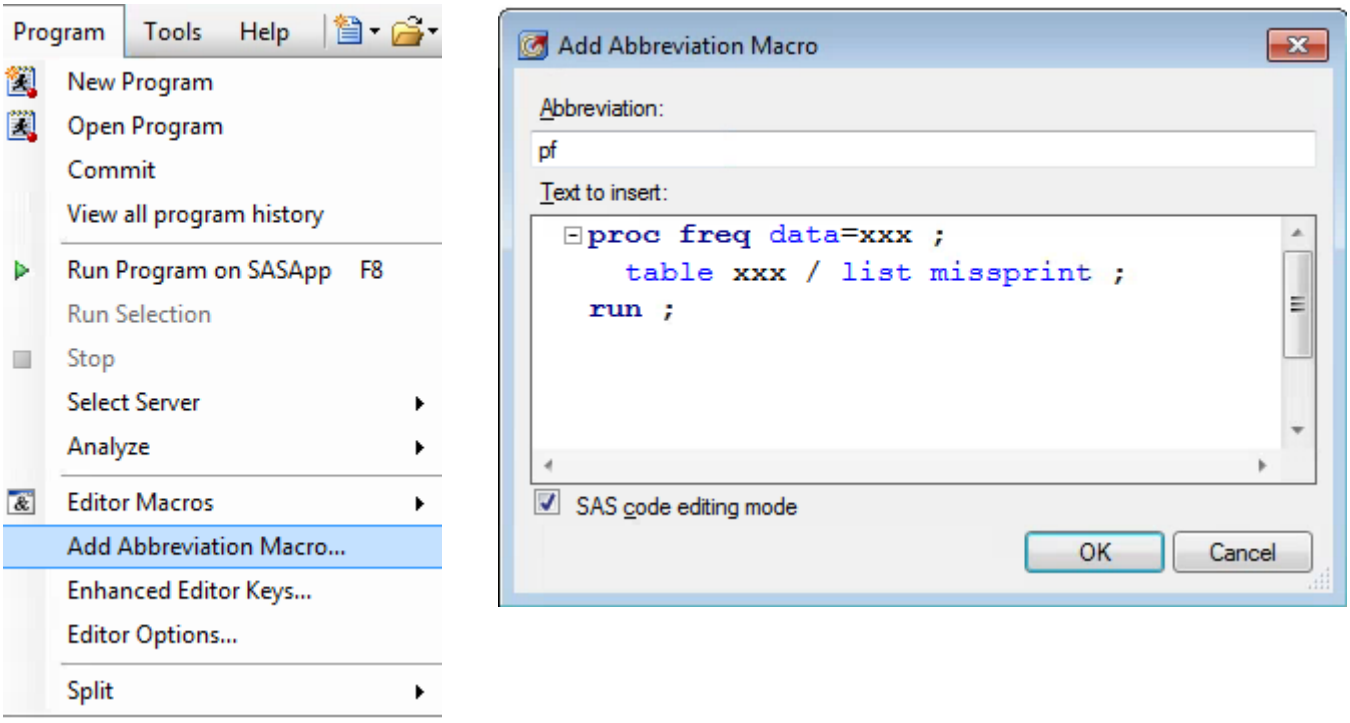


Figure 1. Adding an abbreviation macro

In the program editor, type the abbreviation and the beginning of the code will display in a pop-up box. To bring the code in to the editor, press enter. See Figure 2. To ignore the code, simply continue typing. The 'xxx' placeholders can then be updated to produce the desired code on any data set and variables.

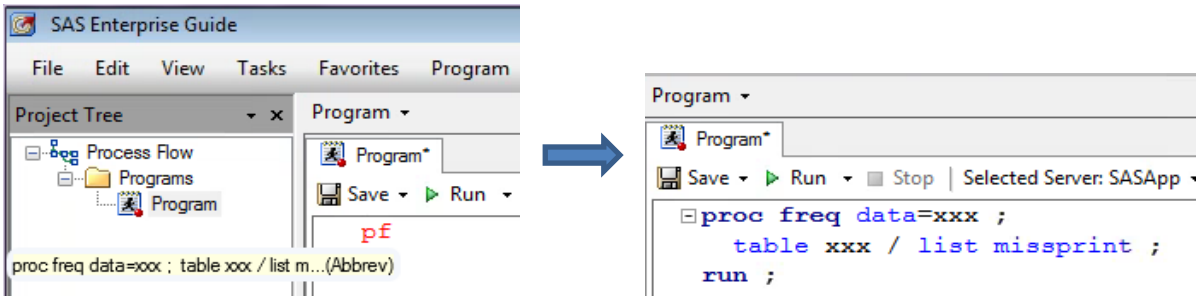


Figure 2. Implementing an abbreviation macro for PROC FREQ.

Another example of a useful abbreviation macro is PROC SORT, shown in Figure 3. This abbreviation sets the by variable to USUBJID since it is often the first sort variable used, and just like a placeholder, can be easily replaced with the required variables.

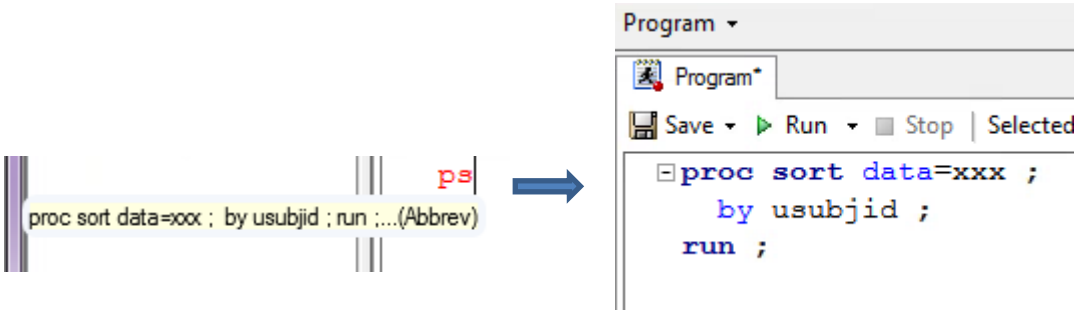


Figure 3. Implementing an abbreviation macro for PROC SORT

COMPARING DATA

Programming and validating data sets often requires the comparison of two data sets. Two useful ways to compare data sets are PROC COMPARE and the use if the in= option in a DATA STEP merge. These are both additional useful abbreviation macros to set up for quick recall in interactive programming.

PROC COMPARE has several options available and setting these up in an abbreviation macro has them stored right at your fingertips, as shown in Figure 4.

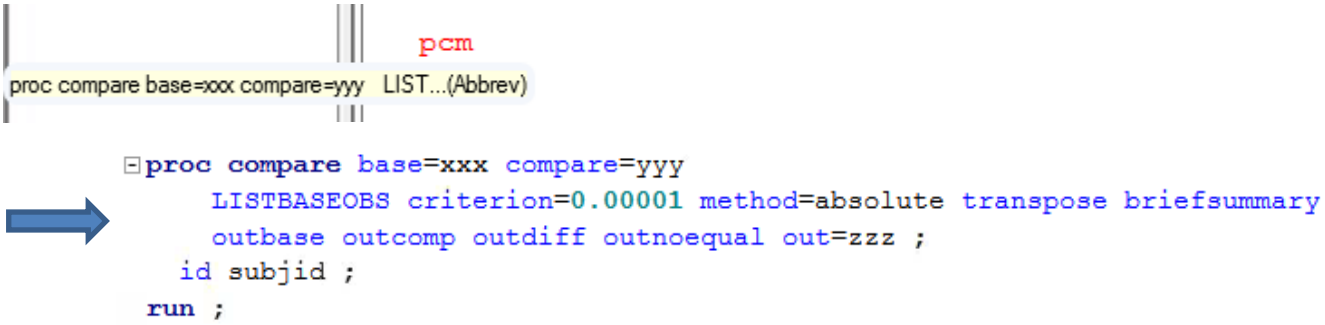
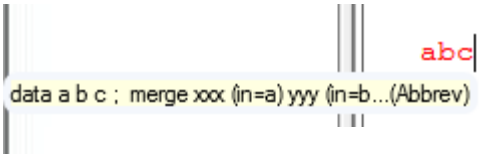


Figure 4. Implementing an abbreviation macro for PROC COMPARE

PROC COMPARE is a great tool to compare the values from one data set to another. After updating the code from the abbreviation macro, a simple example of running a PROC COMPARE is as follows:

```
proc compare base=dfsdtm compare=mydf ;
  id usubjid dfcat dfscat dfobj dfeval dftestcd ;
run ;
```

What about when the PROC COMPARE shows that there are records in one data set that are not in the other data set based on the BY or ID variables? When the ID variables don't match, the mismatching records can be discovered by using a DATA STEP merge with the in= option. The below example will output two data sets with the observations that don't match on the BY variables. Data set A will contain the records from data set DFSDTM that do not match to MYDF, and data set B will contain the records from data set MYDF that do not match to DFSDTM, and finally, all matching observations will be in data set C. Figure 5 shows the use of another abbreviation macro and the code after updating the placeholder code.




```
data a b c ;
   merge dfsdtm (in=a) mydf (in=b) ;
   by usubjid dfcat dfscat dfobj dfeval dftestcd ;
   if a & ^b then output a ;
   if ^a & b then output b ;
   if a & b then output c ;
 run ;
```

Figure 5. An abbreviation macro to quickly filter out mismatching records

PROC SQL FOR DATA CHECKS

There are many tools available in SAS® for doing data checks, including the DATA STEP. Sometimes multiple DATA STEPs are needed to solve one task. PROC SQL has the power of subqueries to perform a multitude of tricky tasks.

This example uses a subquery to select all records from one data set for subjects that meet a criteria in another data set. The subquery selects the unique USUBJID values from HADURTN where HADURHR > 168. Then the outer query subsets ADAM.ADHACHE using that list of subjects:

```
proc sql ;
  create table cklong as
    select usubjid, adt, haid, hadur, hadur / 60 as hadurhr
      from adam.adhache
        where usubjid in (
          select distinct usubjid
            from hadurtn
              where hadurhr > 168
        )
        order by usubjid, adt, haid
  ;
quit ;
```

CONCLUSION

Programming data sets and checking that data throughout the process can be time consuming. A few quick tips were presented to help you program more efficiently and quickly get the results you need.

CONTACT INFORMATION

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

Amie Bissonett
Syneos Health
amie.bissonett@syneoshealth.com