**Paper CC20**

# Keeping your macro robust:

# Does your dataset exist, does your variable exist?

Lara E.H. Guttadauro, MS, i3 Statprobe, West Chester, OH


## ABSTRACT

Your macro program was working great. Now you're missing datasets or variables. Ugh! What to do? This paper addresses both of these situations and presents code to check for missing datasets and variables and creates them if needed to help keep your macro robust.


## MOTIVATION

As a programmer, one of the most rewarding things for me is writing a macro program that creates many outputs. It takes some time to get the macro running correctly on the front end. But creating several outputs in several minutes on the back end is great!

I had written a macro that created about 20 outputs and then additional outputs were needed that were subsets of the original data. Suddenly my macro had all kinds of errors and wasn't creating the correct results. My first thought was to make another version of the macro taking into account the subsetting. On second thought, the macro was pretty complicated and I didn't want to have to maintain multiple versions of it. I had to figure out how to deal with missing datasets and variables. The results are below.

## MISSING DATASETS

A procedure that was supposed to create a dataset containing p-values was not working correctly since there were not enough observations to meet the conditions the procedure needed to create the p-values and corresponding dataset. Not a problem unless you reference the missing dataset later in your code. SAS® really doesn't like when that happens. You will need to check to see if the dataset exists and if not, create the dataset and the variables that were supposed to be in it. The EXIST function returns a 1 if the dataset exists and a 0 if the dataset does not exist.

```
** Create dataset XXXX if it does not get created in the procedure above **;
data _null_;
  RC=exist('XXXX', 'data');
  call symput ('CHK_DATA', RC);
run;

** Create empty dataset if needed **;
%macro CREATE_DATA;
  %if &CHK_DATA=0 %then %do;
    data XXXX;
      VAR1=.;
      VAR2=.;
      VAR3=.;
      output;
    run;
  %end;
%mend CREATE_DATA;

%CREATE_DATA;
```

## MISSING VARIABLES

In this situation, the procedure created the necessary dataset, but was unable to create all the expected variables because of a lack of data. The variables were referenced later in the code. Of course, SAS doesn't like missing variables either. You will need to check to see if the variables exist and if not, create them.

```
** Create variables (VAR1 & VAR2) if they were not created in the YYYY dataset **;
ods output  position=VARLIST;
  proc datasets;
    contents  data=YYYY  varnum;
  run;
ods output close;
quit;

** Retain variables to create a list of all variables **;
data VARLIST2  (keep=CHK_VAR);
  set VARLIST (keep=MEMBER NUM VARIABLE);
  by MEMBER NUM;
  length CHK_VAR $500.;
  retain CHK_VAR;
  if first.MEMBER then CHK_VAR=VARIABLE;
    else CHK_VAR=upcase(compbl(CHK_VAR||' '||VARIABLE));
  if last.MEMBER then output;
run;

** Merge VARLIST onto YYYY dataset **;
** Check for needed vars & create if they do not exist **;
data YYYY (drop=CHK_VAR);
  merge YYYY VARLIST2;
  if index(CHK_VAR, 'VAR1')=0 then VAR1=.;
  if index(CHK_VAR, 'VAR2')=0 then VAR2=.;
run;
```

## CONCLUSIONS

By applying the ideas and code above, I was able to keep and maintain just one macro program that was robust enough to handle missing datasets and variables.

## CONTACT INFORMATION

Lara E.H. Guttadauro, MS
i3 Statprobe
9050 Centre Pointe Drive
West Chester, OH 45069
Email: Lara.Guttadauro at i3statprobe.com