Paper CC-19

Three Easy Ways around Nonexistent or Empty Datasets

Spencer Childress, Rho, Inc., Chapel Hill, NC
Brandon Welch, Rho, Inc., Chapel Hill, NC

ABSTRACT

Nothing derails a program quite like a nonexistent dataset. Statistical programmers often merge summary statistics with p-values, but if the data aren't present, the p-value and its dataset won't generate. We present three simple methods in a macro that check the existence of a dataset and return an observation count. The first method combines SAS® File I/O code with data from the SASHELP library. The second needs only SAS File I/O code. The third also makes use of SAS File I/O code with PROC SQL. If the existence-check returns false or the observation count returns 0, then the macro creates a dummy dataset with one observation, ready to merge. Regardless of the method, any programmer can tackle those nonexistent datasets. The techniques we present offer a good overview of basic data step programming and macro processing appropriate for all levels of SAS capability. While this article targets a clinical computing audience, the techniques apply to a broad range of computing scenarios.

INTRODUCTION

This macro checks a dataset's existence and number of observations with three different methods. It also outputs a dataset of the same name given a nonexistent or empty dataset. Rather than hard-coding to combat insufficient data, this macro makes our code robust. Often, particularly with categorical variables, p-values aren't produced because only one level of a variable or set of variables exists. This situation is troublesome because, as in the clinical programming industry, summary tables must match table shells, regardless of the data.

Much of the macro involves SAS I/O processing outside the data step, with the macro function %SYSFUNC, which allows the use of data step functions outside the data step. SAS File I/O processing is the reading and writing of SAS data files – data sets, catalogs, index files, etc. With SAS File I/O, we can open and close datasets and return their attributes, including existence and number of observations. Below, we explain each step of the macro, and detail the techniques that go into each method.

STEP-BY-STEP

1. At the beginning of the macro, we localize and initialize three macro variables: &InDat, &Exist, and &NumObs.
2. The user assigns values to the macro variable &InDat using the %let statement. &InDat denotes the name of the dataset whose existence will be verified.
3. By default, at the beginning of each method, &Exist and &NumObs are set to "No" and "0", respectively.
4. Each method uses a different combination of existence/observation checks.
5. Given a negative existence check or a 0-observation dataset, the macro finishes by creating a single-observation dataset with a missing p-value.

METHOD 1

SASHELP.VMEMBER is an automatic SAS view that contains the name of each library, the name of each dataset within each library, the type of dataset, and various descriptive variables. We are only interested in the data sets residing in the the current WORK directory. Therefore, when accessing these data sets, we subset to LIBNAME = "WORK" AND MEMTYPE = "DATA".

For the existence check, if the dataset's name (`MEMNAME = "&InDat"`) is in VMEMBER, then function CALL SYMPUT assigns 'Yes' to &Exist.

Given a positive existence check, the macro returns the number of observations with the ATTRN, function - a function that returns various numeric dataset attributes. ATTRN's parameter of interest is NOBS, which returns the number of

observations. However, the dataset must be opened. With %SYSFUNC, most data step functions can be used in macro code. In this case, we use %SYSFUNC in conjunction with the OPEN data step function to assign a value to DSNId. With ATTRN we assign the number of observations to DSObs. Finally we use CLOSE to close the dataset. Given an existent dataset, we now have its number of observations.

METHOD 2

Again with %SYSFUNC, Method 2 checks existence with data step function EXIST, which verifies the existence of a SAS library member and defaults to member type DATA.

Given existence, we check the number of observations as we did in Method 1, namely with `%SYSFUNC(ATTRN(&DSNId, nobs))`.

METHOD 3

Method 3 verifies existence exactly as Method 2 does, with `%SYSFUNC(EXIST(&InDat))`.

To return the number of observations, we use the automatic PROC SQL variable &SQLObs. &SQLObs contains the number of rows executed by an SQL procedure statement; in our case, it is assigned the number of rows output by the SELECT statement.

CONCLUSION

P-value computation is a ubiquitous and important statistical task that checks the extremity of a test statistic. Different types of p-value have different data requirements for output, and in general, a lack of data can prevent the creation of a dataset, whether for a p-value or not. This macro checks a dataset's existence and number of observations with three different methods, and outputs a dataset of the same name given a nonexistent or empty dataset. It is intended to be robust to data, to prevent hard-coding rows into an output table.

CONTACT INFORMATION

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

Spencer Childress
Rho, Inc
6330 Quadrangle Dr., Ste. 500
Chapel Hill, NC 27517
Phone: 919-595-6638
Fax: 919-408-0999
Email: Spencer_Childress@rhoworld.com
Web: www.rhoworld.com

APPENDIX

%macro *DatExist*;

```
  %local InDat Exist NumObs;
  %let InDat  = pval;

  %*METHOD 1 - SASHELP LIBRARY + SAS I/O;
  %let Exist  = No;
  %let NumObs = 0;

  DATA _null_;
    set sashelp.vmember(where = (libname = "WORK" and memtype = "DATA"));
    if upcase(memname) = "%upcase(&InDat)" then call symput("Exist",'Yes');
  RUN;

  %if &Exist = Yes %then %do;
    %let DSNId = %sysfunc(open(&InDat));
    %let DSObs = %sysfunc(attrn(&DSNId.,nobs));
```

```
    %let rc = %sysfunc(close(&DSNId.));
    %let NumObs = &DSObs.;
  %end;

  %put ****************METHOD 1 - SASHELP LIBRARY + SAS I/O****************;
  %put EXIST: &Exist;
  %put NUMBER OF OBS: &NumObs;
  %put ********************************************************************;

  %*METHOD 2 - SAS I/O;
  %let Exist  = No;
  %let NumObs = 0;

 %if %sysfunc(exist(&InDat)) %then %let Exist = Yes;

  %if &Exist = Yes %then %do;
    %let DSNId = %sysfunc(open(&InDat));
    %let DSObs = %sysfunc(attrn(&DSNId.,nobs));
    %let rc = %sysfunc(close(&DSNId.));
    %let NumObs = &DSObs.;
  %end;

  %put;
  %put ****************METHOD 2 - SAS I/O *********************************;
  %put EXIST: &Exist;
  %put NUMBER OF OBS: &NumObs;
  %put ********************************************************************;

  %*METHOD 3 - SAS I/O + PROC SQL;
  %let Exist  = No;
  %let NumObs = 0;

 %if %sysfunc(exist(&InDat)) %then %let Exist = Yes;

  %if &Exist = Yes %then %do;
    %*METHOD 3 - PROC SQL;
    PROC SQL noprint;
      select *
      from &InDat;
    QUIT;

    %let NumObs = &SQLObs;

  %end;

  %put;
  %put ****************METHOD 3 - SAS I/O + PROC SQL**********************;
  %put EXIST: &Exist;
  %put NUMBER OF OBS: &NumObs;
  %put ********************************************************************;

  %if &Exist = No or
      &NumObs = 0 %then %do;
    DATA pval;
      prob = .;
    RUN;
  %end;

%mend;
```