SESUG Paper 184-2018

Knowing What You've Got

Stephanie R. Thompson, Datamum

ABSTRACT

Have you ever needed to use data from a large collection of files that you know nothing about? If so, this paper is for you. PROC FREQ is a handy way to look at the contents of your data but sometimes you just want to generate a summary of a bunch of tables all at one time. This paper uses a case study of over 700 tables from several databases and one simple, easy, and reusable method is presented. PROC DATASETS, PROC SQL, the MACRO facility, and even PROC IML are used. A dataset of table information and summaries of the populated variables in each table are the end result.

INTRODUCTION

Much of the data used in day to day work is familiar and well understood. Maybe it is sales, inventory, or population data that we use to generate routine reports. We know what variables are there as well as their type. Using this data is just part of the everyday landscape. We know what to expect in the results and can tell if something does not look right and we know were to look to see if there is an underlying issue.

At other times, we are presented with a new data source that needs to be incorporated into our regular work. If it is one table or dataset, running PROC FREQ, PROC UNIVARIATE, and printing some observations to the output window can tell us what we need to know to start using the data. We may need to ask a data steward or subject matter expert some questions related to the specifics of certain variables but it is not an overwhelming task.

However, what do you do when you are given a collection of 700 tables across several databases to make sense of and there is no data dictionary or in-house expert? How do you go about finding the things you need in such a vast amount of information? Is there even anything you need in those tables? The approach below will walk you though one way to get an understanding of a large selection of data sources to help determine what you have so you can use it. The end result is a dataset that you can refer to that has some basic information about the size of the tables as well as individual datasets for each table that contain information regarding each variable and how many of the observations are missing or not missing. All of the code is listed in Appendix A while excerpts are shown in the body of the paper.

GETTING SET UP

Setting up the libnames is the initial task in the process. Since operating systems vary, the code will use a generic reference to the data sources - one of which is a collection of SAS® datasets. Here we will use three libraries:

```
/* establish libraries */
libname DB_TEN - collection of SAS datasets on a fie share
libname GROUP1 – Oracle database residing on a server
libname st – where I want to store this information
```

Once the libnames are set, use PROC DATASETS to get a quick look at what you have:

```
/* inventory tables */
proc datasets lib=DB_TEN details;
run;
quit;
```

```
proc datasets lib=GROUP1 details;
run;
quit;
```

EVALUATE THE SIZE OF THE CHALLENGE

The first step in looking at the tables is to see how many of them actually contain data. Many databases from vendor solutions contain all of tables for the full suite of options. Your organization may not be using the full functionality so only some tables are populated. The first step is to generate a dataset that contains information on the tables in an easy to understand format.

The MACRO manyfiles will go through each dataset listed on the if statement on the DATA Step that creates infosee. The PROC SQL gets the count of datasets to be used in the do loop.

Here the information is compiled for the libnames of interest.

```
/* get counts of rows in tables */
data infosee;
set sashelp.vtable;
if libname in ('DB_TEN', 'GROUP1') then output;
run;

proc sql noprint;
select count(*) into :tablecount
from infosee;
quit;
```

The macro iterates for each table in the libnames in the DATA Step.

```
%macro manyfiles;

%do i=1 %to &tablecount;
      data _null_;
      set infosee;
      if  _n_ = &i;
      call symput('ln', trim(libname));
      call symput('mem', trim(memname));
      run;

      %put &ln;
      %put &mem;

      proc sql;
      create table newcount as
      select count(*) as rows
      from &ln..&mem;
      quit;

      data newcount;
      length ln $8. mem $32. ;
      set newcount;
      ln = "&ln";
      mem = "&mem";
      run;

      proc append base = st.mytableinfo data=newcount;
      run;
```

```
%end;
%mend;
%manyfiles;
```

FOUNDATION FOR THE NEXT MACRO

Once you have looked at the st.mytableinfo dataset, this part of the program will create a dataset for each table or dataset that you want to see how much data is missing by variable. The DATA Step creating the dataset infosee2 can contain exclusions if you identify tables that are not if interest or contain no observations. Infosee2 could also be generated using PROC SQL to exclude zero observation tables from st.mytableifno if you would like. For the purposes of the project at hand, only the table - receipt - was excluded since it did not contain readable data.

```
data infosee2;
set sashelp.vtable;
if memname ne 'RECEIPT';    /* can exclude tables if you want */
if libname in ('DB_TEN', 'GROUP1') then output;
run;

proc sql noprint;
select count(*) into :tablecount2
from infosee2;
quit;
```

THE WORKHORSE MARCO

The manyiml MACRO is the core of this program. This is where each variable is evaluated to see is how many observations are missing and not missing for each data source identified in the step above. Here character and numeric variables are evaluated separately.

```
%macro manyiml;

%do i=1 %to &tablecount2;

      data _null_;
      set infosee2;
      if  _n_ = &i;
      call symput('ln', trim(libname));
      call symput('mem', trim(memname));
      run;

      %put &ln;
      %put &mem;

      data imltest;
      set &ln..&mem;
      run;

      proc iml;
      use imltest;
      read all var _NUM_ into x[colname=nNames];
      n = countn(x,"col");
      nmiss = countmiss(x,"col");

      read all var _CHAR_ into x[colname=cNames];
      close imltest;
```

```
            c = countn(x,"col");
            cmiss = countmiss(x,"col");

            /* combine results for num and char into a single table */
            Names = cNames || nNames;
            rNames = {"    Missing", "Not Missing"};
            cnt = (cmiss // c) || (nmiss // n);
            create &ln.&mem.counts from cnt [rowname=rNames colname=Names ];
            append from cnt [rowname=rNames];
            close &ln.&mem.counts;
            quit;

            data &ln.&mem.counts;
            length ln $8. mem $32. ;
            set &ln.&mem.counts;
            ln = "&ln";
            mem = "&mem";
            run;

    %end;
    %mend;
    %manyiml;
```

This macro will generate one dataset for each table or dataset in infosee2. Additional steps can be added to summarize the data from each of those datasets using approaches similar to those in this paper. The task at hand did not require this.

THE END RESULTS

This program generates two different sets of information. One is the st.mytableinfo dataset and the other is the datasets generated by the manyiml macro.

SUMMARY DATASET ST.MYTABLEINFO

The program generates one dataset, st.mytableinfo, that contains a list of each dataset with the associated libname and the number of rows in each. This information can be helpful as it lets you know which tables do not contain any information. Maybe you can ignore these or maybe you need to investigate why. It will also give you a feel for the overall magnitude of the tables. You can quickly get a feel for what may be a transactional table, a fact table, or a key table based on number of rows.

VIEWTABLE: Work.Sashelptables

	ln	mem	rows
1	SASHELP	AACOMP	1544
2	SASHELP	AARFM	61
3	SASHELP	ADSMSG	426
4	SASHELP	AFMSG	1090
5	SASHELP	AIR	144
6	SASHELP	APPLIANC	156
7	SASHELP	ASSCMGR	402
8	SASHELP	BASEBALL	322
9	SASHELP	BEI	23905
10	SASHELP	BMT	137
11	SASHELP	BUY	11
12	SASHELP	BWEIGHT	50000
13	SASHELP	CARS	428

PROC FREQ provides a summary of the distribution of rows in the collection of data sources. The code used is below.

```
proc freq data =st.mytableinfo;
tables rows * ln / nocum nocol norow nopercent ;
run;
```

The SAS System

The FREQ Procedure

Table of rows by ln

rows	ln	
Frequency	SASHELP	Total
0	17	17
1	7	7
2	3	3
3	1	1
4	7	7
5	4	4
6	10	10
7	3	3
8	6	6
9	1	1
10	5	5

%MANYIML DATASETS

Figure 1 presents a partial look at one of the output datasets generated by the manyiml MACRO. There is a column for each variable in the dataset with a row corresponding to the count of rows in the dataset that are missing or not missing. Both character and numeric variables are shown together.

ln	mem	rNames	ADJ_ALT_ID	ADJ_DECISION	ADJ_PREFIX	COR_ADDRESS	COE_UID_EMAIL	ADJ_USE_PDF	ADJ_ROOT
GROUP1	ADJUSTMENTS	Missing	12984	170	0	5892	5894	0	12982
GROUP1	ADJUSTMENTS	Not Missing	4	12818	12988	7096	7094	12988	6

Figure 1. Partial View of %manyiml Generated Dataset

As noted earlier, each of these datasets can be programmatically run through to count variables or to see how many are completely missing. You could even calculate a percentage missing if you wanted to. There are many options in this regard depending on your need.

CONCLUSION

Hopefully this paper provided some tips on how to easily summarize information on large collections of tables or datasets. These can be local or stored in an external database such as a data warehouse. The code is flexible and generates much of the needed information through the use of macros and macro variables generated with RPOC SQL. You just need to know where you want to look.

REFERENCES

Wicklin, Rick. 2011. “Count the number of missing values for each variable” from “The DO Loop” blog. Accessed August 6, 2018. https://blogs.sas.com/content/iml/2011/09/19/count-the-number-of-missing-values-for-each-variable.html.

Wicklin, Rick. 2012. “Reading ALL variables INTO a matrix” from “The DO Loop” blog. Accessed August 5, 2018. https://blogs.sas.com/content/iml/2012/01/16/reading-all-variables-into-a-matrix.html.

CONTACT INFORMATION

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

Stephanie R. Thompson
Datamum
901-326-0030
Stephanie@datamum.com
http://www.datamum.com

APPENDIX A – FULL CODE

```
/* establish libraries */

libname DB_TEN - collection of SAS datasets on a share
libname GROUP1 – Oracle database residing on a server
libname st – where I want to store this information

/* inventory tables */
proc datasets lib=DB_TEN details;
run;
quit;

proc datasets lib=GROUP1 details;
run;
quit;

/* get counts of rows in tables */

data infosee;
set sashelp.vtable;
if libname in ('DB_TEN', 'GROUP1') then output;
run;

proc sql noprint;
select count(*) into :tablecount
from infosee;
quit;


%macro manyfiles;

%do i=1 %to &tablecount;

data _null_;
set infosee;
if  _n_ = &i;
call symput('ln', trim(libname));
call symput('mem', trim(memname));
run;

%put &ln;
%put &mem;


proc sql;
create table newcount as
select count(*) as rows
from &ln..&mem;
quit;

data newcount;
length ln $8. mem $32. ;
set newcount;
ln = "&ln";
```

```
mem = "&mem";
run;

proc append base = st.mytableinfo data=newcount;
run;


%end;
%mend;

%manyfiles;


data infosee2;
set sashelp.vtable;
if memname ne 'RECEIPT';    /* can exclude tables if you want */
if libname in ('DB_TEN', 'GROUP1') then output;
run;

proc sql noprint;
select count(*) into :tablecount2
from infosee2;
quit;



%macro manyiml;

%do i=1 %to &tablecount2;

data _null_;
set infosee2;
if  _n_ = &i;
call symput('ln', trim(libname));
call symput('mem', trim(memname));
run;

%put &ln;
%put &mem;

data imltest;
set &ln..&mem;
run;

proc iml;
use imltest;
read all var _NUM_ into x[colname=nNames];
n = countn(x,"col");
nmiss = countmiss(x,"col");

read all var _CHAR_ into x[colname=cNames];
close imltest;
c = countn(x,"col");
cmiss = countmiss(x,"col");

/* combine results for num and char into a single table */
Names = cNames || nNames;
```

```
rNames = {"    Missing", "Not Missing"};
cnt = (cmiss // c) || (nmiss // n);
create &ln.&mem.counts from cnt [rowname=rNames colname=Names ];
append from cnt [rowname=rNames];
close &ln.&mem.counts;
quit;

data &ln.&mem.counts;
length ln $8. mem $32. ;
set &ln.&mem.counts;
ln = "&ln";
mem = "&mem";
run;

%end;
%mend;

%manyiml;

proc freq data =st.mytableinfo;
tables rows * ln / nocum nocol norow nopercent ;
run;
```