

See What You Are Missing: Finding Missing Data Records

Kim Truett, KCT Data, Inc., Alpharetta, GA

ABSTRACT

In relational databases, a common question asked of the database manager is “Are all records present in the database”? Ensuring that no database contains extra records and that each database contains the expected records can be a very time-consuming process. However, this verification step is critical, since missing data will be detected during any subsequent usage of the data.

To facilitate this type of review, this paper presents a generic macro which uses a graphical representation of the data records present in the relational database. The macro produces one “grid per timepoint” indicating the presence or absence of records within every dataset. If at least one record is present, then the grid shows the actual number of records. Visual review of this grid allows for instant identification of extra and missing data.

INTRODUCTION OF PROBLEM

In relational databases, the database manager is often asked to confirm that all expected records are present in a database. This can be a time-consuming process – ensuring that no database contains extra records and that each database does contain the expected records is not as simple and straightforward of a process as it sounds. However, this verification step is critical since missing data will be detected during any subsequent usage of the data.

OVERVIEW OF SOLUTION

To facilitate this review, I developed a generic macro that outputs a graphical representation of the data records present in a database. The macro produces “one grid per timepoint” indicating the presence or absence of a record within every dataset for each unit. A unit might be a patient in a clinical trial or a truck for shipping logs or a client for corporate data. This graphical format simplifies review and allows for rapid identification of all missing records.

In order to maximize flexibility and create re-usable code, my solution needed to:

1. Determine all datasets within any LIBNAME assigned.
2. Determine the presence or absence of records within each dataset.
3. Output the results in an easily reviewed fashion.

DETAILS OF PROGRAM

A. DEFINE LIBNAME.

The first step defines the libname for the library to be checked and defines a macro variable that specifies the variable (or variables) which identify the records and which link the datasets in the database (subsequently referred to as the ‘ID variable’, although it could be, and often is, several variables). This allows the macro to be run without modification, independent of the variable names. Several titles are also defined. In my sample code, SITE and PATIENT are my ID variables which link the datasets in the database.

```
LIBNAME SAMPLIB 'C:\PROT01\DATA';

%LET REL_ID=SITE PATIENT;

DATA _NULL_; CALL SYMPUT('PRTDT', LEFT(PUT(TODAY()),WORDDATE.)); RUN;
TITLE1 "NEW DRUG PROTOCOL 1";
TITLE2 "DATE PRODUCED: &PRTDT";
TITLE4 'VERIFY CORRECT DATA RECORDS FOR EACH PATIENT AND TIMEPOINT';
```

B. CREATE A DATASET WITH ALL EXPECTED ID VARIABLES AND TIMEPOINT NUMBERS.

This step primes a dataset, ALL, with a list of all ID variable values and timepoints. This example uses a database named VISITS that contains exactly this information, but, depending on your data, any database that should have a record for every ID variable at every timepoint (like a primary analysis variable, a record of patient visits, a list of customer transactions, etc.) can be used to prime the dataset. This information is also printed, so that it can be reviewed for completely missing timepoints.

```
PROC SORT DATA=SAMPLIB.VISITS (KEEP=&REL_ID TIMEPOINT) OUT=ALL; BY &REL_ID
TIMEPOINT;
```

```
* Create table of ID variable by timepoint number to check for completely missing
timepoints;
```

```
PROC FREQ DATA=ALL FORMCHAR(1,2,7)= '   ';
  TABLES &REL_ID*TIMEPOINT / NOPERCENT NOROW NOCOL;
```

C. CREATE A DATASET CONTAINING ALL DATASET NAMES IN THE ASSIGNED LIBNAME.

The next step creates a dataset, ALLDS, containing one record for each SAS dataset in our test library. There are numerous ways to generate this. I've chosen to use a Proc Contents with the NOPRINT option and an OUTput dataset (which I call MEMLIST). Data step processing is then used to keep each unique MEMNAME and assign a counter (DSN) which increments and tallies the total number of datasets. The final total number of datasets is then stored in a global macro variable.

```
* Run Proc contents and output (no print) results;
PROC CONTENTS DATA=SAMPLIB._ALL_ NOPRINT OUT=MEMLIST;

* Create data set containing one record for each data set in library;
DATA ALLDS;
  SET MEMLIST(KEEP=MEMNAME) END=LASTONE;
  BY MEMNAME;
  IF LAST.MEMNAME THEN DO;
    DSN+1;
    OUTPUT;
  END;
  IF LASTONE THEN CALL SYMPUT("NUMDS",DSN);
RUN;
```

Thus my dataset, ALLDS, is created with two variables: MEMNAME and DSN and one macro variable (NUMDS) is created.

D. TIMEPOINTS ARE NOT NEEDED FOR YOUR DATA

In some databases, some datasets may have timepoints and others may not (for example, demography data would not). Because this macro requires TIMEPOINT, the first step within the macro below assigns a timepoint number to datasets without TIMEPOINT. This can be assigned two ways. First, a unique value, which is not a valid timepoint number, could be set, and these datasets will appear in a separate grid. Or a valid timepoint number (i.e. Timepoint 1 or last timepoint) can be chosen, with the understanding that TIMEPOINT is fabricated. This is handled here by creating a mock timepoint (in this case equal to 1) to reduce the amount of output, but it could also have been set to 0 to list these datasets out first. Note that the TIMEPOINT variable reference in the CHECKDS macro must match the format (either character or numeric) of the TIMEPOINT variable found in the datasets.

For some databases, there may never be timepoints involved. If your data does not contain timepoints, a simplified version of this macro could be used to output a single grid of the data in the database

E. DETERMINE THE PRESENCE OR ABSENCE OF EACH ID VARIABLE WITHIN EACH DATASET FOR EACH TIMEPOINT.

The first macro is then invoked. This macro runs one time for each dataset named in the observations in ALLDS. For each observation, it creates a new macro variable (NAME), and sets it equal to the value of MEMNAME and then calls the CHECKDS macro.

Macro 1: DSNAME:

```
%MACRO DSNAME();
  %DO I=1 %TO &NUMDS;
    DATA _NULL_;
      SET ALLDS(WHERE=(DSN=&I));
      CALL SYMPUT('NAME',MEMNAME);
    RUN;
    %CHECKDS(NAME=&NAME);
  %end;
%MEND;
```

When the second macro is invoked, the dataset name contained in the macro variable (NAME) becomes the actual dataset name within the code, thus this macro will now run once for every dataset within the library, without ever specifying the name of any datasets.

The macro determines which ID variables and timepoints have records within the dataset. The macro creates a new variable with the same name as the dataset, and increments that new variable by '1' for every ID variable / timepoint combination. For example, for a demographic dataset (DEMO), the output would be the ID variables, TIMEPOINT, and a new variable named DEMO, where DEMO is set to '1' (assuming that there is just one demography record for that ID variable). This small dataset is then merged into a larger dataset named "ALL". For a second dataset, HISTORY, perhaps there are 4 records with the same ID variable, so the variable HISTORY is incremented to 4. When HISTORY is merged into the dataset ALL, ALL will now contain the reference ID, TIMEPOINT plus the variables DEMO, and HISTORY which correspond to the datasets in the database, and each of which has been set equal to the number of records found for that reference ID and timepoint. In our example, DEMO = 1 and HISTORY =4.

Macro 2: CHECKDS

```
%MACRO CHECKDS(NAME);

* Add mock timepoint for data sets without the variable timepoint;

DATA &NAME;
    TIMEPOINT = 1;
    SET SAMPLIB.&NAME;
PROC SORT; BY &REL_ID TIMEPOINT;

* Set variable with name corresponding to data set name for ;
* each observation in the data set.                                ;

DATA &NAME;
    SET &NAME(KEEP=&REL_ID TIMEPOINT); BY &REL_ID TIMEPOINT;
    IF FIRST.TIMEPOINT THEN &NAME = 0;
    &NAME + 1;
    IF LAST.TIMEPOINT THEN OUTPUT;
PROC SORT; BY &REL_ID TIMEPOINT;

* Merge newly created data set with other data sets previously checked.;

DATA ALL;
    MERGE ALL &NAME;
    BY &REL_ID TIMEPOINT;
PROC SORT; BY &REL_ID TIMEPOINT;

RUN;

%MEND;
```

F. PRINT THE RESULTING DATABASE.

After the macro runs for all of the datasets, ALL is printed thereby producing the grid.

```
* Sort resulting data and print;

PROC SORT DATA=ALL; BY TIMEPOINT;
PROC PRINT DATA=ALL; BY TIMEPOINT; PAGEBY TIMEPOINT;
```

IMPROVEMENTS AND CONCLUSION

Ultimately, this tool presents a simplified solution to a previously difficult task of identifying missing data within a database.

COMPLETE SAS CODE:

To use this code, you must modify:

- The LIBNAME SAMPLIB reference ('C:\PROT01\DATA' is used in this code) - to match the library to be checked;
- The REL_ID variable or variables (SITE and PATIENT are used in this code) – to match the variable or variables that link the datasets in the library to be checked;

- The dataset used to create the ALL dataset (a dataset named VISITS is used in this code) – to match a dataset which contains a record for every ID variable at every timepoint;
- The TIMEPOINT variable (TIMEPOINT is used in this code) – to match a variable in your datasets that provides an axis along which records can be compared. Most likely, this would be a variable relating to time – visit number, transaction date, login time – but you could also use a variable relating to location – such as region, state, store number – if that would present a more understandable picture of your data. The only restriction is that this variable must appear in the dataset used to create the ALL dataset.

The code:

```

OPTIONS LS=120 PS=85 SYMBOLGEN MACROGEN;
DATA _NULL_; CALL SYMPUT('PRTDT',LEFT(PUT(TODAY(),WORDDATE.))); RUN;

* Variable Code: Insert LIBNAME, ID variableIdentifier, and desired titles;
* Note that REL_ID should not include TIMEPOINT;

LIBNAME SAMPLIB 'C:\PROT01\DATA';

%LET REL_ID=SITE PATIENT;

DATA _NULL_; CALL SYMPUT('PRTDT',LEFT(PUT(TODAY(),WORDDATE.))); RUN;
TITLE1 "NEW DRUG PROTOCOL 1";
TITLE2 "DATE PRODUCED: &PRTDT";
TITLE4 'VERIFY CORRECT DATA RECORDS FOR EACH PATIENT AND TIMEPOINT';

%MACRO CHECKDS(NAME);

* Add mock timepoint for data sets without the variable timepoint;

DATA &NAME;
    TIMEPOINT = 1;
    SET SAMPLIB.&NAME;
PROC SORT; BY &REL_ID TIMEPOINT;

* Set variable with name corresponding to data set name for ;
* each observation in the data set. ;

DATA &NAME;
    SET &NAME(KEEP=&REL_ID TIMEPOINT); BY &REL_ID TIMEPOINT;
    IF FIRST.TIMEPOINT THEN &NAME = 0;
    &NAME + 1;
    IF LAST.TIMEPOINT THEN OUTPUT;
PROC SORT; BY &REL_ID TIMEPOINT;

* Merge newly created data set with other data sets previously checked.;

DATA ALL;
    MERGE ALL &NAME;
    BY &REL_ID TIMEPOINT;
PROC SORT; BY &REL_ID TIMEPOINT;
RUN;

%MEND;

* Macro below runs once for each data set, to call macro to output observations;
%MACRO DSNAME();
    %DO I=1 %TO &NUMDS;
        DATA _NULL_;
            SET ALLDS(WHERE=(DSN=&I));
            CALL SYMPUT('NAME',MEMNAME);
        RUN;
        %CHECKDS(NAME=&NAME);
    %end;

```

```

%MEND;

* Create -ALL- data set using the base table to develop a record of;
* ID variables and timepoints.
PROC SORT DATA=SAMPLIB.VISITS (KEEP=&REL_ID TIMEPOINT) OUT=ALL;
    BY &REL_ID TIMEPOINT;

* Create table of ID variable by timepoint number to check for completely missing
timepoints;
PROC FREQ DATA=ALL FORMCHAR(1,2,7)= '  ';
    TABLES &REL_ID*TIMEPOINT / NOPERCENT NOROW NOCOL;

* Run Proc contents and output (no print) results;
PROC CONTENTS DATA=SAMPLIB._ALL_ NOPRINT OUT=MEMLIST;

* Create data set containing one record for each data set in library;
DATA ALLDS;
    SET MEMLIST(KEEP=MEMNAME) END=LASTONE;
    BY MEMNAME;
    IF LAST.MEMNAME THEN DO;
        DSN+1;
        OUTPUT;
    END;
    IF LASTONE THEN CALL SYMPUT("NUMDS",DSN);
RUN;

* Run macro;
%DSNAME;

* Sort resulting data and print;
PROC SORT DATA=ALL; BY TIMEPOINT;
PROC PRINT DATA=ALL; BY TIMEPOINT; PAGEBY TIMEPOINT;

RUN;

```

COPY OF OUTPUT FOR A 4-TIMEPOINT DATABASE:

NEW DRUG PROTOCOL 1
DATE PRODUCED: December 2, 2005

1

VERIFY CORRECT DATA RECORDS FOR EACH ID VARIABLE AND TIMEPOINT

----- Timepoint Number=1 -----

								E	F		M				
		S						X	O	M	I		V		
	I	U				E	E	I	L	E	S	P	T	S	
	N	B	A	B	D	L	L	F	F	H	C		I		
0	V	J	D	A	E	I	I	V	O	I	E	O	E	G	
b	N	N	R	S	M	G	G	A	R	U	D	M	S	N	
s	O	O	S	E	O	1	2	R	M	P	S	M	T	S	
	1	1	9001	1	1	1	1				1	1	1	2	1
	2	1	9002	3	1	1	1				1	2	2	1	1
	3	1	9003	1	1	1	1				1	4	2	1	1
	6	1	9008	4	1	1	1				1	2	1	1	1
	7	1	9009	1	1	1	1				1	1	2	1	1
	8	1	9010	3	1	1	1				1	3	1	1	1
	9	1	9011	1	1	1	1				1	4	1	1	1
10	1	9012	1	1	1	1	1				1	5	2	1	1
11	1	9013		1	2	1	1				1	6	1	1	1
12	1	9014		1	1	1	1				1	1	1	1	1
13	1	9016	6	1	1	1	1				1	3	2	1	1
14	1	9017	1	1	1	1	1				1	2	1	1	1
15	1	9018	1	1	1	1	1				1	3	1	1	1

NEW DRUG PROTOCOL 1
DATE PRODUCED: December 2, 2005

2

VERIFY CORRECT DATA RECORDS FOR EACH ID VARIABLE AND TIMEPOINT

----- Timepoint Number=2 -----

0 b s											E	F				M						
											X	O	M				I					
	I	S								E	I	L	D				S	P	V	I		
	N	U	B	A	B	D	E	L	E	F	T	L	H	M	C	C	E	T	S			
V		J	D	A	E	I	I	V	O	F	O	W	I	E	O	T	I	G				
N		N	R	S	M	G	G	A	R	U	S	D	M	S	N							
O		O	S	E	O	1	2	R	M	P	T	S	M	T	S							
46	1	9001		1				1	1		1						1	1				
47	1	9002		1				1	1		1						1	1				
48	1	9003		1				1	1		1						1	1				
51	1	9008		1				1			1						1	1				
52	1	9009		1				1	1		1						1	1				
53	1	9010		1				1	1		1						1	1				
54	1	9011		1				1	1		1						1	1				
55	1	9012		1				1	1		1						1	1				
56	1	9013						1														
57	1	9014						1														
58	1	9016		1				1	1		1						1	1				
59	1	9017		1				1			1						1	1				
60	1	9018		1				1	1		1						1	1				
61	1	9019		1				1	1		1						1	1				
62	1	9021		1				1	1		1						1	1				
63	1	9022		1				1	1		1						1	1				
64	1	9028		1				1	1		1						1	1				

CONTACT INFORMATION

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

Kim Truett

11585 Jones Bridge Road; Suite 420-115

Alpharetta, GA 30022

770.372.0989

Email: KCTData@comcast.net

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.