MWSUG 2016 - Paper BB12

Time is of Essence: The power of “MISS” is “NOT being missed”!

Gowri Madhavan, Cincinnati Children’s Hospital, Cincinnati, OH
Brittney Delev, Cincinnati Children’s Hospital, Cincinnati, OH

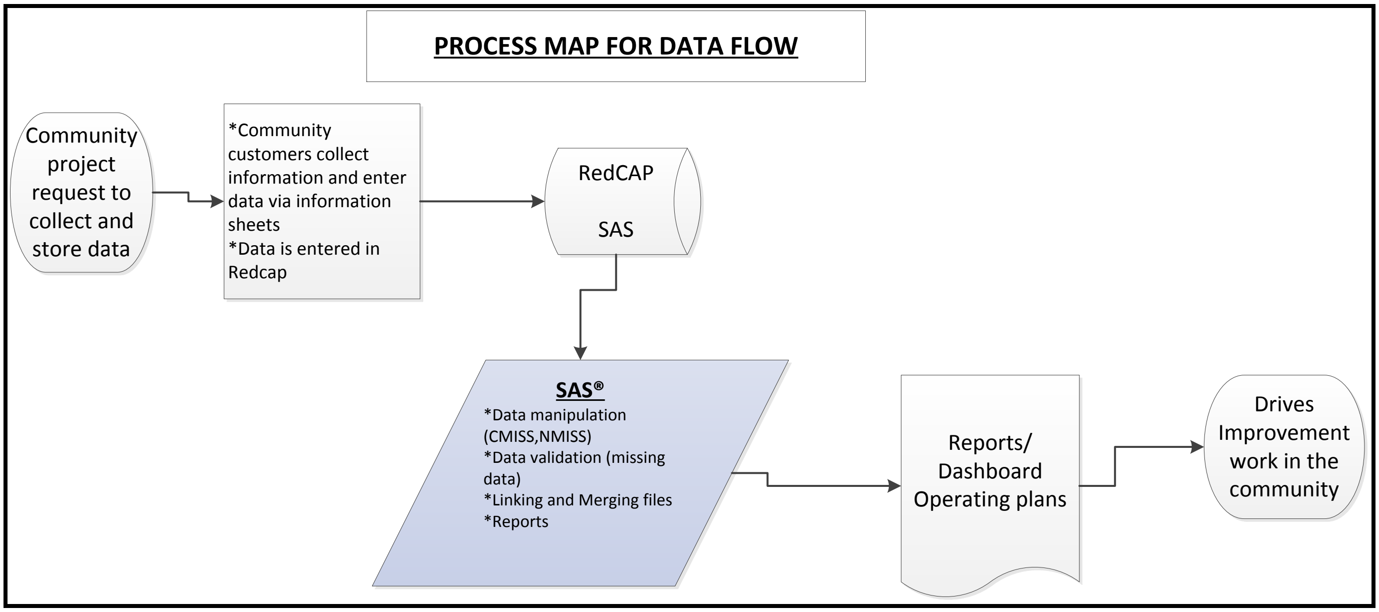
ABSTRACT

Faced with managing data from an external databases such as REDCap (research electronic data capture) for customers; one is often encountered with analyzing data and must conduct quick data manipulation for validating, error checking and running reports. REDCap provides the ability to export data into SAS using macro driven codes into SAS datasets. Time is always of essence and as your clock is ticking away, let the NMISS and CMISS twin codes take charge of your data that will display the number of missing values for each variables and count of missing values for each observation. The NMISS used with a proc means statement will display missing values for each variable. This coupled with the CMISS function can store the number of missing values for both numeric and character variable for each observation. These codes are powerful when used together to generate a SAS dataset using a mathematical operator that will produce the observations with missing values. This output can further be enhanced using traffic color codes to output using proc report with Output Delivery System (ODS) to highlight the observations with missing values vs. validated to your end use customers. A bonus step is to delete observations from a dataset when all or most of the variables have missing data is by using custom macros to manage the variables and invoke the macro variable in a data step. This paper will give you the bandwidth to perform multiple functions.

INTRODUCTION

Projects associated with community health and population health work are unique and different in the aspects of data collection, storing, analyzing and reporting since they necessarily don’t always have analytical models that are associated with them. However, the data that is collected is powerful in informing the customer about capacity building and development. Data capturing for community customers is a challenge that facilitates exploring databases to manage data collection and facilitate analytical work. The figure below outlines the process path for a typical community work.

Figure1


PROCESS MAP FOR DATA FLOW
Community project request to collect and store data
*Community customers collect information and enter data via information sheets
*Data is entered in Redcap
RedCAP
SAS
SAS®
*Data manipulation (CMISS,NMISS)
*Data validation (missing data)
*Linking and Merging files
*Reports
Reports/ Dashboard Operating plans
Drives Improvement work in the community

This paper will help the readers understand the processes associated with importing the data from RedCAP into SAS® and data manipulations steps with CMISS and NMISS to identify missing variables and conducting data and quality check prior to generating reports.

GETTING THE DATA READY FOR SAS

The survey data in entered and saved in a research database (referred to as REDCap). REDCap is a secure web application for building and managing online surveys and databases. A macro driver syntax extracts the data stored in REDCap using an infile statement and implements basic data steps and stores the data in the designated libname

Getting data from REDCap into SAS: A macro program extracts the data as a csv file and outputs as a SAS dataset (partial syntax displayed to conserve sharing all the variables).

Comment: The syntax below is generated by REDCAP in the process of exporting the data into SAS®. This is a macro driven syntax that uses a pathway mapper to convert a csv file into a SAS dataset. The scope of this paper does not include an explanation of this syntax.

```
%macro removeOldFile(bye);
      %if %sysfunc(exist(&bye.)) %then
            %do;

                  proc delete data=&bye.;
                  run;

            %end;
%mend removeOldFile;

libname redcap
'C:\Users\myuserid\Desktop\RedCAP\BlockbyBlock\June1_2016\';

%removeOldFile(redcap.redcap);

data REDCAP;
      %let _EFIERR_ = 0;
      infile 'C:\Users\myuserid\Desktop\RedCAP\BlockbyBlock\June1_2016\
            2559DATA_NOHDRS_2016-06-01_1302.CSV'
            delimiter = ',' MISSOVER DSD lrecl=32767 firstobs=1;
      informat record_id $500.;
      informat import_date_stamp yymmdd10.;
      informat walk_date yymmdd10.;
      informat street_name $500.;
      format record_id $500.;
      format import_date_stamp yymmdd10.;
      format walk_date yymmdd10.;
      format street_name $500.;
      input
            record_id $
            import_date_stamp
            walk_date
```

```
           street_name $;

      if _ERROR_ then
           call symput('_EFIERR_',"1");
run;

data redcap;
      set redcap;
      label record_id='Record ID';
      label import_date_stamp='Import date stamp';
      label walk_date='Walk date';
      label street_name='Street name';
run;

proc format;
      value blockname_ 1='ABC Apartments' 2='Disney Av';
      value address_status_ 1='FY15 Outlier' 2='FY15 Target'
                            3='FY16 Target' 4='FY16 Outlier';
run;

data redcap;
      set redcap;
      format blockname blockname_.;
      format address_status address_status_.;
run;

proc contents data=redcap;

data redcap.REDCAP;
      set REDCAP;
run;

proc format library=work.formats cntlout = redcap.formats;
run;

proc format library=redcap.formats cntlin=redcap.formats;
run;
```

DATA QUALITY

In the context of this community project, it was important to validate the missing data for a subset of variables that feed into a report.   One such example was to determine the missing data for a block of addresses championed by a resident ambassador for intervention that improves the needs of residents. The variables that governed this kind of report are: block name, block captain, address status, status of a given address by fiscal year, components of bundle elements (tailored towards intervention work).

The CMISS function makes it easy to count the number of missing variables across rows for each observation that is defined in the CMISS function. This helps to eliminate multiple lines of coding.

For the purpose of this project; we were interested to determine the number of addresses (referred to as record id) that are missing in a select set of variables that could generate a report for our customers to educate them on the missing data that was not entered in the REDCap database from their survey sheets.

```
data test_missing_rep1;
     set redap_2559;
     howmanymiss=cmiss (of street_name blockname block_captain
     walk_date address_status statustargethome  address_status ); ①
     keep record_id street_name blockname block_captain walk_date
     address_status statustargethome  address_status howmanymiss;
run;
```

① A variable ‘howmanymiss’ is created that uses the function CMISS for the set of set of variables that are required towards a quality check to enable the customers to view and validate their data.

<u>COMMENT</u>
This step forms two dataset(s). One as GREEN for those where there are no rows missing with any of the predefined variable(s). And the second one as RED that will display the rows with one or more missing values (blanks or have a length of zero).

```
data green red; ②
     set test_missing_rep1_revised
          if howmanymiss<1 then output green;
else if howmanymiss>=1 then
     output red;
run;
```

② This step forms two datasets to facilitate a drilldown report for custom reports.   In this example a datasets ‘green’ and ‘red’ are created that are filtered by an if else statement on the variable ‘howmanymiss’.

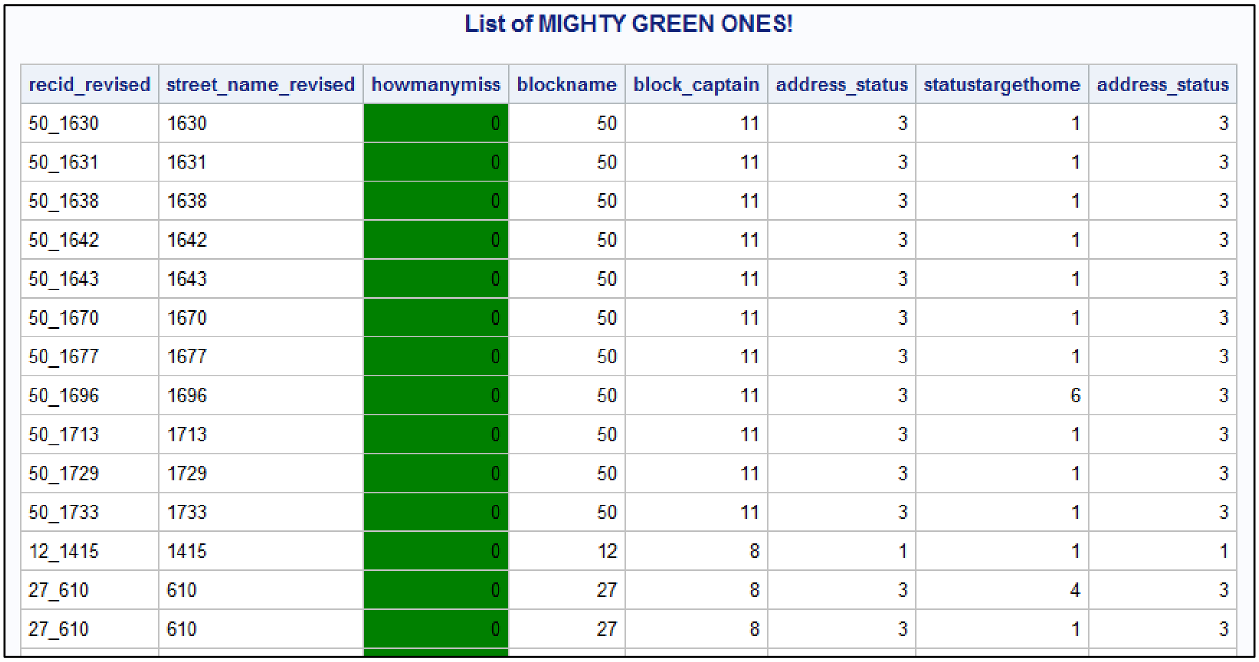
<u>COMMENT</u>
The ODS HTML option opens the HTML destination followed by the PROC REPORT statement. This enables customers to view their report in predefined color coded scheme custom build for each report.

```
title 'List of MIGHTY GREEN ONES!';
ODS HTML BODY='TEMP.HTML';

PROC REPORT DATA=green NOWD;
     COLUMN recid street_name howmanymiss blockname block_captain
     address_status statustargethome address_status;
     DEFINE howmanymiss/DISPLAY;
     DEFINE recid/DISPLAY;
     COMPUTE howmanymiss;

     if howmanymiss<1 then
     call define (_COL_, "STYLE", "STYLE={BACKGROUND=GREEN}");
     ENDCOMP;
RUN;
```

```
ODS HTML CLOSE;
title
```

Partial Listing of output:

Figure2



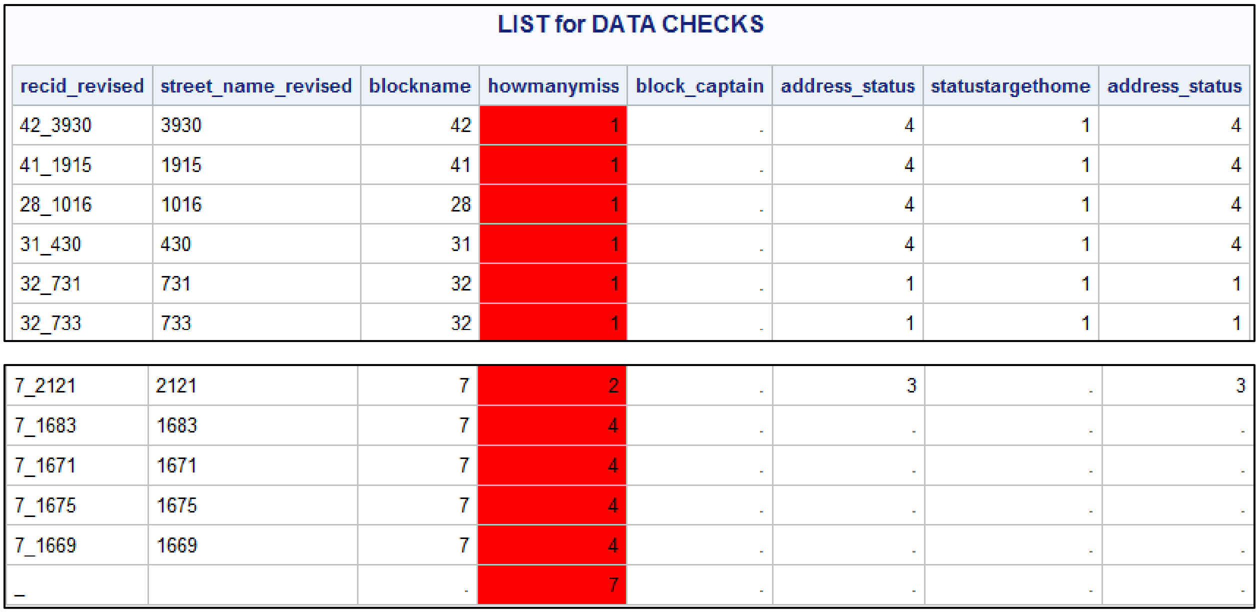
List of MIGHTY GREEN ONES!

recid_revised	street_name_revised	howmanymiss	blockname	block_captain	address_status	statustargethome	address_status
50_1630	1630	0	50	11	3	1	3
50_1631	1631	0	50	11	3	1	3
50_1638	1638	0	50	11	3	1	3
50_1642	1642	0	50	11	3	1	3
50_1643	1643	0	50	11	3	1	3
50_1670	1670	0	50	11	3	1	3
50_1677	1677	0	50	11	3	1	3
50_1696	1696	0	50	11	3	6	3
50_1713	1713	0	50	11	3	1	3
50_1729	1729	0	50	11	3	1	3
50_1733	1733	0	50	11	3	1	3
12_1415	1415	0	12	8	1	1	1
27_610	610	0	27	8	3	4	3
27_610	610	0	27	8	3	1	3

```
proc sort data= red out=red_sorted;
      by howmanymiss;
run;

title 'LIST for DATA CHECKS';
ODS HTML BODY='TEMP.HTML';

PROC REPORT DATA=red_sorted NOWD;
      COLUMN recid street_name howmanymiss blockname block_captain
      address_status statustargethome address_status;
      DEFINE howmanymiss/DISPLAY;
      DEFINE recid/DISPLAY;

      COMPUTE howmanymiss;

           IF howmanymiss>=1 THEN
                CALL DEFINE (_COL_, "STYLE","STYLE={BACKGROUND=RED}");
      ENDCOMP;
RUN;

ODS HTML CLOSE;
TITLE
```

Partial Listing of output:

Figure3



LIST for DATA CHECKS

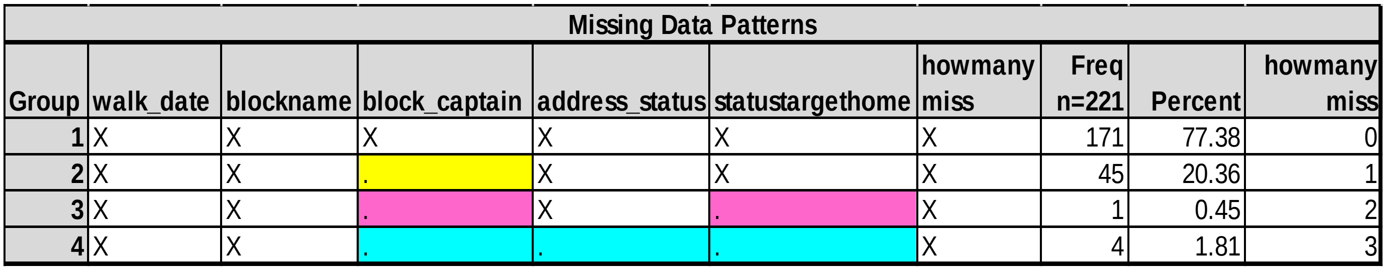
recid_revised	street_name_revised	blockname	howmanymiss	block_captain	address_status	statustargethome	address_status
42_3930	3930	42	1	.	4	1	4
41_1915	1915	41	1	.	4	1	4
28_1016	1016	28	1	.	4	1	4
31_430	430	31	1	.	4	1	4
32_731	731	32	1	.	1	1	1
32_733	733	32	1	.	1	1	1
7_2121	2121	7	2	.	3	.	3
7_1683	1683	7	4	.	.	.	.
7_1671	1671	7	4	.	.	.	.
7_1675	1675	7	4	.	.	.	.
7_1669	1669	7	4	.	.	.	.
_		.	7	.	.	.	.

An additional enhancement using the **PROC MI** displays the pattern and frequency of the missing variables for the entire dataset. For example- on running a quality check on a dataset for a predefined set of variables; **PROC MI** with a VAR statement outputs a table for the variables defined in the dataset

- Of the 221 rows, 171 rows have no blanks (or a length of zero),
- 45 rows with blank data in 1 variable (block_captain),
- 1 row with blank data in 2 variables (block_captain, statustargethome),
- 4 rows with blank data in 3 variables (block_captain, address_status, statustargethome)

```
proc mi data=test_missing_rep1_revised; ③
ods select misspattern;
run;
```

Figure4



Missing Data Patterns

Group	walk_date	blockname	block_captain	address_status	statustargethome	howmany miss	Freq n=221	Percent	howmany miss
1	X	X	X	X	X	X	171	77.38	0
2	X	X	.	X	X	X	45	20.36	1
3	X	X	.	X	.	X	1	0.45	2
4	X	X	.	.	.	X	4	1.81	3

③The **PROC MI** statement with the **ODS** option called **misspattern** outputs a table of the missing data pattern that are present in the sample data file. A data file with select numeric variables are formed here to output the results from **PROC MI** statement. For the purpose of this report, the output associated with the **PROC MI** associated with the group means are not used for paper.

DATA QUALITY FOR ‘INTERVENTION’ TOUCHPOINTS (NMISS FUNCTION)

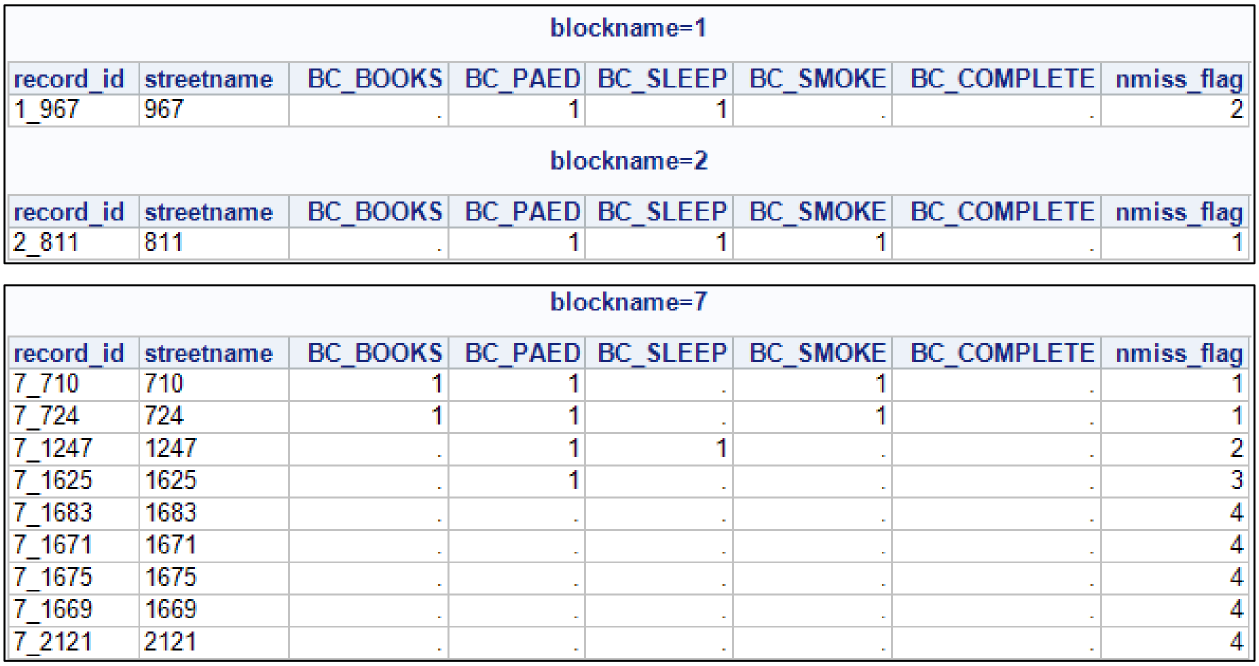
The function **NMISS** is used extensively for validating multiple numeric values and determine the missing ones. For the same dataset that has several touchpoints of completion status for each ‘intervention’ defined as bundle status, the **NMISS** function is very powerful to validate the number of touch points that are missing for a given project.

```
data test_missing_rep1_revised;
     set test_missing_rep1;

     ****COMMENT: nmiss is for validating numeric variables;
     nmiss_flag=nmiss(of bc_books,bc_paed,bc_sleep,bc_smoke);④

     keep record_id street_name blockname block_captain walk_date
address_status statustargethome howmanymiss nmiss_flag bc_books
bc_complete bc_paed bc_sleep bc_smoke;

run;
```

④ The **NMISS** function calculates the missing data values and assigns a rank for the number of missing values for each row of data.

Partial output for missing data for intervention touchpoints by each block. The variables labeled as `BC_BOOKS` etc. are discrete variables for the dataset that refer to each intervention/ touchpoint. In the display provided below `nmiss_flag` indicates the number of missing intervention touchpoints for each row.

Figure5



blockname=1

record_id	streetname	BC_BOOKS	BC_PAED	BC_SLEEP	BC_SMOKE	BC_COMPLETE	nmiss_flag
1_967	967	.	1	1	.	.	2

blockname=2

record_id	streetname	BC_BOOKS	BC_PAED	BC_SLEEP	BC_SMOKE	BC_COMPLETE	nmiss_flag
2_811	811	.	1	1	1	.	1

blockname=7

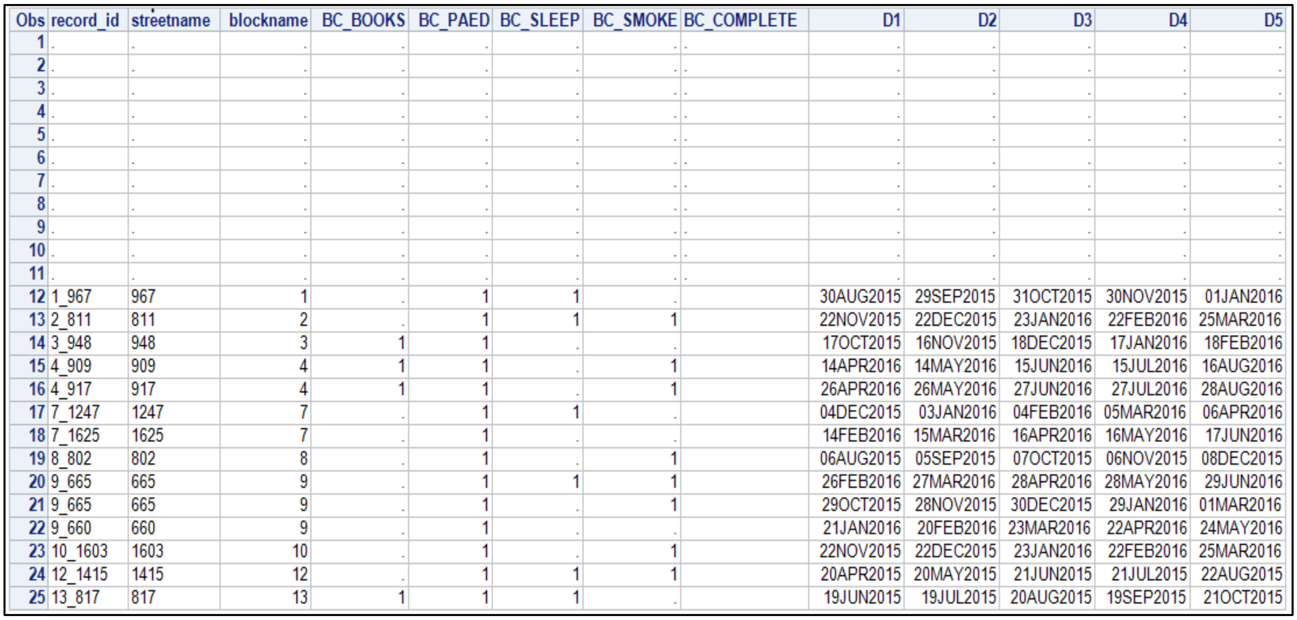
record_id	streetname	BC_BOOKS	BC_PAED	BC_SLEEP	BC_SMOKE	BC_COMPLETE	nmiss_flag
7_710	710	1	1	.	1	.	1
7_724	724	1	1	.	1	.	1
7_1247	1247	.	1	1	.	.	2
7_1625	1625	.	1	.	.	.	3
7_1683	1683	.	.	.	.	.	4
7_1671	1671	.	.	.	.	.	4
7_1675	1675	.	.	.	.	.	4
7_1669	1669	.	.	.	.	.	4
7_2121	2121	.	.	.	.	.	4

DATA MANAGEMENT FOR MISSING DATA

It is not uncommon to observe multiple rows of missing data in a community level project. It is obviously important to delete these rows with missing data prior to any analytical step.

Partial listing of the dataset **`test_missing`** displays multiple rows of null data due to data entry error(s). In the display below the first eleven rows do not have any data and therefore can be deleted. A macro variable can facilitate this step very efficiently.

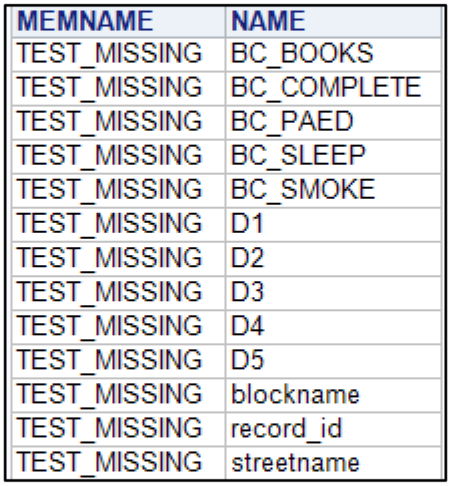
Figure6



Obs	record_id	streetname	blockname	BC_BOOKS	BC_PAED	BC_SLEEP	BC_SMOKE	BC_COMPLETE	D1	D2	D3	D4	D5
1	.	.	.	.	.	.	.	.	.	.	.	.	.
2	.	.	.	.	.	.	.	.	.	.	.	.	.
3	.	.	.	.	.	.	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.	.	.	.	.	.	.
6	.	.	.	.	.	.	.	.	.	.	.	.	.
7	.	.	.	.	.	.	.	.	.	.	.	.	.
8	.	.	.	.	.	.	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.	.	.	.	.	.	.
10	.	.	.	.	.	.	.	.	.	.	.	.	.
11	.	.	.	.	.	.	.	.	.	.	.	.	.
12	1_967	967	1	.	1	1	.		30AUG2015	29SEP2015	31OCT2015	30NOV2015	01JAN2016
13	2_811	811	2	.	1	1	1		22NOV2015	22DEC2015	23JAN2016	22FEB2016	25MAR2016
14	3_948	948	3	1	1	.	.		17OCT2015	16NOV2015	18DEC2015	17JAN2016	18FEB2016
15	4_909	909	4	1	1	.	1		14APR2016	14MAY2016	15JUN2016	15JUL2016	16AUG2016
16	4_917	917	4	1	1	.	1		26APR2016	26MAY2016	27JUN2016	27JUL2016	28AUG2016
17	7_1247	1247	7	.	1	1	.		04DEC2015	03JAN2016	04FEB2016	05MAR2016	06APR2016
18	7_1625	1625	7	.	1	.	.		14FEB2016	15MAR2016	16APR2016	16MAY2016	17JUN2016
19	8_802	802	8	.	1	.	1		06AUG2015	05SEP2015	07OCT2015	06NOV2015	08DEC2015
20	9_665	665	9	.	1	1	1		26FEB2016	27MAR2016	28APR2016	28MAY2016	29JUN2016
21	9_665	665	9	.	1	.	1		29OCT2015	28NOV2015	30DEC2015	29JAN2016	01MAR2016
22	9_660	660	9	.	1	.	.		21JAN2016	20FEB2016	23MAR2016	22APR2016	24MAY2016
23	10_1603	1603	10	.	1	.	1		22NOV2015	22DEC2015	23JAN2016	22FEB2016	25MAR2016
24	12_1415	1415	12	.	1	1	1		20APR2015	20MAY2015	21JUN2015	21JUL2015	22AUG2015
25	13_817	817	13	1	1	1	.		19JUN2015	19JUL2015	20AUG2015	19SEP2015	21OCT2015

```
proc contents data=test_missing out=contents_test (keep=memname name);
run;
```

Figure7



MEMNAME	NAME
TEST_MISSING	BC_BOOKS
TEST_MISSING	BC_COMPLETE
TEST_MISSING	BC_PAED
TEST_MISSING	BC_SLEEP
TEST_MISSING	BC_SMOKE
TEST_MISSING	D1
TEST_MISSING	D2
TEST_MISSING	D3
TEST_MISSING	D4
TEST_MISSING	D5
TEST_MISSING	blockname
TEST_MISSING	record_id
TEST_MISSING	streetname

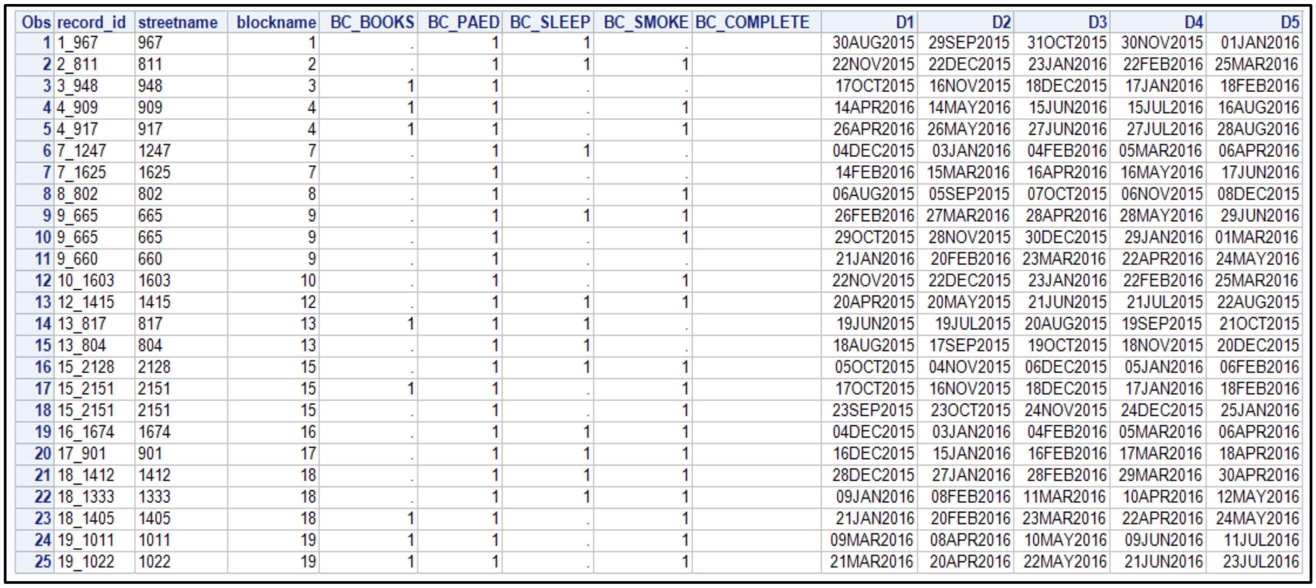
```
proc sql;
     select distinct name into: names separated by ','
          from contents_test  where memname='TEST_MISSING';⑤
QUIT;
```

⑤ A macro variable 'names' is created that lists the variables names in the dataset with a null/missing value.

```
data remove_missing3;
     set nmiss_sorted_missing3;

     if n(&names) lt 1 then
          delete;
run;
```

Partial list of the dataset that has been removed for rows with missing data.

Figure8

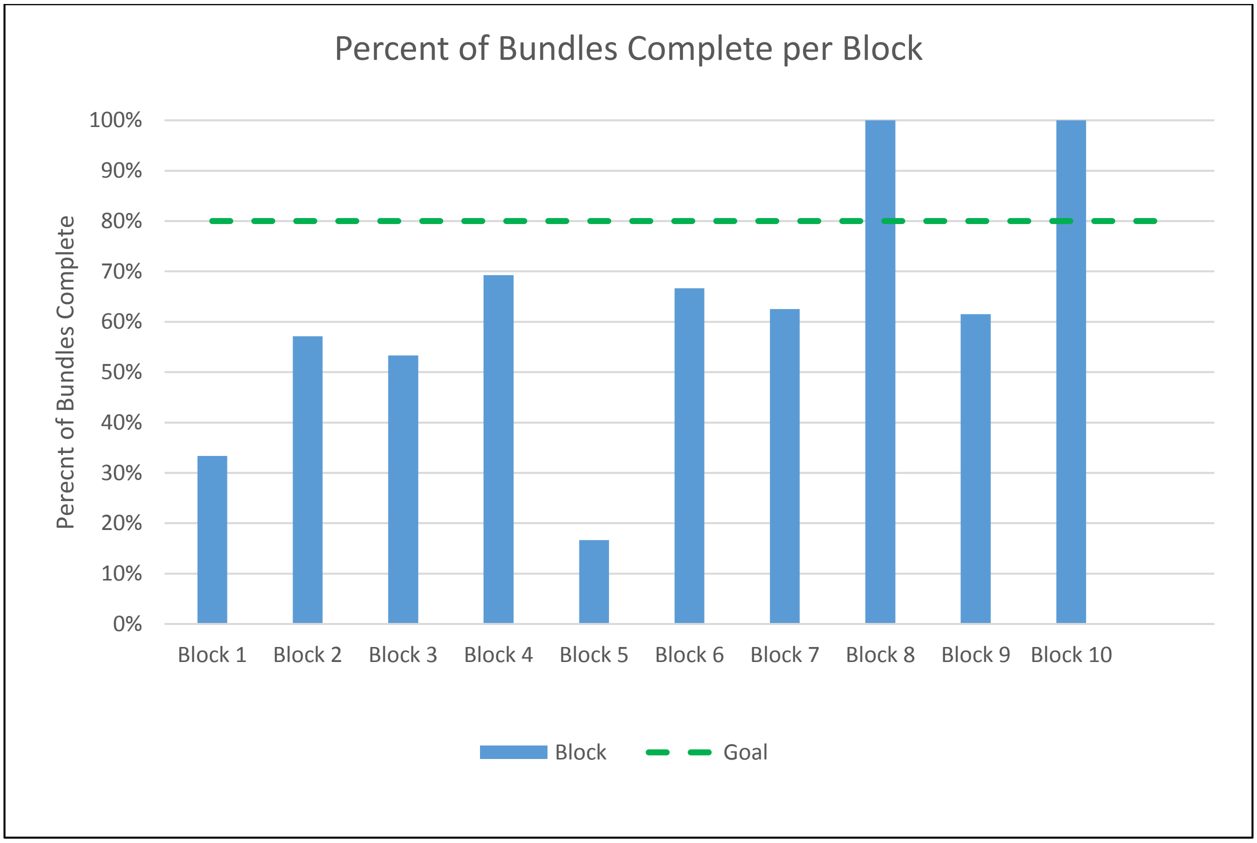


Obs	record_id	streetname	blockname	BC_BOOKS	BC_PAED	BC_SLEEP	BC_SMOKE	BC_COMPLETE	D1	D2	D3	D4	D5
1	1_967	967	1	.	1	1	.		30AUG2015	29SEP2015	31OCT2015	30NOV2015	01JAN2016
2	2_811	811	2	.	1	1	1		22NOV2015	22DEC2015	23JAN2016	22FEB2016	25MAR2016
3	3_948	948	3	1	1	.	.		17OCT2015	16NOV2015	18DEC2015	17JAN2016	18FEB2016
4	4_909	909	4	1	1	.	1		14APR2016	14MAY2016	15JUN2016	15JUL2016	16AUG2016
5	4_917	917	4	1	1	.	1		26APR2016	26MAY2016	27JUN2016	27JUL2016	28AUG2016
6	7_1247	1247	7	.	1	1	.		04DEC2015	03JAN2016	04FEB2016	05MAR2016	06APR2016
7	7_1625	1625	7	.	1	.	.		14FEB2016	15MAR2016	16APR2016	16MAY2016	17JUN2016
8	8_802	802	8	.	1	.	1		06AUG2015	05SEP2015	07OCT2015	06NOV2015	08DEC2015
9	9_665	665	9	.	1	1	1		26FEB2016	27MAR2016	28APR2016	28MAY2016	29JUN2016
10	9_665	665	9	.	1	.	1		29OCT2015	28NOV2015	30DEC2015	29JAN2016	01MAR2016
11	9_660	660	9	.	1	.	.		21JAN2016	20FEB2016	23MAR2016	22APR2016	24MAY2016
12	10_1603	1603	10	.	1	.	1		22NOV2015	22DEC2015	23JAN2016	22FEB2016	25MAR2016
13	12_1415	1415	12	.	1	1	1		20APR2015	20MAY2015	21JUN2015	21JUL2015	22AUG2015
14	13_817	817	13	1	1	1	.		19JUN2015	19JUL2015	20AUG2015	19SEP2015	21OCT2015
15	13_804	804	13	.	1	1	.		18AUG2015	17SEP2015	19OCT2015	18NOV2015	20DEC2015
16	15_2128	2128	15	.	1	1	1		05OCT2015	04NOV2015	06DEC2015	05JAN2016	06FEB2016
17	15_2151	2151	15	1	1	.	1		17OCT2015	16NOV2015	18DEC2015	17JAN2016	18FEB2016
18	15_2151	2151	15	.	1	.	1		23SEP2015	23OCT2015	24NOV2015	24DEC2015	25JAN2016
19	16_1674	1674	16	.	1	1	1		04DEC2015	03JAN2016	04FEB2016	05MAR2016	06APR2016
20	17_901	901	17	.	1	1	1		16DEC2015	15JAN2016	16FEB2016	17MAR2016	18APR2016
21	18_1412	1412	18	.	1	1	1		28DEC2015	27JAN2016	28FEB2016	29MAR2016	30APR2016
22	18_1333	1333	18	.	1	1	1		09JAN2016	08FEB2016	11MAR2016	10APR2016	12MAY2016
23	18_1405	1405	18	1	1	.	1		21JAN2016	20FEB2016	23MAR2016	22APR2016	24MAY2016
24	19_1011	1011	19	1	1	.	1		09MAR2016	08APR2016	10MAY2016	09JUN2016	11JUL2016
25	19_1022	1022	19	1	1	.	1		21MAR2016	20APR2016	22MAY2016	21JUN2016	23JUL2016

REPORTING OUT

To wrap up data manipulation steps using CMISS, NMISS and macro variables, a dataset is formed that is subsequently used to generate a report. The report gives a simple representation of the percent of homes that have received the intervention elements with a predicated GOAL (green dashed line).

Figure9



CONCLUSION

Healthcare data is in a continuous process of evolving from paper based to electronic systems and fine-tuned to agile dashboards. In such instances, it is extremely valuable using SAS® to manage data efficiently and one such process is using a series of simple SAS functions which has been described in this paper. There are numerous parallel methods to achieve similar results to manipulate data. The scope of this paper was focused on using CMISS, NMISS and a macro variable to manage data that is entered by external customer(s) with beginner to intermediate knowledge in data entry. Therefore it was necessary for the core data team to conduct data validation and process reports.

ACKNOWLEDGMENTS

We would like to thank the following for their invaluable contributions to this project:

John Besl, M.A., Demographer, Cincinnati Children’s Hospital
Erika McDaniel, Student II, Cincinnati Children’s Hospital
Kenya Simmons, BSEE, MBA, Consultant: Community- Engagement, Cincinnati Children’s Hospital
Lisa Watkins, MPA, Block by Block Coordinator, Santa Maria Community Services
(Members of the Healthy Homes: Block by Block Team)

CONTACT INFORMATION

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

Gowri Madhavan
Cincinnati Children’s Hospital
Senior Analyst, Q&T Process Improvement
James M Anderson Center for Health Systems Excellence
3333 Burnet Avenue, MLC 7014, Cincinnati, OH 45229
Gowri.Madhavan@cchmc.org

Brittney Delev
Cincinnati Children’s Hospital
Student II, Q&T Process Improvement
James M Anderson Center for Health Systems Excellence
3333 Burnet Avenue, MLC 7014, Cincinnati, OH 45229
Brittney.Delev@cchmc.org