Paper 99-26

Quick and Dirty Data Laundering: A Scalable Solution for Range Checking Data

Michael Yee, M.P.H., The Lewin Group, San Francisco, CA

ABSTRACT

Data cleaning is a crucial component of statistical analysis. Collected data is rarely provided ready to analyze, and therefore must be range checked to ensure its integrity. Range checks are a common means of determining whether a given value falls within a defined range. A SAS® software technique for checking data was developed at The Lewin Group that is not only universally applicable to a wide variety of data sets but also capable of adapting dynamically to changing data dimensions. The technique would need to be able to accommodate the addition of new observations (vertical scaling) as well as the adoption of new range checks (horizontal scaling). BASE SAS and SAS macros are used to systematically create a range checking macro program that will output an error level data set. This technique was adapted to the Windows OS and can be easily implemented by the intermediate SAS programmer with a working knowledge of macros. The flexibility of this technique has greatly reduced both programming and processing time required by our projects.

INTRODUCTION

To a health care consulting organization like The Lewin Group, the integrity of the data drives the quality of the analysis. The SAS programmer is responsible for ensuring data integrity by performing range checks on the data. Out of range data must be resolved by querying the data sources. A SAS application was needed to perform these tasks for multiple projects. This tool would need to be scalable in order to accommodate any number of range checks.

SAS software was used to develop such a system here at The Lewin Group. A reduced-scale version of the program is presented which can be easily scaled upwards if more range checks or observation are added.

STEP 1: CREATING THE RANGE CHECK TABLE

The first step would be to create the range check table that describes each range check to be conducted. This table would be the only part of the process that is specific to a given analysis data set. The table would contain the following variables:

ERRORID – An identifier used to code a particular error type.
ERRORVAR – The variable upon which the range check is performed
ERRORSAS – The SAS code used to define when a variable is outside of acceptable range.

Microsoft Access and Microsoft Excel are most often used to create the range check table. ODBC drivers or PROC IMPORT are used to import the table as a temporary SAS data set. In this example SAS software and the DATA step will produce the range check table. Adding new range checks to the lookup table is as simple as assigning a new ERRORID, new ERRORVAR, and identifying the proper code for ERRORSAS. ERRORSAS **must indicate the out of range condition** (not the in range condition) in the form of SAS programming code. The ability to add new range checks simply by adding new lines to the range check table contributes to the ease of horizontal scaling of this process.

```
data rangetbl;
input   @1 errorid $8.
        @9 errorvar $8.
        @17 errorsas $40.;
datalines;
1.1     race      race not in (1,2,3)
1.2     sex       sex not in (1,2)
2.1     labval1   not(50<=labval1<=100)
2.2     labval2   labval2 not in (.,0)
2.3     labval3   labval3^=.
;
run;
```

STEP 2: ENCODING THE MACRO VARIABLES

Each variable in the range check table is systematically assigned to a macro variable. For example the values in the data set RANGETBL will be assigned the following macro values:

```
        &id1    resolves to 1.1
        &var1   resolves to race
        &code1  resolves to race not in (1,2,3)
        &id2    resolves to 1.2
        &var2   resolves to sex
        &code2  resolves to sex not in (1,2)
```

and so forth, until the data lines are exhausted. CALL SYMPUT statements within the DATA _NULL_ will systematically create the numbered macro variables.

```
data _null_;
set rangetbl end=last;
retain count;
if _n_=1 then count=0;
  count=count+1;
  call symput('id'||left(trim(put(count,5.))),
    trim(errorid));
  call symput('var'||left(trim(put(count,5.))),
    trim(errorvar));
  call symput('code'||left(trim(put(count,5.))),
    trim(errorsas));
if last then call symput
    ('checkct',left(trim(put(count,8.))));
run;
```

A single macro variable CHECKCT counts the number of range check macro sets created. In the above example:

```
        &checkct resolves to 5
```

Since macro variables are stored in memory and are resolved prior to reading in data via the program data vector (PDV), this technique can reduce the amount of processing time required to evaluate the range checks.

STEP 3: PREPARING THE DATA SET

This step creates the data set to be range checked. This data set is typically sent by a client and needs to be converted to a SAS data set prior to range checking. In this example, the DATA step creates a temporary data set.

```
data dataset;
input   @1  patid   3.
        @4  race    3.
        @7  sex     3.
        @10 labval1 3.
        @13 labval2 3.
        @16 labval3 3.;
datalines;
1  1  1  75 2  12
2  2  2  99 7  17
3  9  1  55 2  15
4  1  0  80 3  16
5  1  1  25 5  10
6  2  2  75 .  12
7  3  2  0  0  0
;
run;

proc sort data=dataset;
  by patid;
run;
```

Increasing or decreasing the size of the data set does not change the data checking process. This is the vertical scaling advantage of this technique.

STEP 4: OUTPUT THE ERROR-LEVEL DATA SET

The following macro program will output a new observation into an error-level data set whenever the range check condition is not met. The macro program checks the variable indicated in ERRORVAR against the condition previously indicated in ERRORSAS. If an out of range condition is found, then the macro program outputs one new observation per out of range variable per study subject.

```
%macro checkvar(id, var, check );
if &check then do;
   errorid ="&id.";
   errorvar="&var.";
   errorval=&var;
   output;
end;
%mend checkvar;
```

The error level data set will contain ERRORID, ERRORVAR, ERRORVAL and patient identifier PATID. Patient identifiers from DATASET are automatically passed through and together the PATID and the ERRORID create a unique identifier for the ERROR_DS data set.

```
data error_ds (keep=patid errorid errorvar
errorval);
  set dataset;
 length errorid errorvar $8 errorval 8;
  %macro runcheck;
    %do j=1 %to &checkct;
      %checkvar(%str(&&id&j),
                %str(&&var&j),
                %str(&&code&j));
    %end;
  %mend runcheck;
  %runcheck;
run;

proc print data=error_ds;
  title 'List of variable to be queried';
run;
```

The macro program RUNCHECK will iterate through the macro program CHECKVAR as many times as the macro variable CHECKCT indicates. The values for ERRORID, ERRORVAR, and ERRORSAS will be passed through the macro and each observation will be subject to all of the range checks in succession. The %STR function is required in case any of the fields contain embedded commas, equal signs or other dedicated SAS punctuation. Range errors will be output to the data set ERROR_DS.

STEP 5: ANALYZING THE ERROR-LEVEL DATA

The error level data set now contains four variables. The PATID is the unique identifier for the patient. The ERRORID is the identifier that indicates the type of error. ERRORVAR is the variable that was out of range and ERRORVAL is the value that was out of range.

List of variables to be queried

Obs	PATID	ERRORID	ERRORVAR	ERRORVAL
1	1	2.2	labval2	2
2	1	2.3	labval3	12
3	2	2.2	labval2	7
4	2	2.3	labval3	17
5	3	1.1	race	9
6	3	2.2	labval2	2
7	3	2.3	labval3	15
8	4	1.2	sex	0
9	4	2.2	labval2	3
10	4	2.3	labval3	16
11	5	2.1	labval1	25
12	5	2.2	labval2	5
13	5	2.3	labval3	10
14	6	2.3	labval3	12
15	7	2.1	labval1	0
16	7	2.3	labval3	0

CONCLUSION

Base SAS and the SAS macro facility provide a useful tool for rapidly range checking large amounts of data. This process easily accommodates additional range checks or additional observations with minimal new programming. Range checking data is an important step of data analysis because it can reveal inconsistencies in the data that should be investigated and resolved before statistical testing occurs.

CONTACT INFORMATION

Thank you for taking the time to read this paper and/or attend the presentation. Your comments and questions are valued and encouraged. Contact the author at:

Michael Yee, M.P.H.
The Lewin Group
490 2nd Street, Suite 201
San Francisco, CA 94107
Office: 415-495-8966
michael.yee@lewin.com

Remember to always improve, innovate, and inspire!