

## Data Validation Using A SAS Format

John R. Gerlach, IMS America; Plymouth Meeting, PA  
Naoko S. Stearns, Research Triangle Institute; RTP, NC

### Abstract

Proper data analysis requires well-defined data that have been validated. Typically, the transition from raw data to a viable SAS® data set requires more effort than just writing an INPUT statement. In fact, the quest for validated data often becomes a project in itself consuming tremendous resources. Considering that many validation checks can be written as Boolean expressions, it's possible that the checks can be stored in a user-defined format. The SAS solution described here involves a macro and a user-defined format that performs validation checks on a data set, then produces a report. Consequently, the validation checks are maintained by simply modifying the format and the validation process is performed, as needed, by invoking the macro. This paper explains an easy and self-documenting approach to the validation process.

### The Real World

The real-world of data analysis tends to be rampant with convoluted specifications and unclear data. Even worse, the proliferation of SAS programs often emulates an ad hoc analysis, rather than a well organized and documented system of SAS programs ensuring correct results that can be replicated. Consequently, the project endures many hardships, complete with a nasty deadline, while attempting to associate ambiguity (specifications) with inconsistency (data).

Rather than address the human condition that might improve the real world of data analysis, this paper pursues the easier task of ensuring good consistent data by proposing a SAS solution.

### The SAS Solution

Consider a clinical trial study consisting of several protocols representing hundreds of patients and lots of data, including: demographics, lab tests, vital signs, and dosing, each represented by its own SAS data set. It would be nice to have a data validation tool that can be used repeatedly, thereby facilitating data management and ensuring integrity of the data. Also, it would be even better if the validation checks were well organized rather than buried in a collection of programs.

Assuming that the validation checks can be written as Boolean expressions, it is possible to store each check as the label portion of a numeric SAS format such that there would be *n*-levels in the Value statement

of the FORMAT procedure. Then, using the format, along with the Macro Language and some Base SAS, the validation process becomes almost trivial and very efficient. In fact, for the end-user, the validation becomes a matter of maintaining a format library and invoking a SAS macro with several parameters, as needed.

### The QC Format Library

Below is a syntax diagram that illustrates how to create the validation checks using Proc Format.

```
proc format;

    value Name of format containing QC checks
        1 = "Boolean expression"
        2 = "Boolean expression"
        :
        n = "Boolean expression";

run;
```

Below is an example of creating several formats that contain validation checks. The numeric formats *qcpat*, *qclab*, and *qcdose* contain a collection of checks specific to the several data sets PATIENTS, LABS, and DOSE, respectively. Notice that each Boolean expression must be reasonable and syntactically appropriate with respect to the variables in the data set. Otherwise, obviously, you would get syntax errors when invoking the %qcprep macro that performs the validation task.

```
proc format;

    value qcpat
        1 = "gender not in('M','F')"
        2 = "age gt 24 | age eq ."
        3 = "1 le wt le 35"
        4 = "ht eq ."
        5 = "gender eq 'M'";

    value qclab
        1 = "syst gt 200 and dias gt 150"
        2 = "syst eq . or dias eq .";

    value qcdose
        1 = "date eq ."
        2 = "dose not in (3,6,12,25)";

run;
```

With this arrangement, you can add or delete validation checks simply by modifying the format. Also, notice that the format library containing the validation checks and the data sets can reside in separate data libraries. Moreover, each format is an independent collection of Boolean expressions that are pertinent to a specific SAS data set used for the sole purpose of validating it.

## The %qcrep Macro

The %qcrep macro performs the validation process. It contains five parameters, two of which are positional parameters that require user input while the remaining are keyword parameters that have default values and are not required, depending on the situation. The parameters are listed below.

- qcdset** Specify the name of the temporary or permanent SAS data set being validated.
- qcfmt** Specify the name of the user-defined numeric format that contains the validation checks.
- qclib** (optional) Specify the name of the SAS data library where the format containing the validation checks resides. The default is the WORK library. Also, the format need not reside in the same library as the data set being validated.
- qcvar** (optional) List the variables in the data set that are pertinent to the validation process. The default is *all* variables. This option is useful when you wish to limit the variables being processed, excluding those variables that are typically irrelevant to the validation process (e.g., patient's name). This option affects the detailed listing, as well.
- qcdet** (optional) Request a detailed listing of errors, Yes or No. The default is Yes; otherwise, you will receive only a report indicating the frequency of each error.

The following explains how the %qcrep macro performs the validation process:

The OPTIONS statement ensures that the output is paginated and dated.

The FORMAT procedure selects the format containing the validation checks and creates an output data set containing only the variable *start*, which is numeric ranging from 1 to *n*, denoting the number of checks.

A Data \_NULL\_ step proceeds to create the macro variables: *&qchk1.*, *&qchk2.*, ..., *&qchkn.*, representing the validation checks and the macro variable *&qcnchk.* denoting the number of validation checks.

The next Data step analyzes the data set as specified by the *&qcdset* parameter, keeping only those variables of interest depending on the *&qcvar* parameter. The %DO loop generates sequential IF statements, each with a DO / END block, for every validation check. The macro variable *&&qchk&i.* resolves to an IF expression, that is, the respective validation check. Consequently, the Data step will generate an observation whenever the expression is true (e.g., gender not in ('M', 'F')). The

result is the data set *qcrep* which contains all the variables of interest originating from the user-specified data set being analyzed and the variable *qcchk* which indicates the error in the data set.

The FREQ procedure creates an output data set containing the variables *qcchk* and *count* that represent the frequency of occurrence of each check. FREQ does not produce standard output because of its 16-byte limitation (prior to SAS Release 6.12). Subsequently, the PRINT procedure produces the first report using the very same format which contains the validation checks.

Assuming that the user wants a detailed report of the validation checks, the SORT and PRINT procedures generate the second report. Otherwise, the user obtains only a frequency report.

### Caveat

In the event that the data set being validated contains no errors, the %qcrep macro generates no reports. Instead, the SAS log indicates that the reporting procedures inside the macro processed data sets having 0 observations. No run-time error occurs since the data sets exist, albeit empty; hence, the option NODSNFERR is not required. By design, the macro does not even check for this situation.

### Invoking the %qcrep Macro

The following examples illustrate proper use of the %qcrep macro.

```
%qcrep(prot021.patient, qcpat)
```

Validates the data set *patient* using the *qcpat* format, which resides in the WORK library because of the default value of the *qclib* parameter. Also, the macro uses all the variables in the patient data set and generates a detailed report, as well.

```
%qcrep(prot054.lab, qclab, qclib=prot021,
      qcvar=pid vdate syst dias, qcdet=N)
```

Validates the data set *lab* using the *qclab* format, which resides in a different library. Also, the macro uses only several variables from the data set and does not generate a detailed report.

```
%qcrep(prot054.dose, qcdose, qclib=prot021,
      qcvar=pid date dose, qcdet=Y)
```

Validates the data set *dose* using the *qcdose* format, which resides in a different library. Also, the macro uses only several variables from the data set and generates a detailed report.

### Using an Existing Format

Suppose you wish to validate a data set that contains dosing information in a clinical trial study and you learn of an existing format that contains most of the forty plus validation checks of interest. However, you need to

exclude some of the checks and add new ones. And, of course, you do not have access to the original program that created the original format.

Using the **FORMAT** procedure and a subsequent **Data** step, you can clone an existing format to meet your specific needs. Consider the following code that clones an existing format from one protocol for the purpose of validating a dose file belonging to another protocol.

```
proc format library=prot08
  cntlout=qcdose(keep=fmtname type start
    label) fmtlib;
  select qcdose;
run;

data qcdose;
  set qcdose(where=(left(start)
    not in('1', '3', '18', '23')));
  start = put(_n_,8.);
run;

proc format library=prot15 cntlin=qcdose;
run;
```

The **CNTLOUT** option above creates a control data set that contains the information you need to revise the format. Also, the **FMTLIB** option produces a report so that you can determine which validation checks to keep.

Upon inspection of the format, you decide to exclude checks 1,3,18, and 23. The **Data** step modifies the control data set accordingly. Notice the **LEFT** and **PUT** functions. The variable *start* in the control data set, which denotes the *k*th validation check, contains right justified character data having a length of 16 bytes. Thus, the **LEFT** function properly subsets the data set, that is, it deletes those unwanted validation checks. Then, the **PUT** function assigns new ordered values (1,2, ..., *n*) to the variable *start* using SAS automatic variable **\_N\_**. Finally, the **FORMAT** procedures uses the control data set and creates a new format that resides in a different data library, as shown below. Adding validation checks to an existing format requires more effort.

Alternatively, the validation checks could be stored in a data set, even maintained by a front-end. Then, using a **Data** step and the **CNTLIN** option of the **FORMAT** procedure, the format can be created during the validation process.

### Conclusion

Whether your data represent a clinical trial intending to prove the efficacy of a drug or an analysis attempting to discern trends in the stock market, proper data analysis requires validated data. The initial goal of attaining good data must be done efficiently and effectively; otherwise, the expense incurred could, in fact, jeopardize the project or the integrity of the analysis.

The **%qcrep** macro facilitates the validation process and helps to ensure the integrity of the data. The macro is easy to use and produces two very useful reports. Also, the user-defined format that contains the

actual validation checks is *de facto* documentation of the validation process.

### The Code

```
%macro qcrep(qcdset,qcfmt,qclib=work,
  qcvar=_all_,qcdet=Y);

  options number pageno=1 date;

  proc format library=&qclib.
    cntlout=qcchk(keep=start);
    select &qcfmt.;
  run;

  data _null_;
    set qcchks nobs=n;
    if _n_ eq 1
      then call symput('qcchks',
        left(put(n,2.)));
    call symput('qcchk' ||
      trim(left(put(_n_,2.))),
      left(put(_n_,&qcfmt.)));
  run;

  data qcrep;
    set &dsn.(keep=&qcvar.);
    %do i = 1 %to &qcchks.;
      if &&qcchk&i...
        then do; qcchk = &i.; output; end;
    %end;
  run;

  proc freq data=qcrep noprint;
    tables qcchk / out=qcchks;
  run;

  title1 "QC Analysis of %upcase(&qcdset.)";

  proc print data=qcchks noobs label;
    var qcchk count;
    format qcchk &qcfmt.. count comma8.;
    label qcchk = 'QC Check'
      count = '#';
    title2 "Frequency of QC Checks";
  run;

  %if %upcase(&qcdet.) eq Y
    %then %do;
      proc sort data=qcrep;
        by qcchk;
      run;
      proc print data=qcrep label noobs;
        by qcchk;
        format qcchk &qcfmt..;
        title2 "Detailed Listing";
      run;
    %end;

%mend qcrep;
```

### Author Information

John R. Gerlach  
IMS America  
600 West Germantown Pike  
Plymouth Meeting, PA 19462-1048  
610.832.5493

Naoko S. Stearns  
Research Triangle Institute  
Research Triangle Park, NC 27709-2194  
919.541.7369

SAS is a registered trademark of SAS Institute.