Paper 032-2008

Clinical-Data Acceptance Testing Procedure

Sunil Gupta, Quintiles, Thousand Oaks, CA

ABSTRACT

In the pharmaceutical industry, there is a regulatory responsibility, 21 CFR Part 11, to analyze only the clinical data that has passed data acceptance testing or is considered 'clean data' after a database lock. Clinical data acceptance testing procedure involves confirming the validity of critical data variables. These critical data variables might need to be non-missing, consist only of valid values, be within a range, or be consistent with other variables. If incorrect clinical data is analyzed, then invalid study conclusions can be drawn about the drug's safety and efficacy.

In 2001, the Data Warehousing Institute conducted a survey of over 600 business professionals. Across all industries, the survey results estimate that data quality problems cost U.S. corporations more than $ 600 billion per year. Proactive steps need to be taken to identify, isolate and report clinical data issues using a system that is flexible, easy to update and facilitates good communication with the Clinical Data Management (CDM) department to help resolve these data quality problems.

This paper will review an effective method to implement a clinical data acceptance testing procedure using edit check macros for creating an RTF file with minimum SAS® expertise and maintenance. In addition, because all clinical studies have common issues, the edit check macros developed could easily be used to check similar data issues across other clinical studies.

THE PROBLEM WITH DATA ISSUES

In general, the CDM department may not spend enough resources to check the quality of the data. This is because CDM's main responsibility is to collect and structure the incoming data. Since the biostatistics department is generally responsible for the final study results, they must often exercise control on data quality before accepting the raw clinical data. The problem often occurs when SAS statistical programmers and statisticians in the biostatistics department process the original 'unchecked' clinical data to get incorrect results and conclusions. For example, even simple checks such as viewing invalid values for the variable gender are not performed. This could result in confusion and frustration.

According to the 2001 survey by the Data Warehousing Institute in figure 1, the sources of data quality problems across all industries can be identified below. It is interesting to note that while most data issues are caused by data entry errors, there is still a substantial amount of data issues that are caused by system related changes, conversions or errors. This indicates that similar types of validation checks should be applied throughout the process of data collection, storage, transfer, conversion and update. For clinical trials, various studies suggest that up to 5 percent of raw data values in clinical trial databases are erroneous initially.

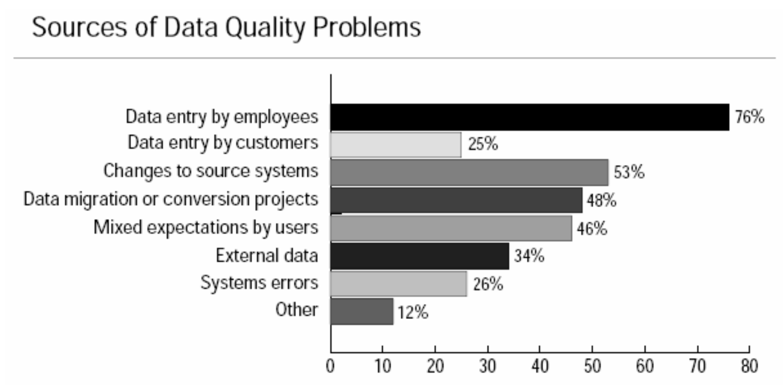


Figure 1. Sources of Data Quality Problems across all Industries

Examples of using 'unchecked' data that resulted in significant delays and costs include:

- In February 2003, the U.S. Treasury Department mailed 50,000 Social Security checks without a beneficiary name. The missing names data issue was due to a software program maintenance error.
- In October 1999, the $ 125 million NASA Mars Climate Orbiter, an interplanetary weather satellite, was lost in space due to a data conversion error. The data issue was due to performing certain calculations in English units (yards) when it should have used metric units (meters).

Specifically, this paper will review an effective method to implement a clinical data acceptance testing procedure to check data quality with each data transfer, conversion or update. The two main categories of clinical data issues may be grouped as incorrect and incomplete data. In general, incorrect data issues consist of unexpected raw values, invalid raw values, incorrect conversion of raw values or inconsistent raw values with another variable or record. Also, incomplete data issues consist of missing values when required.

THE SOLUTION TO RESOLVE DATA ISSUES

As SAS statistical programmers, you can easily write programs to list all unique values of the gender variable, for example, to inform the team that an invalid value exists for that variable. Once you can isolate clinical data issues, they become 'known' and can be 'accounted for' to explain differences in expectations and conflicts. Implementing the clinical data acceptance testing procedure involves developing a collection of single purpose macros with basic requirements. Once the system is in place for one clinical study, multiple studies could also be checked as a universal set of macros since the checks are all repetitive and standard.

The benefits of using these macros are increased productivity by quickly and easily apply the macros to other clinical studies, the acceptance of CDM to use the systematic approach method of communicating common issues/concerns, and the biostatistics department having more confidence in the raw clinical data. The end result is that deadlines are not missed since SAS programs do not have to be written defensibly to account for these data issues.

According to the same 2001 survey by the Data Warehousing Institute in figure 2, the benefits of high quality data across all industries can be identified below. During the FDA submission process, a single version of the truth and increased customer satisfaction are very important to recognize reduced costs and minimum delays to get the drug approved. These outcomes are well worth the average cost of $20 to $25 per case report form page or up to 15 % of the clinical research budget to ensure data quality.

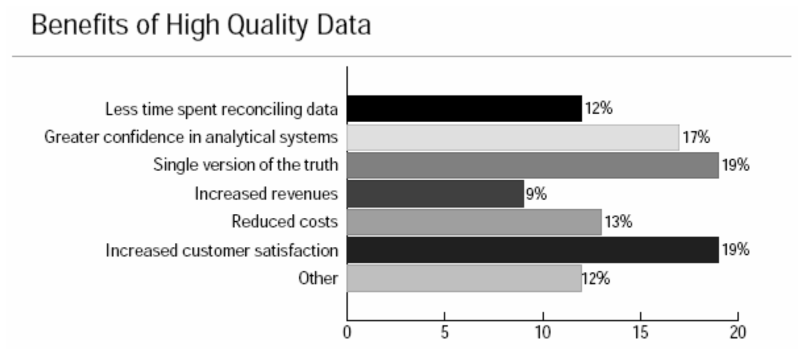
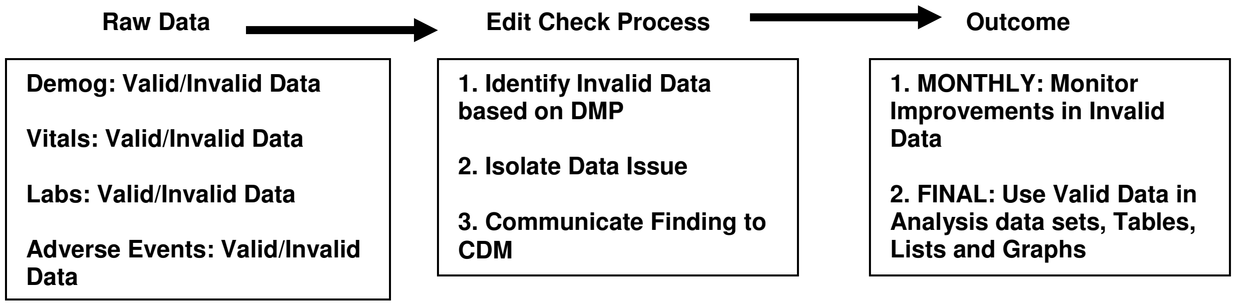


Figure 2. Benefits of High Quality Data across all Industries

Overall, the process flow consists of accessing raw data, which may contain invalid data, with edit check macros to monitor data issues so that only valid data is used in the final analysis data sets, tables, lists and graphs. With this solution, if invalid data is used in the outcome, then the unexpected results can be explained.


Raw Data
Edit Check Process
Outcome
Demog: Valid/Invalid Data
Vitals: Valid/Invalid Data
Labs: Valid/Invalid Data
Adverse Events: Valid/Invalid Data
1. Identify Invalid Data based on DMP
2. Isolate Data Issue
3. Communicate Finding to CDM
1. MONTHLY: Monitor Improvements in Invalid Data
2. FINAL: Use Valid Data in Analysis data sets, Tables, Lists and Graphs

Specifically, the solution involves these four steps before having the database lock:
1. Specifying Requirements in Data Management Plan (DMP)
2. Developing and Testing Edit Check Macros
3. Communicating Results with Clinical Data Management (CDM)
4. Monitoring the Metrics of Data Issues

SPECIFYING REQUIREMENTS IN DATA MANAGEMENT PLAN (DMP)

The first critical step in data acceptance testing is to specify the requirements in a Data Management Plan (DMP). Within the DMP, the requirements should be clear and complete for all possible data issues. It will be helpful for the subject matter expert to use the case report forms and the protocol when developing the requirements. In addition, often, important variables used in tables, lists and graphs maybe included in the DMP. The data checks performed should check each of the conditions specified for each clinical raw data set.

As a minimum, for example, the DMP should include the following clinical data checks:

1. All unique key variables in each raw data set are required.
    - Example: Patient ID variable is non-missing and unique.
2. Confirm minimum and maximum values of selected variables.
    - Example: Demog data set: valid age values within lower and upper range values. Lab data set: valid toxicity and hemoglobin values within lower and upper range values. Vitals data set: valid temperature and blood pressure values within lower and upper range values.
3. Display all unique values of selected variables.
    - Example: Demog data set: valid treatment (active, placebo).
4. Display values of selected variables to meet specific database queries.
    - Example: Endpt data set: valid primary and secondary variables.
5. Confirm the logic between two variables.
    - Example: Adverse Events data set: adverse event description, preferred term, and system organ class are required variables if any are non-missing.
6. Confirm the consistency between two clinical dates.
    - Example: Adverse Events data set: Adverse start dates before or same day as adverse stop dates.
7. Check for duplicate records.
8. Compare and identify differences of common variables between two data sets.
    - Example: Raw Adverse Events data set and Analysis Adverse Events data set.

In addition to the minimum checks to perform, these additional checks help ensure a more successful clinical study by monitoring important clinical issues:

1. Are there any protocol violations that should be excluded from analysis?
2. Are the treatment groups randomly distributed based on safety subset population?
3. Have lab values been correctly converted from reported units to standard international units?
4. For each lab, are the normal range flags correct based on the lower and upper range values?
5. For each lab, are there major deviations in value from baseline over time?
6. For each lab data transfer, are patients correctly identified?
7. Are the top 10 adverse events expected?
8. Are patient follow-up visit windows in compliance with the protocol?
9. For any critical variable, are there any outliers?

By applying standard edit check macros to perform standard data checks, more time can be spent on investigating the unique and more complex data issues of clinical studies. In addition, the focus for SAS statistical programmers is on generating more data checks since it is easy to copy a single edit check macro call in one SAS line instead of copying a block of SAS code for each new data check. In the Unix environment, the process of copying and pasting a single SAS line takes only two key strokes. In addition, any edit check macro call can be turned on or off with the asterisk character '*'.

A secondary benefit is that the traditionally very lengthy SAS program is now much easier to maintain since it is easier to read and update. As an example, for one clinical study, the clinical data acceptance testing program contains 75 edit check macro calls. The number of SAS lines in this program is only 150 lines as compared to over 2,000 lines (150 x 14) with traditional SAS programming techniques. For better organization, these edit check macro calls are grouped by the raw clinical data set checked and then by the type of data check performed. Using edit check macros improves productivity of standard data checks by at least 80%.

DEVELOPING AND TESTING EDIT CHECK MACROS

The challenge is to develop a system that is flexible enough to not run selected data checks, modify data checks and add new data checks as requested. In addition, it is important for the system to display the message 'No records found' to confirm that the data quality check was performed and that data assumptions were met. Finally, feedback from CDM needed to be incorporated with the reporting process for each data issue identified. This is important to prevent 're-inventing the wheel'.

Due to limited programming resources and short timelines, a simple approach using ODS and minimum SAS macro programming is taken to develop and test task-oriented macros. Some macros are individually associated with selected SAS procedures such as Proc Freq, Proc Means, Proc Compare and Proc Print. These macros offered the ability to apply the most frequently used SAS procedure options as standards with a consistent reporting layout.

When designing the macros, the following were functional requirements:

A. Macros use basic macro programming techniques that are easy to understand
- Quick development of new macros
- Quick enhancements to existing macros

B. Macros provide informative feedback in titles
- Input data set name
- Variables checked
- Any subset condition applied

C. Macros provide reference information in footnotes
- Program name
- Output file name
- Date executed

D. Macros display data issues
- Patient and visit identification
- Data values of variable checked
- Supporting variables (if any)
- One data issue/page
- Findings saved to one RTF file

Essentially, the input parameters to the edit check macros are directly applied in the SAS procedure and displayed in the title since it is important to display the source data set, the variable being checked and the condition to report the data issue. These are important details for CDM to first confirm the biostatistics data issue finding before taking any action. As footnotes, the date executed and the source program name are displayed so that exact raw snapshot data sets could be compared.

To meet the requirements of the DMP, data issues are categorized so that edit check macros would have specific functions.

In general, the edit check macros are designed to meet the following types of data issues:

Type of Data Issue	**Brief Description**
Acceptable Values	Values are one of the valid values for variable
Character Formats	Format of values within character variables are as expected
Consistency Across Variables	Values are consistent across multiple variables
Consistency Across Data sets	Values are consistent across multiple data sets
Non-Duplicate Records	Each record is unique and not duplicated
Protocol Compliance Rules	Study specific logic-based check to confirm data compliance, ex. lab conversion
Range Check	Values are within a specified range
Required Value	Value is non-missing
Unique Value	Values are unique

Below are brief descriptions of selected edit check macros. To help better apply these edit check macros, a user guide with all macro parameters and example macro calls was developed.

Brief description of selected edit check macros

Macro	**Brief Description**
%crt_comp	Compare data values and variable attributes of two data sets (Proc Compare).
%exprpt	Exception Reporting: data set exists?, variable exists?, records exists?, non-missing values exist?
%negval	Check for negative values.
%subqry	Display selected variables in one data set based on condition in another data set, useful for checking data across data sets (Proc SQL with subquery). Note that this is an exception macro since most all other edit check macros can only process a single data set.
%u_eccust	Display selected variables based on customized user conditions using a single IF statement. Note that this is an exception macro since most all other edit check macros use the WHERE statement.
%u_ecdup	Check for duplicate records.
%u_ecfreq	Display frequency of selected single or multiple crossed variables (Proc Freq).
%u_ecmens	Display descriptive statistics including range values of continuous variables (Proc Means).
%u_ecprnt	Display selected variables with selected conditions (Proc Print).

Below are general rules when applying the edit check macros:

1. The variable patient is required in almost all macros.
2. All variables must be in a single data set to use macros except for macros such as %subqry which allows for two data sets.
3. It is recommended to create the macro variable ‘study’ to be used in the title of the output.
4. In many cases, data set option can be applied to the input data set for additional flexibility.
5. Most macros allow for subset condition using a valid ‘where statement’ that is flexible to handle most any expression with function or calculation. The %u_eccust macro, however, uses the ‘if statement’. For many macros, other valid SAS statements such as the ‘format statement’ may be used with or without the ‘where statement’.

To apply standardization across the edit check macros, standard macro parameters are used.

Basic edit check macro call with standard macro parameters

```
%let study = proto1;  < libref of data sets to check – raw or analysis data sets >

%edit_check_macro(

       edsn = &study.XXX < data set name with optional data set option or if used
                with a SAS procedure then a SAS procedure option >,
       ecvr = < one or more edit check variable names >,
       byvars = < variable such as patient for by-group processing >,
       ecvrl = < one or more extra supporting variable names to display >,
       esub = < subset condition using “where statement” >,
       fnote = < user defined message to better understand data check > );
```

To understand the concept of the simple approach taken to develop these edit check macros, the %u_ecfreq macro is displayed below. The %u_ecfreq macro is used to display frequency of selected single or multiple crossed variables. It is one of the most frequently used macros since it easily sorts and counts all unique combinations of variables. This becomes helpful, for example, to validate one variable derived from another variable. With the ‘list’ and ‘missing’ Proc Freq options, the unique variable combinations are listed in rows instead of in separate columns.

In the %u_ecfreq macro call below, all of the variables, from the keyword ‘_all_’, in the ptinfo data set in the &study libref will be individually summarized. There are no subset conditions applied so all records will be included. The title displays the study number, data set accessed, variable checked and the subset condition applied if any.

In addition, only the default footnote will be displayed. Notice that multiple user supplied footnotes can be applied with the ‘\’ delimiter. The footnotes block of code is conditionally applied if &fnote macro parameter is specified. Finally, the &ecvr macro parameter is directly applied to the ‘table’ statement and the &esub macro parameter is a separate independent SAS statement within Proc Freq. Since statements such as ‘where’ or ‘format’ are valid SAS procedure statements, the macro accepts these as input macro parameters. In addition, with the inclusion of the ‘;’, more than one SAS statement can be specified for additional flexibility.

SAS code for %u_ecfreq macro

```
%macro u_ecfreq(
         edsn = &study.ptinfo,   /* dataset name */
         ecvr = %str(_all_/list missing),  /* variable or list of variables also,
                                  can pass cross tables with options */
         esub = ,                /* subset where condition,
                                  can also be a format statement */
         fnote =                 /* optional footnotes with ‘\’ delimiter */
         );

 title1 "Study: &study - Edit Check in &edsn dataset";
 title2 "PROC FREQ of %upcase(&ecvr) variable with the condition: &esub";

 %if %length(&fnote)>0 %then %do;
    %if %index(&fnote,\)=0 %then footnote3 &fnote.;;
       %if %index(&fnote,\)>0 %then %do;
         %let fnote3=%scan(&fnote,1,\);

         footnote3 &fnote3;
         %let fnote4=%scan(&fnote,2,\);
         footnote4 &fnote4;

       %end;
 %end;

 proc freq data=&edsn;
  tables &ecvr;
  &esub;
 run;

footnote3;

%mend u_ecfreq;
```

Below is a %u_ecfreq macro call to display sex and race unique values. Notice that aside from the footnote, the complete macro call takes only one SAS line which can easily be copied and modified for other data checks.

Example of %u_ecfreq macro call to check for invalid Sex and Race values

```
%u_ecfreq(edsn = &study..ptinfo, ecvr = sex race/list missing, esub =,
    fnote = %str(Note to CDM: Confirm valid gender\Note to CDM: Confirm valid race) );
```

The output from these edit check macros is standardized to contain all of the relevant information needed by CDM to confirm the data issue finding before taking any action: study number (proto1), data set name (ptinfo), variable checked (sex, race), subset condition (if any), patient number (if applicable), visit date (if applicable), unexpected data values, supporting variables (if any), program name (t_data_acceptance_checks.sas) and date checked (12MAY2007), and user specified footnote (Note to CDM) to communicate data issue found. Note that each edit check macro call displays results on the next page so that the data checks are not confused.

As can be seen from the results below, invalid sex values ‘Mal’ and invalid race values ‘X’ exist in the ptinfo data set. Once CDM identifies the source of these data issues, then the ‘Note to CDM’ footnote can be updated to document these data issue.

Output of %u_ecfreq macro call

```
Study: proto1 - Edit Check in proto1.ptinfo dataset
PROC FREQ of SEX RACE/LIST MISSING variable with the condition:

The FREQ Procedure

                                     Cumulative    Cumulative
sex        Frequency      Percent     Frequency      Percent
_____________________________________________________________

Female             4        40.00             4        40.00
Mal                2        20.00             6        60.00
```

```
Male             4       40.00           10       100.00



                                     Cumulative    Cumulative
race      Frequency      Percent      Frequency       Percent
──────────────────────────────────────────────────────────────
Asian            1       10.00            1        10.00
Black            1       10.00            2        20.00
White            7       70.00            9        90.00
X                1       10.00           10       100.00

Program: t_data_acceptance_checks.sas  Output: edit_check_results.rtf    12MAY2007

Note to CDM: Confirm valid gender
Note to CDM: Confirm valid race
```

The %crt_comp macro is used to compare common variables between raw and analysis data sets. For common variables with different names, the variable names from both data sets need to be specified in the same corresponding order. For common variables with the same name, an option exists to automatically identify these common variables. This automatic feature saves programmer's time from this administrative task. Adding this automatic feature does, however, require some knowledge of the SAS data dictionary tables.

The %crt_comp macro is useful to handle most all cases of comparing two data sets. Not only can permanent or temporary data sets be compared, but the %crt_comp macro can apply subset conditions or data set options on each data set, check common and uncommon variables, remove duplicate records, and apply Proc Compare standard options. Note that the %crt_comp macro requires knowledge of Proc SQL and Proc Compare and is one of the more complicated edit check macros.

In the %crt_comp macro call below, all of the input parameters are specified. Note that the %str() macro function may be required if any input parameter contains special characters such as '=', the equal sign or ';', the semicolon.

Example of blank %crt_comp macro call with all input macro parameters

```
%crt_comp(
   ec1dir =,            /* prefix study directory data set 1 */
   ec1dsn =,            /* data set name 1 */
   ec1var =,            /* variable names from data set 1 */

   ec2dir =,            /* prefix study directory data set 2 */
   ec2dsn =,            /* data set name 2 */
   eco2dsn =,           /* output data set name 2 */
   ecr2dsn = 0,         /* remove duplicate records 0 = no, 1 = yes */
   ec2var =,            /* variable names from data set 2 */

   ecvrlt =,            /* common variable names to compare or _all_ */

   ecid   =,            /* matching variable name for data set 1 */
   ecid2  =,            /* matching variable name for data set 2 */

   ec1wst =,           /* optional where condition on data set 1 */
   ec2wst =,           /* optional where condition on data set 2 */

   ec1opt =,           /* optional option condition on data set 1 */
   ec2opt =,           /* optional option condition on data set 2 */

   cmpopt = %str(method=exact)  /* proc compare method option - brief or all */
   );
```

In the single, one line %crt_comp macro call below, a comprehensive Proc Compare using the results of the data dictionary tables is performed. This macro has also been very useful in the validation of analysis data sets created from raw data sets.

Example of %crt_comp macro call to compare ptinfo and rptinfo data sets

```
%crt_comp(ec1dir=a,ec1dsn=ptinfo,ecid=patient,ec2dir=a,ec2dsn=rptinfo,ecid2=patient);
```

The results of the %crt_comp macro call to compare two data sets ptinfo and the corresponding raw data set rptinfo show that the macro automatically identified and compared the four common variables between the two data sets. As shown below, the two data sets have the same number of variables and observations. For the non-missing id variable patient, there are differences in values for sex and age. Since the raw data set, rptinfo, has valid values for sex and age, there must be a conversion problem when creating sex and age variables in the ptinfo data set.

Output of %crt_comp macro call

```
Study: proto1 - Edit Check in proto1.ptinfo dataset
Compare aproto1: BASE Dataset ptinfo with aproto1: COMPARE Dataset rptinfo
Compare common variables


Column Name
________________________________

patient
age
sex
race


Study: proto1 - Edit Check in proto1.ptinfo dataset
Compare aproto1: BASE Dataset ptinfo with aproto1: COMPARE Dataset rptinfo
Compare common variables

The COMPARE Procedure
Comparison of WORK.PTINFO with WORK.RPTINFO
(Method=EXACT)



Data Set Summary

Dataset                Created          Modified  NVar    NObs

WORK.PTINFO   21MAY07:06:00:52  21MAY07:06:00:52     4      10
WORK.RPTINFO  21MAY07:06:00:52  21MAY07:06:00:52     4      10


Variables Summary

Number of Variables in Common: 4.
Number of ID Variables: 1.
Number of VAR Statement Variables: 4.


Observation Summary

Observation      Base  Compare  ID

First Obs           1        .  patient=.
                    .        1  patient=101
First Match         2        2  patient=102
First Unequal       2        2  patient=102
Last  Unequal       3        3  patient=105
Last  Obs          10       10  patient=115

Number of Observations in Common: 9.
Number of Observations in WORK.PTINFO but not in WORK.RPTINFO: 1.
Number of Observations in WORK.RPTINFO but not in WORK.PTINFO: 1.
Total Number of Observations Read from WORK.PTINFO: 10.
Total Number of Observations Read from WORK.RPTINFO: 10.

Number of Observations with Some Compared Variables Unequal: 2.
Number of Observations with All Compared Variables Equal: 7.

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 2.
Number of Variables Compared with Some Observations Unequal: 2.
Total Number of Values which Compare Unequal: 3.
Maximum Difference: 40.


Variables with Unequal Values

Variable  Type  Len  Ndif   MaxDif

age       NUM     3     2   40.000
sex       CHAR   25     1

Value Comparison Results for Variables
_________________________________________________________
          ||       Base    Compare
 patient  ||        age        age      Diff.     % Diff
 _______  ||  _________  _________  _________  _________
          ||
     102  ||          0    40.0000    40.0000          .
     105  ||    -2.0000    20.0000    22.0000      -1100
_________________________________________________________


_________________________________________________________
          ||  Base Value           Compare Value
 patient  ||  sex                   sex
 _______  ||  ___________________+  ___________________+
          ||
     102  ||  Mal                   Male
_________________________________________________________
```

The %u_ecdup macro is used to check for duplicate records. This standard check should be applied to all raw and analysis data sets. With a simple macro call in one SAS line, this becomes an easy task to perform for all raw and analysis data sets. The two types of duplicate records check that can be performed using %u_ecdup macro are for all variables in the data set and for key variables.

The '_all_' keyword can be used when the key variables are unknown or if known, then key variables can be confirmed. Options exist to create a data set of unique key variables along with any supporting variables. While the %u_ecfreq macro contains only one Proc Freq, the %u_ecdup macro is a little more complex since it contains five SAS procedures and two data steps.

Example of %u_ecdup macro call to check for duplicate records

```
* Check for duplicate records by all variables;
%u_ecdup(
   edsn = &study..ptinfo,   /* Data set name */
   ecvr = _all_,            /* Error check variables, default is all variables */
   ecvrl =,                 /* Extra variables to be kept in the output dataset */
   fnote =,                 /* Optional footnote */
   eoutdsn =                /* Output dataset */
   );

* Check for duplicate records by key variables;
%u_ecdup(edsn = &study..ptinfo, ecvr = patient);
```

One of the important benefits of using these macros is that a message is always displayed whether the data issue exists or not. This helps to confirm that the specific data issue is in fact programmed and that no data issue was found. Since clinical data dumps occur on a monthly basis, data checks that are clean one month may not be clean the next month.

Output of %u_ecdup macro call

```
Study: proto1 - Edit Check in proto1.ptinfo dataset
Indicate the result of checking for duplicated records

             Error Code

No duplicated records found for _ALL_

Program: t_data_acceptance_checks.sas  Output: edit_check_results.rtf    12MAY2007

Study: proto1 - Edit Check in proto1.ptinfo dataset               (Results on next page)
Indicate the result of checking for duplicated records

              Error Code

No duplicated records found for PATIENT

Program: t_data_acceptance_checks.sas  Output: edit_check_results.rtf    12MAY2007
```

The %u_eccust macro is used to check for almost any customized data checks. This macro allows for more complex checks that can be performed using a valid single 'if' statement.

For example, the macro call below checks for incorrect hemoglobin normal range flags. In the 'if' statement, there are four conditions applied any of which will display data issues once the first condition is met:

1. Non-missing hehgb, hehgbl and hehgbh values
2. If 'L' flag is incorrectly assigned based on lab value and lower lab value range
3. If 'H' flag is incorrectly assigned based on lab value and higher lab value range
4. If 'N' flag is incorrectly assigned based on lab value and lower and higher lab value ranges

Example of %u_eccust macro call

```
%u_eccust(
     edsn = c990736.vlabs,          /* Harmonized dataset name */
     ecvr = labdt,                  /* Variable name for edit check */
     ecst = %str(if labnme='wbc' and not ((.1 < labv < 100))),
                                   /* Actual If condition statement with NOT built in */
     titles =,                      /* Extra Title */
     fnote =,                       /* Optional footnote */
   );

%u_eccust(edsn = h&study..hematol
         ,ecvr  = hehgb hehgbf hehgbl hehgbh
         ,ecst  = if nmiss(hehgb, hehgbl, hehgbh) = 0
                  and ((hehgb < hehgbl and hehgbf ne 'L')
                  or (hehgb > hehgbh and hehgbf ne 'H')
                  or (hehgbl <= hehgb <= hehgbh
                  and hehgbf ne 'N')) );
```

As discussed, one of the advantages of having the edit check macro call on one SAS line is to easily copy the line and change the variable name or subset condition for the next edit check. Below are the multiple %u_ecfreq macro calls from the production edit check program to check for missing or negative values of 14 hematology lab variables. As a SAS statistical programmer, this code is not only easier to read, but also easier to maintain. The 14 SAS lines below would have taken over 200 lines (14 x 15) in a traditional SAS program. As you can see with this approach, data checks are easy to produce with minimum SAS programming.

Multiple %u_ecfreq macro calls

```
%u_ecfreq(edsn=&study..hematol, ecvr=hewbc, esub=where hewbc <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=heneut, esub=where heneut <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=hesegs, esub=where hesegs <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=heband, esub=where heband <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=hegran, esub=where hegran <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=hebaso, esub=where hebaso <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=heeosn, esub=where heeosn <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=helymp, esub=where helymp <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=hemono, esub=where hemono <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=heatyp, esub=where heatyp <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=hemeta, esub=where hemeta <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=hemylc, esub=where hemylc <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=heprmy, esub=where heprmy <= .);
%u_ecfreq(edsn=&study..hematol, ecvr=hemylb, esub=where hemylb <= .);
```

Below are the multiple %u_ecmens macro calls from the production edit check program to check for descriptive statistics and range values of 20 lab variables. Since the results of each lab variable are saved on individual pages, specific pages in the RTF file can be referenced when identifying data issues in a summary report. The 20 SAS lines below would have taken over 350 lines (20 x 18) in a traditional SAS program.

Multiple %u_ecmens macro calls

```
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='wbc');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='neut');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='segs');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='band');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='gran');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='baso');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='eosn');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='lymp');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='mono');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='atyp');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='meta');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='mylc');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='prmy');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='mylb');
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='hct', byvars=class);
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='hgb', byvars=class);
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='plt', byvars=class);
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='rbc', byvars=class);
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='wbc', byvars=class);
%u_ecmens(edsn=&study..vlabs, ecvr=labv, esub=where labnme='anc', byvars=class);
```

COMMUNICATING RESULTS WITH CLINICAL DATA MANAGEMENT (CDM)

To have an effective system, all parties involved need to work together to improve communication, coordination and cooperation. The data issues identified by the biostatistics department are only as good as they are timely communicated in a clear, reproducible method to be received for some type of data corrective action. In general, before CDM takes any data corrective action, their first step is to confirm *your* findings of the raw clinical data that they supplied. If a discrepancy exists between what the biostatistics department expects and what CDM supplies, then an adjustment to the database design and system may be required.

Each edit check macro is designed to provide all of the following required information needed to identify data issues: study number, data set name, variable checked, subset condition (if any), patient number (if applicable), visit date (if applicable), unexpected data values, supporting variables (if any) and date checked. The RTF file containing one page for each data issue can be e-mailed to CDM for their review. Once CDM accepts the data issues, then time can be spent to resolve and prevent these data issues. This requires coordination with the source of the raw clinical data to educate the staff and possible cooperation with application developers to prevent data entry and inconsistent data errors. It may be that additional training on how to complete more difficult sections of case report forms is required to prevent selected data issues. If possible, SOPs should include standard processes for identifying and reporting data issues and validation rules should be applied at the source of data entry to better resolve data issues.

For some special cases, it may be acceptable to correct obvious data issues. For these cases, it is strongly recommended to clearly specify what conditions and what changes can be performed. In addition, upper senior management must approve these data mapping changes. Finally, the original raw clinical data value should always be kept for auditing purposes so that the new data value is stored in another variable. An example of an obvious data issue could be having a blank or 'No' to the question "Has the patient had any Adverse Events?" and adverse event records actually existing for that patient. It may be acceptable to change the blank or 'No' to 'Yes'.

As an alternative, a new derived variable could be created based on the condition if any adverse event record exists, then value is 'Yes' and the original raw value is kept unchanged in another variable not used in analysis. This is the preferred method used when converting lab values from reported units to standard units. As always, this data correction should be checked as part of the edit check process. In this case, these two conditions should be consistent with each other.

MONITORING THE METRICS OF DATA ISSUES

Very often, because raw data dumps occur on a monthly basis, data checks can be monitored with each raw data dump. By creating a metrics tracking system, the monitoring of data checks becomes more standardized. The metrics tracking system also serves as a report card that can be communicated to all parties involved including senior management. In general, the metrics tracking system consists of the timeliness of the data dump, accuracy of the data, completeness of the information, and consistency of the data across studies.

On a monthly basis before database lock, the goal is to monitor improvements such as reduction of invalid or inaccurate data and increase of required data. Note that the true accuracy of the raw clinical data can only be determined by comparing the raw data set with the case report forms which the data was entered from. In general, it is not the responsibility of the biostatistics department to perform an audit using the case report forms.

One approach for measuring the quality of information in the metrics tracking system is by the type and number of failed data acceptance tests. In general, for any critical data checks performed, all tests must pass at database lock time before fully accepting the clinical data. With this requirement, none or minimum data issues are expected in the final study results.

Specifically, several methods exist to determine the error rate calculation of the data issues identified. The numerator and denominator used in the error rate calculation depend on the unit base selected. In general, the unit base could be one of the following three options: total number of data checks performed, total number of records checked, or total number of patients checked.

Using the ptinfo data set of 10 records as an example, below are three unit base options for error rate calculation:

1. Total number of data checks performed

   a. Number of checks failed/total number of checks performed, ex. 4/5, 80 %

   In this data set of 10 records and 10 patients, 80% of the 5 data checks performed failed. This indicates that at least one patient has multiple data issues across all types of clinical data. The scope of data issues is not good for these patients. Note that this metric is similar to a moving target since checks can be dropped, modified or added.

   b. Number of failed records/(total number of records x total number of checks performed), ex. 6/(10 x 5), 12 %

   In this data set of 10 records and 10 patients, 6 failed records from the 5 data checks indicates that not all patients have data issues. Based on this metric value, only 12 % of the records checked have data issues.

   c. By the data check performed: Number of failed records/(total number of records x total number of checks performed)

Ptinfo data set with 5 data checks performed: 10 patients (1 record/patient)

Data Check Performed	Records Passed Test	Records Failed Test	% Records Failed
1. ptno is valid	9	1	10 %
2. age is valid	8	2	20 %
3. sex is valid	8	2	20 %
4. race is valid	9	1	10 %
5. no duplicate records	10	0	0 %
Total	44	6	12 %

The table above indicates that age and sex had the most data issues and that no duplicate records exist in the data set.

2. Total number of records checked

   Number of problem records/total number of records in data set, ex. 3/10, 30 %

   Ptinfo data set records

```
Obs     ptno     age     sex        race

  1        .      35     Mal        X
  2      102       0     Mal        White
  3      105      -2     Female     White
  4      107      30     Male       Black
  5      110      31     Male       White
  6      111      28     Female     White
  7      112      33     Female     White
  8      113      35     Male       Asian
  9      114      42     Male       White
 10      115      54     Female     White
```

   This method gives, at best, only a high level view of the data issue since it combines all data issues as a single data issue per record. While the 30 % failure rate is more than double the 12 % failure rate using the previous method, this approach could have been based on 1 or 100 data checks. It will be more useful to include the number of checks performed at a summary level so that a detailed breakdown can then be future analyzed.

3. Total number of patients checked

   Number of patients with data issues/total number of patients in data set, ex. 3/10, 30 %

   Since this method is similar to the second method of combining all data issues as a single data issue per patient, the same reasoning as above is also applicable to this error rate calculation. Note that in this simple example, the 30 % matches the error rate calculation of the second method because the data set has one record per patient.

The total number of data checks performed method is selected as the unit base to more accurately track the improvements in data issues. This is the only unit base method that takes into consideration the amount of time it takes to SAS statistical programmers to write edit checks instead of assuming that all edit checks are automatically available without a need to unselect, modify or add more data checks. This unit base method provides information on the scope of data issues as well as the impact of each data issue.

If the type of failed data acceptance test indicates a systemic problem, then corrective action needs to be taken immediately to fix the clinical data and to fix the clinical data process component that failed. An example of this would be the incorrect conversion of clinical lab values from reported units to standard units.

The costs associated with processing poor clinical data can be measured by submission delays as well as making incorrect conclusions in the clinical study report submission for product approval. While submission delays can cost pharmaceutical companies many millions of dollars in lost sales per day, the cost of submitting invalid clinical study results may be much more. In addition, the cost increases as patient lives become affected.

Without the established data acceptance testing process, more time is spent on trying to develop and validate tables, lists and graphs that contain data issues, which are the cause for delays, confusions and misinterpretations of the clinical results. Because the reputation of SAS statistical programmers and statisticians on the team are at stake, it is more important to first identify clinical data issues before generating the final analysis data sets, tables, lists or graphs.

SUMMARY

By implementing a simple, yet effective solution to this major problem, the biostatistics department is able to allocate minimum resources with minimum SAS statistical programming experience to update and add new edit checks for all clinical studies. With the addition of the e-mail notification, the biostatistics department is able to take actions more quickly and communicate the data issues to the team more effectively. In addition, because these techniques to review clinical data are effective, some edit check macros are also useful for analysis data set validation.

At a very high level, clinical data acceptance testing procedure for data can be considered similar to the traditional user acceptance testing for system applications. The same principles of good requirements, valid data and coding and comprehensive testing should be applied in both cases. SOPs and guidelines need to be written to assure proper steps are in place for processing correct clinical data. All team members are responsible for being trained and following these SOPs and guidelines.

It is important to note that the data acceptance testing procedure assumes that the data in the data sets accurately reflects the case report forms since the auditing of the case report forms is a separate task. The edit check macros are very useful to confirm the consistency and validity of the clinical data to make valid critical study conclusions. Once mission-critical data acceptance tests fail, the team is automatically informed so that clinical results based on the current data is not analyzed without first making data corrections.

Future plans to enhance the edit check macros are to use SAS version 9.0 functions. The new SAS version 9.0 functions can be used to easily identify unexpected data values in character variables. For example, the following functions ANYALNUM(), ANYALPHA(), ANYDIGIT(), and ANYPUNCT() will return the first location of the alphanumeric, letter, digit or punctuation corresponding. In addition, the corresponding inverse of these functions are NOTALNUM(), NOTALPHA(), and NOTDIGIT().

REFERENCES

Bentley, John E., "Software Testing Fundamentals – Concepts, Roles, and Terminology", SUGI 30, Planning, Development and Support

Brunelle, Rocco and Robert Kleyle, "A Database Quality Review Process with Interim Checks", Drug Information Journal, Vol. 36, pp. 357-367, 2002

Ciambrone, Katherine R., "Fit for Reporting", Data Basics, Society for Clinical Data Management, Summer 2004 newsletter, http://www.scdm.org

Cody, Ron. 1999. *Cody's Data Cleaning Techniques Using SAS® Software*, Cary, NC: SAS Institute Inc.

Collins, Sylva H., "Ensuring Quality Data", Data Basics, Society for Clinical Data Management, Summer 2007 newsletter, http://www.scdm.org

Doles, Wanda, "Managing Laboratory Data", Data Basics, Society for Clinical Data Management, Spring 2004 newsletter, http://www.scdm.org

Eckerson, Wayne, 'Data Quality and the Bottom Line', The Data Warehousing Institute, 2002 http://www.dataflux.com/Resources/file-stream.asp?rid=37

English, Larry, 'Don't just comply; Prevent', SAS.com, http://www.sas.com/news/sascom/2005q3/column_guestenglish.html

English, Larry, Information Impact International Inc, http://www.infoimpact.com/

FDA, 21 CFR Part 11, Electronic Records, Electronic Signatures; Final Rule, Federal Register, Vol. 62, No. 54, 13429, March 20, 1997.

Fendt, Kaye H., "The Case for Clinical Data Quality", Data Basics, Society for Clinical Data Management, Summer 2004 newsletter, http://www.scdm.org

Gupta, Sunil. 2006. *Data Management and Reporting Made Easy Using SAS® Learning Edition 2.0*. Cary, NC: SAS Institute Inc.

Gupta, Sunil and Curt Edmonds, *Sharpening Your SAS® Skills*, Boca Raton, FL: Chapman & Hall/CRC Press, 2005

Nahm, Meredith, “Data Gone Awry”, Data Basics, Society for Clinical Data Management, Fall 2007 newsletter, http://www.scdm.org

Nahm, Meredith, Dziem, Greg, Fendt, Kaye, Freeman, Lisa, Masi, Joann, Ponce, Zoe, “Data Quality Survey Results”, Data Basics, Society for Clinical Data Management, Summer 2004 newsletter, http://www.scdm.org

Raeburn, Vicki, “Measuring Customer Data Quality”, DM Review, December 2007

Shiang, Keh-Dong, “PROC SQL: A Powerful tool to Improve Your Data Quality”, SUGI 31, Posters

Silva, Greg, “Consistent Variables + Consistent Checks = Cleaner Data”, PharmaSUG 2005

Stephens, R. Todd, “Collaboration by Data Organizations”, DM Review, December 2007

CONTACT INFORMATION

The author welcomes your comments and suggestions.

Sunil K. Gupta
Associate Director, Statistical Programming
Quintiles
325 E. Hillcrest Dr., Suite 200
Thousand Oaks, CA 91360
Phone: (805)-557-7700 E-mail: Sunil.Gupta@Quintiles.com

Sunil is the Associate Director, Statistical Programming at Quintiles. He has been using SAS® software for over 14 years and is a SAS Base Certified Professional. He has participated in over 6 successful FDA submissions. His projects with pharmaceutical companies include the development of a Macro-Based Application for Report Generation and Customized Plots and Charts. He is also the author of *Quick Results with the Output Delivery System*, developer of over five SAS programming classes, developer of Clinical Trial Reporting Templates for quick generation of tables, lists and graphs and was a SAS Institute Quality Partner™ for over 5 years. Most recently, he released his new book, *Data Management and Reporting Made Easy with SAS Learning Edition 2.0,* and has co-authored the book *Sharpening Your SAS Skills.* Currently, is teaching his new class, Best Practices in SAS Statistical Programming for Regulatory Submission.

sas® Certified Base Programmer