Robust Approach for Handling Data Issues Discovered During Statistical Programming

Vineet Jain, Smith Hanley Consulting Group, Jersey City, NJ

ABSTRACT
In pharmaceutical research, a SAS program is expected to produce accurate and reliable results. Programs are often reused from past studies to save time and maintain consistency. During the reuse of prior programs or during subsequent updates of the data after initial programming effort, programmers often do not check the database as thoroughly as they do while initial program creation. This creates a possibility to miss the data issues.

This paper provides a simple and robust approach to avoid such problems, by collecting potential data issues from SAS programs and centrally maintaining their status. A dataset for a study is created which maintains the most up to date data problems status sent by individual programs. A macro is introduced, which can be embedded in any SAS program, to send the potential data issue and its details to the central repository. Finally, a utility is developed to review the dataset.

INTRODUCTION
Experience has shown that though the data management group does an excellent job in cleaning up the data, often statistical programmers discover data issues - not usually checked or missed by data management team - during the programming process.

If the data is not completely clean, reusing prior programs or macros for newer studies becomes problematic. Usually programmers make valid data related assumptions for proper functioning of programs in a certain study. E.g. absence of duplicate records of certain kind, presence of valid data values in each record, proper order of dates etc. When programmers use the same programs for newer studies, it is common that they skip the thorough process of verification of the programs or fail to check whether the prior assumptions hold true for the current data. This increases the likelihood for data issues to slip through the cracks.

Additionally, Statistical programmers usually begin the programming for analysis datasets and TFL (tables, figures, and listings) during the data collection period, i.e. much before the final database lock. They communicate the discovered data issues to data management via emails, spreadsheet or meetings, while the data is still being collected and queried upon. In most of the cases, the data is updated several times before the final database lock. The programmers get much more time to look at the data closely during initial programming effort rather than during the intermediate or final stages of data collection. Without a systematic approach to identify, track and resolve such data issues, these can persist in subsequent database updates.

This paper addresses the above concerns by presenting a simple robust centralized approach to keep track of the status of recent or previously discovered data issues across a team of programmers working on a study, while eliminating the need for programmers to thoroughly recheck the database for data issues discovered in past and thus saving significant time. The approach can be very useful for studies that use older programs that need minimum modifications for fulfilling current requirements.

THE SOLUTION
The proposed system consists of certain components that make the approach robust, reusable across studies and easy to add into existing SAS programs or macros. These are as follows:

1. Macro ISSEND - A flexible macro to easily send & update data issue description & data records to dataset ISSTAT
2. Dataset ISSTAT - A central repository for each study to keep track of status and record for data issues
3. Macro VIEWRPT - A SAS macro/utility to view status of data issues (with option to view either summary or details) from dataset ISSTAT

The proposed approach creates a central dataset ISSTAT, acting as a tracker of the status and details of the current set of potential data problems in the study. This dataset, discussed later in detail, consists of data records for all the issues found by programs across a study. It is populated and updated every time a study related SAS program calls the macro ISSEND to update the dataset. Any SAS program, belonging to the specific study, can report potential issues by embedding small & concise macro calls anywhere within in the programs. With the help of this dataset and
an associated report generating SAS utility VIEWRPT, the lead programmer/manager can quickly review the status for data issues periodically or every time the data/reports are updated.

The components are discussed in detail below.

**MACRO ISSEND**

Macro ISSEND is a simple macro created to send and update data issues in the central dataset ISSTAT. Insert it between any two data/proc steps, in any study related SAS program, whether it is a macro, or a table/listing/figure generating program or one used to create derived dataset. A typical call for macro %ISSEND with most common parameters looks like:

```
%ISSEND(DATALIB = pharmasug,
        PROG     = ae_sample.sas,
        DATA     = ae(keep=subjid visit aecrf aeterm aestrdt where=(aeterm=' ' and aecrf ne ' ')),
        DESCRIBE = AE records with missing AE preferred term
);
```

This macro call sends all records (if any) from AE (Adverse Event) dataset with missing AE preferred terms and non-missing CRF terms and delete any previous data records sent by program ae_sample.sas.

**MACRO PARAMETERS**

The MACRO ISSEND consists of the following parameters:-

1. **DATALIB** Specifies the reference for data library where the dataset ISSTAT is stored or needs to be created
2. **PROG** A text string to identify the source program, preferably the name of the program used to send the data records. Use entire program path if multiple programs exist with same name
3. **DATA** It specifies the dataset containing data records, which needs to be stored into ISSTAT dataset. The DATA parameter accepts dataset name with standard data set options as in a set statement of a data step. This allows users to bypass creating an additional data step/temporary dataset in several cases.
4. **DESCRIBE** Specifies description of data records sent in data parameter
5. **REPEATID** If same piece of code runs multiple times, e.g. same macro code used by multiple programs, then the parameter REPEATID can be assigned unique numerical value to differentiate multiple calls to macro ISSEND, with same description, from each other. (DEFAULT=.)
6. **CLRPREV** This parameter provides the option to clear previous records from dataset ISSTAT. It can have 4 possible values.(DEFAULT=PROG)
   a. **ALL** Clear all records from ISSTAT dataset
   b. **PROG** Clear records in ISSTAT with matching program name parameter
   c. **DESC** Clear records in ISSTAT with matching program name and description parameters
   d. **REPEATID** Clear records in ISSTAT with matching program name, description and REPEATID parameters
7. **NUMOBS** A dataset in data parameter can pass thousands of observations. This parameter limits the number of observations an individual macro call ISSEND can send. By limiting the number of observations, only initial few observations are stored in ISSTAT dataset, making the review process manageable. (DEFAULT=30)

**USING THE MACRO %ISSEND**

**CASE 1: CREATING ISSTAT DATASET**

This is the simplest part. Every time the macro ISSEND runs, it checks for the presence of dataset ISSTAT in the library reference provided in the DATALIB parameter, and if the dataset does not exist, it creates an empty one. Therefore, no separate code is required for this purpose.

**CASE 2: REPORTING A DATA PROBLEM FROM A PROGRAM**

```
%ISSEND(DATALIB = pharmasug,
        PROG     = /sasprog/drug/protocol/profile_sample1.sas,
```
In the above example, the library PHARMASUG has ISSTAT dataset. If ISSTAT dataset does not exist, then the macro would create it. The data parameter is set to 'profile (keep=pid rand treat where=(treat ne rand))', which results in reading profile dataset with observations where treatment arm not equal to randomized arm, and keeping only the relevant variables - pid, rand & treat. Remember, it is easier to review the dataset ISSTAT when the macro sends limited number of variables. It is important to understand the use of parameter CLRPREV that is set to 'PROG' here. This macro call would delete any previous data records sent from this program. One can skip the CLRPREV parameter altogether in the above example as PROG is the default value for the parameter.

CASE 3: REPORTING MULTIPLE DATA PROBLEMS FROM A PROGRAM
This scenario is very likely in clinical SAS programs, where a programmer may want to call the macro for multiple potential data issues.

```sas
%ISSEND(DATALIB = pharmasug,
PROG = prog2.sas,
DATA = data_a(keep=pid rand treat where=(treat ne rand)),
DESCRIBE = Profile records where treatment group is not equal to rand group,
CLRPREV = DESC ) ;
```

The code above demonstrates how to use macro ISSEND in such situations. CLRPREV, which has the value DESC instead of PROG, causes the macro to delete any data records with same program name and same description. This allows a program to call macro ISSEND in the program multiple times, given that the description is different every time. Passing 'DESC' as the parameter for CLRPREV, tells the macro to consider records with a combination of values in DESCRIBE and PROG parameter as a single set of records.

CASE 4: REPORTING A DATA PROBLEM MULTIPLE TIMES FROM A PROGRAM (E.G., WHEN SAME CODE IS RUN MULTIPLE TIMES FOR DIFFERENT DATA)
This is a special case when same block of code runs multiple times for a study but with different data. E.g., a macro used multiple times in a program. The example below demonstrates how to use the macro ISSEND in such scenario.

```sas
%macro effcalc(param=);
   Data eff_err(keep=pid value effparam treat rand visit);
   Set effdata;
   Where effparam = &param and value = .;
%ISSEND(DATALIB = pharmasug,
     PROG = /sasprog/drug/protocol/eff.sas,
     DATA = eff_err,
     DESCRIBE = Subjects with missing calculated efficacy parameter,
     REPEATID = param,
     CLRPREV = REPEATID
    );
run;
%mend effcalc(param='val1');
```
In the above code, assume that the macro EFFCALC runs several times with different parameter each time. In such case, the description of data problem is same, but data is different for each parameter. Here, REPEATID parameter is used to differentiate each macro call from other. At the same time, CLRPREV is set as REPEATID.

**CASE 5: DELETING ISSTAT DATASET RECORDS**

Finally, one may just want to empty the ISSTAT dataset (completely or partially). Parameter CLRPREV can be used in combination with other parameters to achieve this task. E.g. to empty the entire dataset then, use the following code.

```plaintext
%ISSEND(DATALIB = pharmasug,
        CLRPREV  = all
    );
```

To delete only partial set of records, set CLRPREV to either PROG, DESC or REPEATID and set corresponding parameter values for PROG, DESCRIBE and REPEATID.

**DATASET ISSTAT**

It is useful to understand the structure of dataset ISSTAT to review the ISSTAT dataset as an alternative to using the reporting utility VIEWRPT that is designed to summarize the information from ISSTAT. The dataset ISSTAT stores most recent set of data records sent by the macro ISSEND calls in any SAS program. If no data observation qualifies for a macro ISSEND call then no information is sent to ISSTAT for storage and previously stored records (if any) are deleted. Otherwise, ISSTAT stores one observation per data record and four additional records per macro ISSEND call to store the variable attributes - variable name, length, format and label.

Since most likely each macro ISSEND would submit a unique set of variables (e.g. the code in CASE 2 has three variables - PID, RAND & TREAT whereas the code in CASE 4 has six variables - PID, VALUE, EFFPARAM, TREAT, RAND & VISIT), storing data and variable information in dataset ISSTAT is not straightforward. To tackle this problem, the variables from data sent from ISSEND are renamed as a sequence of variables - isvar1, isvar2 and so on. These variables are of type character with length 200 each and store the data values and attributes of the original variables. The dataset has few additional variables to identify the sending program (from PROG parameter), last run date, REPEATID, description (from DESCRIBE parameter), number of observations sent, total variables reported and observation order.

Table 1 shows a sample dataset ISSTAT. It consists of data records generated by the macro ISSEND example in CASE 3 of previous section. The dataset has following variables

- **PROG, DESC & REPEATID** – These variables have the same value as set in the ISSEND macro. In the above example two set of records exist with same program name and different description.
- **RUN_DATE** – Specifies the date when the record was last updated/inserted in dataset ISSTAT
- **NUMVAR** – Specifies total number of variables stored for a set of records.
- **TOTOBS** – It is the total number of observations reported by ISSEND macro. It is helpful when the number of data records sent by ISSEND macro is greater than NUMOBS parameter.
• **ISS_ORD** – It is useful to identify variable attribute and data value observations. For variable name, variable label, length & format ISS_ORD has values -3, -2, -1 and 0. Whereas for data values, it has positive values starting from 1

• **ISVAR1, ISVAR2, ISVAR3 ...** – Variables to store data values and variable attributes.

It is important to remember that one should try to send selected variables to the dataset as it becomes harder to print and review the data issues if dataset ISSTAT consists of huge number of variables.

**MACRO VIEWRPT**

Macro VIEWRPT is a simple SAS macro utility to summarize the data records present in dataset ISSTAT. It has three parameters - DATALIB, PRINT & PROG. Similar to macro ISSEND, VIEWRPT has parameters DATALIB and PROG to specify ISSTAT directory and program name to subset the report for a specific program name. Note that PROG is an optional parameter here, i.e. if it is not used then the report would consist of information for all the data issues in the ISSTAT dataset. PRINT parameter provides user with an option to print either summary or details of the report. Consider the following examples to understand the usage and output of macro VIEWRPT.

**EXAMPLE 1: PRINTING A SUMMARY REPORT**

```
%VIEWRPT(DATALIB= pharmasug,
          PRINT  = summary
        );
```

The above code prints a summary of all the data issues present in dataset ISSTAT by setting parameter PRINT with value ‘SUMMARY’. It prints the Program name, Description, Repeat id, Last run date, Number of observations reported, Number of observations stored (limited by NUMOBS parameter), and Variable names (up to 20 are printed).

Report 1 shows a sample report based on data in table 1. Notice that number of observations reported - variable TOTOBS - in fourth column has same values as the Number of observations stored - from variable ISSORD - in fifth column. This is because number of observations reported is less than NUMOBS parameter in ISSEND macro. If the number of data observations sent to ISSEND macro is more than NUMOBS parameter, then these values would be different in the report below, i.e. when TOTOBS is greater than NUMOBS.

**REPORT 1: MACRO VIEWRPT OUTPUT**

<table>
<thead>
<tr>
<th>Program name</th>
<th>Issue Description</th>
<th>Repeat #</th>
<th>Run date</th>
<th>Number of Obs. reported</th>
<th>Number of Obs. Stored</th>
<th>Variables (printing upto first 20 variables stored in ISTAT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog2.sas</td>
<td>Profile records where treatment group is not equal to rand group</td>
<td>.</td>
<td>25JAN08</td>
<td>5</td>
<td>5</td>
<td>PID, TREAT, RAND</td>
</tr>
<tr>
<td>Prog2.sas</td>
<td>Subjects with missing calculated efficacy parameter</td>
<td>.</td>
<td>25JAN08</td>
<td>2</td>
<td>2</td>
<td>PID, TREAT, FDOSDT, LDOSDT</td>
</tr>
</tbody>
</table>

**EXAMPLE 2: PRINTING A DETAILED REPORT**

The following code prints a detailed report of all the data records present in dataset ISSTAT by setting parameter PRINT with value ‘DETAIL’. The detailed report can be restricted to data record sent by a single program, by passing program name to the parameter PROG. Figure 2 shows first page of report based on data in Table 1. The report prints data records for each program, description and repeat ID set on separate pages, with basic information for the set of observations about the data issue in the header. Since the number of variable for a set of data records can be large, printing columns for all variables can make the report unmanageable. Hence, the maximum number of variables that can be printed is restricted to 10.
To restrict the data records to a specific program, modify the macro call for VIEWRPT as shown below.

```sas
%VIEWRPT(DATALIB= pharmasug, PRINT = detail, PROG = progname);
```

**CONCLUSION**

The proposed system is a simple robust approach to catch data errors that can easily be missed when programs are reused in newer studies or when programmers create programs early during data collection phase. Existing programs can easily integrate MACRO ISSEND without any disruption.

The application of this approach is not just limited to handling the errors in raw data, but it can be used to report and manage problems generated during data manipulation, data derivation or reporting. The programming lead can easily check the ISSTAT dataset, either directly or using macro VIEWRPT, and forward the potential concerns to data management team. This way programming team can save precious time during the reports review process, use prior programs quicker with more confidence, track data problems in an orderly manner and improve quality of reports.

The appendix contains the complete code for the macros ISSEND & VIEWRPT. It can be enhanced or customized for user-specific needs. E.G., creating review reports in a different manner or structuring the ISSTAT dataset in a different manner. The macro presented here is a good starting point for anyone interested to test and start using the approach. Please feel free to use the code and contact me with any suggestions, successes or problems using the code.

**ACKNOWLEDGMENTS**

Many thanks are extended to my wife Shweta Jain, who provided exceptional coding and writing guidance for this paper, to Shridhar Patel for his critical reading of this paper and to Priyanka Jain for editing assistance of this paper.

**CONTACT INFORMATION**

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

Vineet Jain  
Statistical Programmer/Analyst  
45 W 3rd St. #1R  
South Orange, NJ 07079  
Phone: 973-821-5008  
Email: vineet7878@gmail.com

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

MACRO ISSEND
%macro ISSEND(DATALIB=, prog=, data=, describe=, REPEATID=, CLRPREV=PROG, NUMOBS=30);
  ** check if DATASET ISSTAT exists in DATALIB library, create an empty one if it does not **;
  %if %sysfunc(exist(&DATALIB..ISSTAT)) = 0 %then %do;
    %put Dataset does not exist! Creating empty dataset.;
    data &DATALIB..ISSTAT;
    format prog $50. desc $200.;
    prog = ' '; desc = ''; REPEATID = ; NUMVAR=0; TOTOBS=0; RUN_DATE = .;
    if prog = 'x' then output;
  %end;
  ** if DATASET ISSTAT exists then clear previous records depending on the value of CLRPREV parameter **;
  %else %do;
    data &DATALIB..ISSTAT;
    set &DATALIB..ISSTAT;
    %if %upcase(&CLRPREV) = ALL %then delete;
    %else %if %upcase(&CLRPREV) = PROG %then if prog = "&prog" then delete;
    %else %if %upcase(&CLRPREV) = DESC %then if prog = "&prog" and desc = "&describe" then delete;
    %else %if %upcase(&CLRPREV) = REPEATID %then if prog = "&prog" and desc = "&describe" and REPEATID = "&REPEATID"
      then delete;;
  %end;
  ** Read dataset specified by DATA parameter and check if any observations qualify **;
  data iss_inp;
  set &data;
  call symput('TOTOBS',strip(put(_n_,8.)));
  if _n_ <= input(symget('NUMOBS'),8.);
  array tmp_c{*} _character_;
  array tmp_n{*} _numeric_;
  call symput('NUMVAR',strip(put(dim(tmp_c) + dim(tmp_n),8.)));
  proc contents data=iss_inp out=iss_cont noprint;
  proc sql noprint;
    select count(*) into:tobs
    from iss_cont;
  %if &tobs ne 0 %then %do;
    ** Create data value for variable format **;
    data iss_cont;
    set iss_cont;
    if format ne ' ' then varfmt = strip(format) || strip(put(formatl,best.)) || ' ';
    ** Create temporary macro variables for variable attributes **;
    proc sql noprint;
      select name, label, length, varfmt into:var1 - :var&NUMVAR,
      :varl1 - :varlb&NUMVAR,
      :varln1 - :varln&NUMVAR,
      :varf1 - :varf&NUMVAR
      from iss_cont;
    %end;
    ** Update the ISSTAT dataset with variable attributes and data values **;
    data iss_inp;
    set iss_inp;
    NUMVAR = &NUMVAR;
  %end;
if _n_ = 1 then do;
    ISS_ORD = -3;
    %do i=1 %to &NUMVAR;
        length isvar&i $200.;
        isvar&i = "&&var&i";
    %end;
    output;
    ISS_ORD = -2;
%do i=1 %to &NUMVAR;
    isvar&i = "&&varlb&i";
%end;
output;
ISS_ORD = -1;
%do i=1 %to &NUMVAR;
    isvar&i = "&&varln&i";
%end;
output;
ISS_ORD = 0;
%do i=1 %to &NUMVAR;
    isvar&i = "&&varf&i";
%end;
output;
end;
ISS_ORD + 1;
%do i=1 %to &NUMVAR;
    %if &&varf&i ne %then %do;
        isvar&i = strip(put(&&var&i,&&varf&i));
    %end;
    %else %do;
        isvar&i = strip(&&var&i);
    %end;
%end;
output;
keep ISS_ORD &NUMVAR isvar:

** Set Description, repeatid, prog etc variables for latest set of observations **;
data &DATALIB..ISSTAT;
    format RUN_DATE date9.;
    set &DATALIB..ISSTAT iss_inp(in=b);
if b then do;
    desc = "&describe";;
    REPEATID = "&REPEATID";;
    prog = "&prog";;
    TOTOBS = &TOTOBS;;
    RUN_DATE = today();
end;

** Delete datasets not required **;
proc datasets library=work;
    delete iss_inp iss_cont/memtype=data;
quit;

run;
%mend;
MACRO VIEWRPT
%macro VIEWRPT(DATALIB=,print=summary,prog=);

%let endline =  

** If dataset ISSTAT does not exist then display corresponding message **;
%if %sysfunc(exist(&DATALIB..ISSTAT)) = 0 %then %do;
%let TOTOBS = -1;

data temp;
  x = 'Data ISSTAT - The repository for data issues - does not exist in following library';
  output;
  x = "&DATALIB";
  output;
  label x="00"x;

proc report headskip noheader nowd;
  column x;
  define x/group center;
  break after x/skip;
run;
%end;

** If ISSTAT does exist then follow with rest of code **;
%else %do;

proc sql noprint;
  select count(*) into:TOTOBS from &DATALIB..ISSTAT;

** If ISSTAT is empty then show empty report **;
%if &TOTOBS = 0 %then %do;

proc contents data=&DATALIB..ISSTAT out=temp noprint;

data temp;
  format x $30. y $20.;
  set temp;
  if _n_ = 1;
    x = "'";
    y = "'";
    output; output; output; output; output;
    x = "Dataset";
    y = "&DATALIB..ISSTAT";
    output;
    x = "'";
    y = "'";
    output;
    x = "Creation Date";
    y = put(datepart(crdate),date9.); output;
    x = "'";
    y = "'";
    output;
    x = "Last Modification Date";
    y = put(datepart(modate),date9.); output;
  label x="00"x;

proc report headskip noheader nowd;
  title 'ISSTAT Dataset is EMPTY!';
  title5 'The details of the dataset are as following';
  title6 '----------------------------------------';
  column x y;
  define x /display left;
  define y /display left;

  %end;
%end;

%if &TOTOBS > 0 %then %do;
** Subset the ISSTAT dataset if PROG parameter is provided **;
proc sort data=&DATALIB..ISSTAT out=temp;
  by prog desc repeatid iss_ord;
  %if upcase(&prog) ne %then %do;
    where upcase(prog) = upcase("&prog");
  %end;
%end;
proc sql noprint;
select count(*) into:TOTOBS2 from temp;

** if ISSTAT does not have any record for given PROG param then show appropriate message **;
%if &TOTOBS2 = 0 %then %do;
   data temp;
      x = "Note: Data ISSTAT in Library &DATALIB does not have any record with program name &prog";
      output;
      label x="00"x;
   proc report headskip noheader nowd;
      column x;
      define x/group center;
      break after x/skip;
      title1 'Macro VIEWRPT: Summary Report';
      title4 'List of Potential Data Issues';
      title5 '‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐';
   run;
%end;

** Display Summary report **;
%if &TOTOBS2 > 0 and %upcase(&print) = SUMMARY %then %do;
   data temp;
      format varname $200.;
      set temp;
      by prog desc REPEATID;
      retain varname;
      if first.REPEATID then do;
         varname = isvar1;
         array t{*} isvar:;
         do i = 2 to NUMVAR;
            if i <= 20 then varname = strip(varname) || ', ' || t(i);
         end;
      end;
      if last.REPEATID;
      NUMOBS = ISS_ORD;
   proc report data = temp nowd headskip headline ls=&ls split='\' spacing=1;
      column ('__' prog desc REPEATID RUN_DATE TOTOBS NUMOBS varname);
      define prog /group width=18 flow left 'Program name';
      define desc /group width=30 flow left 'Issue Description';
      define REPEATID /group width=10 center 'Repeat # missing';
      define RUN_DATE /display width=10 center 'Run date';
      define TOTOBS /display width=10 center 'Number of Obs. reported';
      define NUMOBS /display width=10 center 'Number of Obs. Stored';
      define varname /display width=35 left flow 'Variables (printing upto first 20 variables stored in ISTAT)';
      break after REPEATID/skip;
   title1 'Macro VIEWRPT: Summary Report';
   title4 'List of Potential Data Issues';
   footnote "&endline";
   run;
%end;

** Display Detailed Report, if requested **;
%else %if %upcase(&print) = DETAIL %then %do;
   proc sort data = temp out=temp_n nodupkey;
      by prog desc REPEATID;

proc sql noprint;
    select count(*) into:count from temp_n;

%do k = 1 %to &count;
    %let REPEATID = ;

    data temp_r;
        set temp_n;
        if _n_ = &k;
            call symput('lastvar',compress('isvar' | put(NUMVAR,best.)));
            call symput('NUMVAR',put(NUMVAR,best.));
            call symput('NUMVAR0',strip(put(NUMVAR,best.)));
            call symput('rundate',strip(put(RUN_DATE,date9.)));
            call symput('progname',strip(prog));
            call symput('desc',strip(desc));
            if REPEATID ne . then call symput('REPEATID', '#|' | strip(put(REPEATID,best.)));
            call symput('TOTOBSd',strip(put(TOTOBS,best.)));

        data temp_r;
            merge temp(in=a) temp_r(in=b);
            by prog desc REPEATID;
            if a and b;
                array t_old{&NUMVAR} isvar1 - &lastvar ;
                array t_new{&NUMVAR} $ _temporary_ ;
                if ISS_ORD = -3 then do;
                    %do j = 1 %to &NUMVAR;
                        call symput("var&j",strip(t_old(&j)));
                    %end;
                end;
                if ISS_ORD = -2 then do;
                    %do j = 1 %to &NUMVAR;
                        call symput("varlb&j",strip(t_old(&j)));
                    %end;
                end;
                if ISS_ORD = -1 then do;
                    %do j = 1 %to &NUMVAR;
                        temp = input(strip(t_old(&j)),3.);
                        if temp < 10 then temp = 10;
                        else if temp > 30 then temp = 30;
                        call symput("varln&j",put(temp,2.));
                    %end;
                end;
            end;
        %if &NUMVAR > 10 %then %let NUMVAR = 10;;

        proc report nowd data=temp_r headline headskip;
        where ISS_ORD > 0;
        column ( '__' %do i = 1 %to &NUMVAR; isvar&i %end); %do i = 1 %to &NUMVAR;
            define isvar&i /display width=&varln&i flow "&\&var&i (\&varlb&i)";
        %end;
        title "Macro VIEWRPT: Detailed Report";
        %if &prog ne %then %do;
            title2 "Report for Prog:&prog only";
        %end;
        title4 "Reporting Program: &progname &REPEATID";
        title5 "Last Run Date:&rundate ,Total Observations:&TOTOBSd ,Total variables:&NUMVAR0";
        title6 "Description: &desc";
        footnote1 "Note: Only upto 10 variables displayed due to page width limitations." ;
    %end;
%end;
footnote2 Also, report shows limited observations using NUMVAR parameter passed to macro ISSEND;