PharmaSUG 2026 - Paper PO-213

%Compare_counts: A Macro for Speeding Up the QC Process When Proc Compare Slows it Down

Michael Garside, Phastar

ABSTRACT

Quality control (QC) is a critical component of clinical trial programming and is necessary to ensure the accuracy and consistency of SDTM, ADaM, and output data sets. Current tools (such as Proc Compare) are useful and necessary. However, they can be inefficient in certain scenarios. This paper introduces the %Compare_counts SAS macro, a macro designed to directly compare the frequency of value combinations across specified variables between production and QC data sets. As a complementary QC tool, %Compare_counts provides a fast and effective approach for pinpointing discrepancies, thereby streamlining the QC process.

INTRODUCTION

The QC process is one of the most common processes that we all must adhere to in the clinical trial industry. Whether we are programming SDTMs, ADaMs, or Outputs, there is a need to verify that the completed work is correct. While there are existing methods, there are times when they may not be the quickest or most efficient way to identify or resolve discrepancies. For example, when there is a differing number of records, a Proc Compare may not be enough to tell us where the issue is. That is where the %Compare_counts macro comes in. This macro is designed to be simple and easy to use. It has a small list of parameters, a straightforward process, and easy-to-read results. Here I will explain the purpose, parameters, and internal logic of the macro, followed by some practical uses for it.

MACRO FUNCTION AND PARAMETERS

The %Compare_counts macro is a simple, yet effective macro that directly compares the count of all values of a list of variables between 2 data sets (i.e. production and QC). There are just 4 parameters for this macro:

- Prod_ds: The production data set.
- Qc_ds: The QC data set.
- Ignore_fmt: If “Y”, then all character variables in both data sets are converted to uppercase and all leading and trailing spaces are removed. Default is “N”.
- Var_list: A space-separated list of variables to compare.

Here is an example of the macro call:

```
%compare_counts(prod_ds = sdtm.lb,
                qc_ds = qc_lb,
                ignore_fmt = N,
                var_list = lbcat lbscat lbtestcd);
```

Program 1. Example Macro Call

This macro is most useful as a supplement to Proc Compare in cases where the source of the discrepancy is not immediately clear. Examples may be where there is a different number of records between the 2 data sets, many variables are differing, or there is a sorting issue between them. Let’s say you run a Proc Compare on an LB SDTM data set, and the following is the result:

Variables with Unequal Values

Variable	Type	Len	Label	Ndif	MaxDif	MissDif
LBSEQ	NUM	8	Sequence Number	6149	330	0
LBSPID	CHAR	40	Sponsor-Defined Identifier	2608		428
LBTESTCD	CHAR	8	Lab Test or Examination Short Name	5988		0
LBTEST	CHAR	40	Lab Test or Examination Name	5988		0
LBCAT	CHAR	40	Category for Lab Test	414		0
LBSCAT	CHAR	40	Subcategory for Lab Test	743		414
LBORRES	CHAR	200	Result or Finding in Original Units	6117		82
LBORRESU	CHAR	40	Original Units	4613		436
LBORNRLO	CHAR	40	Reference Range Lower Limit in Orig Unit	5532		1836
LBORNRHI	CHAR	40	Reference Range Upper Limit in Orig Unit	5605		1834
LBSTRESC	CHAR	200	Character Result/Finding in Std Format	6118		82
LBSTRESN	NUM	8	Numeric Result/Finding in Standard Units	6049	1868	460
LBSTRESU	CHAR	40	Standard Units	4408		436
LBSTNRLO	NUM	8	Reference Range Lower Limit-Std Units	5532	49980	1836
LBSTNRHI	NUM	8	Reference Range Upper Limit-Std Units	5605	99989	1834
LBSTNRC	CHAR	40	Reference Range for Char Rslt-Std Units	1582		1582
LBNRIND	CHAR	20	Reference Range Indicator	2022		1802
LBSTAT	CHAR	8	Completion Status	82		82
LBREASND	CHAR	200	Reason Test Not Done	87		82
LBNAM	CHAR	200	Vendor Name	428		428
LBLOINC	CHAR	8	LOINC Code	1143		112
LBSPEC	CHAR	200	Specimen Type	591		82
LBLOBXFL	CHAR	1	Last Observation Before Exposure Flag	678		678
LBBLFL	CHAR	1	Baseline Flag	678		678
LBFAST	CHAR	1	Fasting Status	576		428
VISITNUM	NUM	8	Visit Number	2553	1800	52
VISIT	CHAR	40	Visit Name	2553		52
VISITDY	NUM	8	Planned Study Day of Visit	2505	176	234
EPOCH	CHAR	40	Epoch	931		64
LBDTC	CHAR	19	Date/Time of Specimen Collection	2588		60
LBDY	NUM	8	Study Day of Specimen Collection	2519	175	42

Figure 1. LB Proc Compare Results

Based on the above results, it is not immediately clear what the issue is. Generally, the next step would be to scroll down to the comparison between specific variables. However, that could prove to be a daunting task. We will later discuss how to apply the %Compare_counts macro in scenarios such as this.

THE MACRO PROCESS

The macro follows the following steps:

1. If the &ignore_fmt variable is set to “Y”, then new data sets are created for both production and QC where all character variables are capitalized and all leading and trailing spaces are removed. These new data sets replace the original input data sets. This allows a more direct comparison of the intended values themselves rather than the format of the values.

```
%if "&ignore_fmt" = "Y" %then %do;
  data prod_ds1;
    set &prod_ds (keep = &var_list);
    array char_vars {*} _character_;
    do i = 1 to dim(char_vars);
      char_vars{i} = strip(upcase(char_vars{i}));
    end;
  run;

  data qc_ds1;
    set &qc_ds (keep = &var_list);
    array char_vars {*} _character_;
    do i = 1 to dim(char_vars);
      char_vars{i} = strip(upcase(char_vars{i}));
    end;
  run;

  %let prod_ds = prod_ds1;
  %let qc_ds = qc_ds1;
%end;
```

Program 2. Ignoring the Format

2. The &var_list macro variable is then formatted in such a way that it can be put directly into a Proc Freq. This new macro variable is then used to obtain the frequencies in the production and QC data

sets. For example, if you input “lbcat lbscat lbtestcd” for &var_list, it then becomes “lbcat*lbscat*lbtestcd”, which is able to be directly used in the Proc Freq.

```
%let var_list_freq = %sysfunc(tranwrd(&var_list, %str( ), *));

proc freq data = &prod_ds noprint;
  tables &var_list_freq / out = prod_freq (drop = percent);
run;

data prod_freq1 (drop = count);
  set prod_freq;
  count_prod = count;
run;

proc freq data = &qc_ds noprint;
  tables &var_list_freq / out = qc_freq (drop = percent);
run;

data qc_freq1 (drop = count);
  set qc_freq;
  count_qc = count;
run;
```

Program 3. Obtaining the Frequencies

3. The production and QC data sets are then merged together by the variables contained in the &var_list macro variable to create a comparison data set. Any missing frequency in either data set is set to 0. The total difference in frequency for each combination of values is also calculated. All other work data sets produced within the macro are then deleted. If there are no differences, there will be no output data set, and instead a message is output to the log stating so.

```
data comp;
  merge prod_freq1
        qc_freq1;
  by &var_list;
  if count_prod = . then count_prod = 0;
  if count_qc = . then count_qc = 0;
  if count_prod ne count_qc;
  diff = count_qc - count_prod;
run;

proc datasets library = work nolist;
  %if "&ignore_fmt" = "Y" %then %do; delete prod_freq prod_freq1 qc_freq qc_freq1
                                           prod_ds1 qc_ds1; %end;
  %else %do; delete prod_freq prod_freq1 qc_freq qc_freq1; %end;
run;

proc sql noprint;
  select count (*)
  into :num_obs
  from comp;
quit;

%if &num_obs = 0 %then %do;
  proc datasets library = work nolist;
    delete comp;
  run;
  %put All values appear the same number of times in both datasets for the given
       variables.
%end;
```

Program 4. Comparing the Counts

APPLICATION

Two examples of macro implementation are shown below.

EXAMPLE 1: SUPPLEMENTAL SDTM

One example of where the %Compare_counts macro is particularly useful is with supplemental SDTM domains, where discrepancies are frequently due to missing values of QNAM. Let us suppose that as a QC programmer, you are running a Proc Compare on a SUPPAE data set, and the following is the result:

Dataset	Created	Modified	NVar	NObs
WORK.PROD_SUPPAE	22SEP25:18:03:58	22SEP25:18:03:58	10	5961
WORK.QC_SUPPAE	22SEP25:18:03:58	22SEP25:18:03:58	10	4664

Figure 2. SUPPAE Compare

Your first instinct may be to try to figure out the issue by looking at the variable mismatches provided by the compare. However, given the difference in the number of records, the better option is to instead run the %Compare_counts macro using the following call:

```
%compare_counts(prod_ds = prod_suppae,
                qc_ds = qc_suppae,
                var_list = qnam);
```

Program 5. Example Macro Call 1

The result is the following data set:

	QNAM	count_prod	count_qc	diff
1	AETRTEM	1300	0	-1300
2	DXTSTS2	0	2	2
3	DXTSTS3	0	1	1

Figure 3. SUPPAE Count Compare

From this you can easily conclude that the QC data set is missing AETRTEM, but the production data set is missing DXTSTS2 and DXTSTS3. This difference in number of records between the 2 data sets is completely accounted for.

EXAMPLE 2: ADAM

In this example, the usefulness of the “ignore_fmt” parameter is shown using an ADLB data set. Suppose that a Proc Compare is run, and this is the initial result:

Obs	Parameter Category 1 Base Value PARCAT1	Compare Value PARCAT1
1	Hematology	HEMATOLOGY
2	Hematology	HEMATOLOGY
3	Hematology	HEMATOLOGY
4	Hematology	HEMATOLOGY
5	Hematology	HEMATOLOGY
6	Hematology	HEMATOLOGY
7	Hematology	HEMATOLOGY
8	Hematology	HEMATOLOGY
9	Hematology	HEMATOLOGY
10	Hematology	HEMATOLOGY
11	Hematology	HEMATOLOGY
12	Hematology	HEMATOLOGY
13	Hematology	HEMATOLOGY
14	Hematology	HEMATOLOGY
15	Hematology	HEMATOLOGY
16	Hematology	HEMATOLOGY
17	Hematology	IRON PANEL
18	Hematology	IRON PANEL
19	Hematology	IRON PANEL
20	Hematology	IRON PANEL

Figure 4. ADLB Compare

While you can tell that both sides really have many of the same values, the capitalization issue is masking more of the true discrepancies. Rather than informing the production programmer of the capitalization error and waiting for it to be remedied from the other end, we want to continue the QC process and identify all possible issues before notifying the production programmer. You first run %Compare_counts with the “ignore_fmt = Y” option on the PARCAT1 and PARAMCD variables to get a better feel of the discrepancies. You run the following macro call:

```
%compare_counts(prod_ds = adam.adlb,
                qc_ds = qc_adlb,
                ignore_fmt = Y,
                var_list = parcat1 paramcd);
```

Program 5. Example Macro Call 2

	PARCAT1	PARAMCD	count_prod	count_qc
1	HEMATOLOGY	FERRITIN	2023	0
2	HEMATOLOGY	IBCT	1621	0
3	HEMATOLOGY	IRON	1890	0
4	IRON PANEL	FERRITIN	0	2023
5	IRON PANEL	IBCT	0	1621
6	IRON PANEL	IRON	0	1890

Figure 5. ADLB Count Compare

The capitalization error for PARCAT1 is ignored, and from this you can easily tell that the FERRITIN, IBCT, and IRON parameters have PARCAT1 = “HEMATOLOGY” for the production data set but have PARCAT1 = “IRON PANEL” for the QC data set.

CONCLUSION

Though Proc Compare is a powerful and necessary tool, it may sometimes be difficult to determine where the root issues are without some extra work. The %Compare_counts macro takes care of that extra work, creating a quick and painless way to pinpoint the root of the discrepancies between the two data sets.

CONTACT INFORMATION

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

Michael Garside
Phastar
Michael.garside@phastar.com