Automated Validation of Complex Clinical Trials Made Easy

Richann Watson, Experis; Josh Horstman, Nested Loop Consulting

ABSTRACT

Validation of analysis datasets and statistical outputs (tables, listings, and figures) for clinical trials is frequently performed by double programming. Part of the validation process involves comparing the results of the two programming efforts. COMPARE procedure output must be carefully reviewed for various problems, some of which can be fairly subtle. In addition, the program logs must be scanned for various errors, warnings, notes, and other information that might render the results suspect. All of this must be performed repeatedly each time the data is refreshed or a specification is changed. In this paper, we describe a complete, end-to-end, automated approach to the entire process that can improve both efficiency and effectiveness.

INTRODUCTION

xxxxxxxxxxxxxxxxxxxxx. Production (PRD) programmer is the individual that is responsible for producing the final output for the deliverable. Quality Control / Verification (VER) programmer is the individual that is responsible for verifying that the results of the final output and if the final output is a table, listing or figure (TLF) that the output meets cosmetic specifications (i.e., titles, footnotes, indentations, etc. are per the requirements). xxxxxxxxxxxxxxxxx

PREPARING PRODUCTION OUTPUT

In order to automate the validation process, the PRD programmer needs to produce a permanent data set that can be used by the VER programmer. If the final output is a SAS data set, then the SAS data set itself is the permanent data set that will be used for verification. However, if the final output is a TLF, then the permanent data set can be produced in one of two ways.

1. Format the data per the TLF specifications and save to a permanent SAS data set. This data set will be used as input into the procedure that will render the final output. There would be no additional formatting or modification to this final data set during the production of the final output. It would be used as is.
2. If using the REPORT procedure, then with the use of OUT= the PRD programmer can produce a permanent data set. The OUT = option will produce a data set with a record for every row that is created for the final output. This includes any blank lines and summary lines. In addition, the OUT = option will produce a variable for every column of the report. When producing these variables SAS will utilize the column name if possible otherwise it will name the variables but position in the output (i.e., _C1_, _C2_), thus it is important to give the variables unique and meaningful names. Furthermore, the OUT = will produce an additional variable _BREAK_ that identifies what type of row was generated.
    a. _BREAK_ equal null indicates a detail row (i.e., a row from the input data set)
    b. _BREAK_ not equal null can be based on two different factors.
        i. If the record is created from a summary, then the value of _BREAK_ is the name of the variable that is used to generate the summary line.
        ii. If the record is created from a COMPUTE BEFORE/AFTER block, then _BREAK_ is the name of the variable used to determine the execution of the compute block. For example, if COMPUTE BEFORE _PAGE_, then _BREAK = '_PAGE_'.

    Since the VER programmer will more than likely not use PROC REPORT to produce their output they would either need to manually code these _BREAK_ records into their data set or the PRD programmer can exclude them from the final output but using a subsetting where clause. See sample code below.

SAMPLE CODE

```
libname PRD "directory where SAS data set from PROC REPORT is stored";

proc report data=all split='~' nowindows missing
            out=PRD.RPTDSN (where=(_BREAK_ ne 'ord'));
  column ord population treatment sort status cnt pct;

  define ord        / noprint order order=data;
  define population / order  'Population';
  define treatment  / order  'Treatment';
  define sort       / noprint order order=data;
  define status     / display order=data 'Status~Reason for Exclusion';
  define cnt        / display 'n';
  define pct        / display '%';

  break after ord / page;
run;
```

DOUBLE PROGRAMMING FOR QC

Independent programmer re-produces the same output. Regardless if the final delivery output is a data set, table, listing or figure; the production output that will be QC'd is a data set. If the PRD programmer created a permanent data set of the data used to produce the final output, then ideally the VER programmer will create a VER data set with the same variables that was created in production.

MANUAL REVIEW OF PROC COMPARE OUTPUT

As noted previously, the creation of a comparison data set to automate the validation process does have some drawbacks, which leads us back to the manual process of reviewing the PROC COMPARE output. With this manual process there are certain things that would need to be kept in mind.

1. Missing Variables
2. Missing Observations
3. Conflicting Types
4. Mismatched ID Variables

Need more details here on this section

For more details regarding these potential pitfalls, refer to "Don't Get Blindsided by PROC COMPARE" [1]

CREATING A COMPARISON DATA SET

After the VER data set is produced, the compare procedure can be used to automate a process that will allow for 100% validation of the ***content*** of the output. By producing a data set of the comparison, the validation of the output can be automated for each potential re-run. In order for this approach to work the following options needed to be utilized:

- **OUT =**: name of the output data set to be created
- **OUTNOEQUAL**: prevents records from being created if the values in the pair of matching observations between BASE and COMP are considered to be equal
- **OUTBASE**: creates a record for each observation in the BASE= data set
- **OUTCOMP**: creates a record for each observation in the COMPARE= data set
- **OUTDIF**: creates a record for each pair of matching observations between BASE and COMP

With the use of these options within PROC COMPARE, a permanent comparison (CMP) data set can be used for checking to see if there were any discrepancies. See sample code below.

```
libname PRD "directory where the production data set is stored";
```

```
libname VER "directory where the validation data set is stored";
libname CMP "directory where the compare data set is stored";

proc compare base=PRD.RPTDSN (drop=_BREAK_)
             compare=VER.V_RPTDSN listall
              out=CMP.RPTDSN outnoequal outbase outcomp outdif;
  id ord sort;
run;
```

If there are no discrepancies between PRD and VER data sets, then the CMP data set will have zero observations. If there are discrepancies, then the CMP data set will contain records of the following type (i.e., _TYPE_ =)

- **BASE**: either record did not have a matching pair in COMPARE = data set or at least one value between BASE = and COMPARE = was unequal
- **COMPARE**: either record did not have a matching pair in BASE = data set or at least one value between BASE = and COMPARE = was unequal
- **DIF**: shows the difference between the BASE record and the COMPARE record
    - If the variable is a character variable, the discrepancy will be noted with an 'X'
    - If the variable is a numeric variable, the discrepancy will be the difference of BASE and COMPARE

Summarizing Results of Comparison Data Set

Once all the VER programmers have completed the programming, the programmer can open the CMP data set to determine if there were any observations and if so then there was a discrepancy between the PRD and VER. However, this can be time consuming if there are numerous data sets to open. An alternative is to run a secondary program that will look at all the CMP data sets to see if they had zero observations or greater than zero observations. The program would then produce a report that summarizes the results. For more details on this approach refer to "Automated or Manual Validation: Which One is for You?"[2]

Limitations with Comparison Data Set

Although it is ideal to want to automate the validation process, the creation of the CMP data set does have its limitations. This approach does require upfront communication between PRD and VER programmers to ensure that the data set structures match (i.e., they need to use the same variable names, lengths and formats). In addition, this process does not produce a record when there are variables in one data set but not the other. Nor does it produce a record if the variable attributes between PRD and VER are different.

PARSING THE PROC COMPARE OUTPUT

Describe the following:

- *What the program is looking for*
- *What the input parameters are*
- *Explain how the macro works*
- *Describe the report*

CHECKING THE LOGS

After all the outputs have passed verification, does not mean the job is done. Part of the validation process is to ensure that the both the PRD and VER programs have executed successfully. One approach to open all the log files and manually scan the logs for various error messages. This can be very time consuming and is prone to human error and can be easily overlooked especially during "crunch" times and you just need to get the outputs delivered.

It is not wise to deliver outputs, without first looking at the logs. On alternative to opening each log file for both the production side and the verification side is to allow SAS to parse through each log and look for the unwanted log messages and then provide a report of the findings.

For more details on the process that allows SAS to check the logs for you, refer to “Check Please: An Automated Approach to Log Checking” [3]

CONCLUSION

xxxxxxxxxxxxx Validation the necessary evil of a programmer’s life. It can be difficult or it can be easy. With the utilization of the CHECKCMPS and CHECKLOGS macros, it can be made significantly easier. xxxxxxxxxxxxxx

REFERENCES

[1] Horstman, J and Muller, R. “Don’t Get Blindsided by PROC COMPARE”
http://support.sas.com/resources/papers/proceedings14/1615-2014.pdf

[2] Watson, R and Johnson, P. “Automated or Manual Validation: Which One is for You?”
http://www.lexjansen.com/pharmasug/2011/AD/PharmaSUG-2011-AD01.pdf

[3] Watson, R. “Check Please: An Automated Approach to Log Checking”
http://support.sas.com/resources/papers/proceedings17/1173-2017.pdf

CONTACT INFORMATION

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

Richann Watson
Experis
513-843-4081
richann.watson@experis.com

Joshua M. Horstman
Nested Loop Consulting
317-815-5899
josh@nestedloopconsulting.com