Paper 081-2019

Validate the validated: A python script for study leads to review clinical outputs

Varaprasad Ilapogu, Ephicacy Consultancy Group; Janet J. Li, Pfizer Inc.; Masaki Mihaila, Pfizer Inc.; Ernesto Gutierrez, Pfizer Inc

ABSTRACT

Datasets and statistical outputs produced by clinical SAS® programming teams in the pharmaceutical industry are often validated individually by parallel programming prior to submission to other teams (e.g. Statistics and Medical Writing). This process leaves out the cross-checking of outputs against other outputs that may have similar information presented. For example, the population size of each treatment group in a study is presented in many of the outputs, yet there is often not a programmatic process in place to check whether the population sizes match across the different statistical outputs. We have developed a python script that addresses checking information across statistical outputs. The script extracts commonalities from each statistical output (e.g. individual tables and figures in Rich Text Format (RTF)) and presents the relevant information in a single, easily accessible document (e.g. an excel spreadsheet) to help facilitate the cross-checking of this information. We hope that this additional process can help enhance the quality and increase the efficiency of the study package review process.

INTRODUCTION

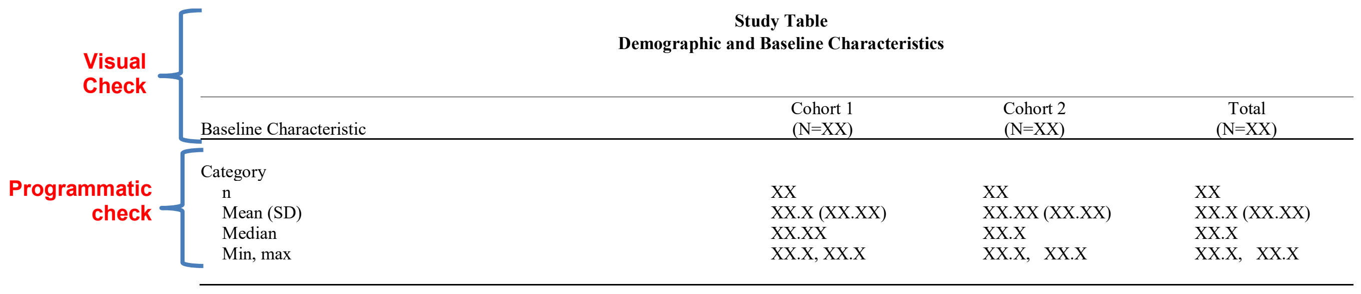
The validation of datasets and statistical outputs produced by clinical SAS® programming teams in the pharmaceutical industry is important in ensuring the accuracy of the submission packet. One common method of validation used is *independent programming*, where two statistical programmers, a production programmer and a validation programmer, independently generate the same output. If their versions match, then the output is considered to have passed validation. This process is repeated for all outputs in the study packet and sent to other teams (e.g. Statistics, Medical Writing, Clinical) for further review.

In this paper, we focus on the validation of tables that contain statistical summaries of the clinical trial data. Most tables that are generated for clinical trial reporting analyze a set of continuous and/or categorical variables across treatment groups. Table generation can be summarized as a three-step process:

1) Sub-setting the population into specific population and treatment sub-groups.
2) Calculating appropriate statistics for the tables.
3) Presenting the information in RTF or PDF format usually using PROC REPORT

To facilitate the validation of the table, the production programmer outputs the final dataset used in the PROC REPORT to a permanent library on the server. The validation programmer carries out steps 1 and 2 of the table generation process and uses PROC COMPARE to detect if any differences exist. Most tables commonly present simple frequencies of events and the percentages are calculated using the underlying analysis population as the denominator. These percentages are usually rounded off to one or two decimal places or as needed.

The double programming method is aimed mostly at checking the body of the output, such as the frequency or proportion of an event presented in a table. In addition to this, programmers perform a visual check of the output and look for spelling errors, alignment issues, population denominator inconsistencies, etc. See **Figure 1** as an example to understand the parts of the table that are validated programmatically and the parts that are checked visually in clinical trials. Validation of the output through visual checking can be subjective, whereas programmatic checking of the entirety of the table is more accurate.



Study Table
Demographic and Baseline Characteristics

Baseline Characteristic	Cohort 1 (N=XX)	Cohort 2 (N=XX)	Total (N=XX)
Category			
n	XX	XX	XX
Mean (SD)	XX.X (XX.XX)	XX.XX (XX.XX)	XX.X (XX.XX)
Median	XX.XX	XX.X	XX.X
Min, max	XX.X, XX.X	XX.X, XX.X	XX.X, XX.X

Figure 1. Visual schematic of the validation of a statistical programming table output

The sample PROC REPORT code provided below demonstrates the need for programmatic validation of the parts of a table that are usually visually checked. While the RTF or PDF output is generated by the PROC REPORT procedure below, only the values (frequencies and percentages) in the `final` SAS® dataset get validated programmatically. This dataset generally does not include the population denominator values. Rather, these are defined in the PROC REPORT DEFINE statement (`N=&C1.`, `N=&C2.`, `N=&C3.`). The denominators are indirectly validated since the percentage values in the `final` dataset are based on the denominators. In a scenario where the production and validation denominators differ by a small value, the percentage calculations in the two datasets may still match if the denominator is a large enough number and if the percentages are rounded. This may give the false impression that the primary or production table has passed validation.

```
proc report data = final split = '~' missing nowindows;
    by pageno;

    column (pageno order text value1 value2 value3);

    define pageno   / order order=internal noprint;
    define order    / order order=internal noprint;
    define text     / display "Generic Name" ;
    define value1   / display "Cohort 1 ~ (N=&C1.)";
    define value2   / display "Cohort 2 ~ (N=&C2.)";
    define value3   / display "Total ~ (N=&C3.)”;
run;
```

Another instance in which the general validation process of tables may fall short is that individual programmers do not cross check one output against a different output routinely. This can be due to a variety of factors, such as time or a lack a formal set of standard operating procedures in place to do so. The assumption made here is that if an individual table passes validation, then it is most likely to be right. However, due to the various criteria and differences in sub-setting the required population, table denominators sometimes end up differing across the study.

The reason why comparing two different outputs is important in clinical trials is because different outputs have some underlying commonalities such as analysis population (Safety, Intent to Treat etc.), a specific overall statistic (Number of subjects who had at least 1 Adverse Event), Number of subjects in each treatment group etc. Some of these numbers are not presented in the body of the table and instead presented in the headers and checked visually if at all.

We have developed a python script that addresses these gaps that may exist in checking information across statistical outputs. The script extracts commonalities from each statistical output (e.g. individual tables and figures in RTF form) and presents the relevant information in a single, easily accessible document (e.g. an excel spreadsheet) to help facilitate the cross-checking of this information.

PYTHON SCRIPT

Python is a high-level, general-purpose programming language. The following is involved for the Python script to run:

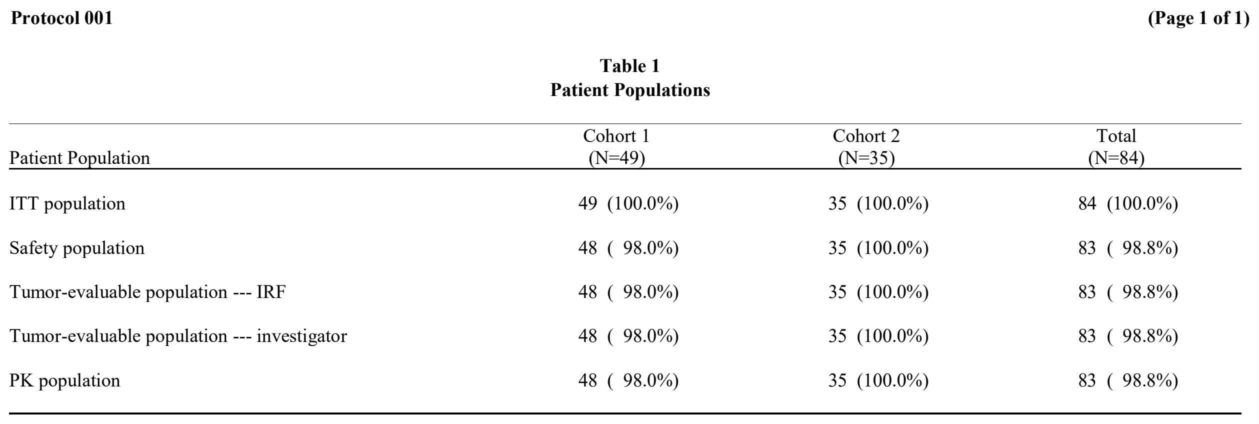
1) *Python 3* should be installed on your PC.
2) *Anaconda 3* should be installed, since most of the utilized libraries for the script are included as site-packages.
3) Obtain *StyleFrame* package by calling "pip install StyleFrame". Pip installs all Python package dependencies required.
4) Obtain *pyth3* library by calling "pip install pyth3". *Pyth* is a python text markup and conversion library.

The python script can be used once the requirements are met. As with any other programming tool, the script needs input, in this case, the location where the tables are stored. Upon execution of the script, Python Graphical User Interface (GUI) app will open and ask the user to point to the location by navigating the Windows File Explorer. The script converts each RTF table in the source location to plaintext string using XHTMLWriter from *pyth3*'s "pyth.plugins.xhtml.writer". The relevant information (['Protocol name','Table number','Total pages','Table title','Header','Date created']) is stripped from the plaintext strings and then outputted to an external Excel file using *Pandas DataFrame* and *StyleFrame* utilities.

CASE STUDY

For this paper, we examine five safety tables from a single case study to highlight gaps that may exist in checking information across statistical outputs and the utility of the Python script to address those gaps. While the script is capable of identifying and addressing multiple issues, we focus on the inconsistencies that can arise from analysis population denominators that are presented in the headers of the tables. The population denominators are not usually compared programmatically across tables.

TABLE 1



Protocol 001 **(Page 1 of 1)**

Table 1
Patient Populations

Patient Population	Cohort 1 (N=49)	Cohort 2 (N=35)	Total (N=84)
ITT population	49 (100.0%)	35 (100.0%)	84 (100.0%)
Safety population	48 (98.0%)	35 (100.0%)	83 (98.8%)
Tumor-evaluable population --- IRF	48 (98.0%)	35 (100.0%)	83 (98.8%)
Tumor-evaluable population --- investigator	48 (98.0%)	35 (100.0%)	83 (98.8%)
PK population	48 (98.0%)	35 (100.0%)	83 (98.8%)

Table 1 is a population table which presents the numbers of subjects within commonly used patient population flags, such as intent-to-treat (ITT), safety, etc. The values for each of the population groups in this table should match the values of the population denominator (*N*) headers in their respective population group tables. For instance, the number of subjects in Cohort 1 for ITT population (Column 1, Row 1) is 49, which should be the number of subjects (*N* = 49) for Cohort 1 for all ITT tables for this study.

TABLE 2

Protocol 001 **(Page 1 of 3)**

Table 2
Demographic and Baseline Characteristics
(ITT Population)

Baseline Characteristic	Cohort 1 (N=49)	Cohort 2 (N=34)	Total (N=83)
Age			
n	49	34	83
Mean (SD)	50.1 (11.48)	53.4 (11.05)	51.5 (11.35)
Median	50.0	52.0	50.0
Min, max	31.0, 74.0	33.0, 75.0	31.0, 75.0

This study's baseline characteristics table for ITT population is presented in **Table 2**. A careful examination of this table headers reveals that the population denominator for Cohort 2 (*N* = 34) is not consistent with the number of subjects in Cohort 2 for ITT population in **Table 1**. One possible, yet commonly occurring, scenario for this inconsistency is that the programmer could have subset the dataset by non-missing age values, instead of calculating the population denominators independently of age values.

TABLE 3

Protocol 001 **(Page 1 of 1)**

Table 3
Dose Modifications Due to Adverse Events
(Safey Population)

	Cohort 1 (N=49)	Cohort 2 (N=35)	Total (N=84)
Patients with no dose reduction due to adverse event	22 (45.8%)	12 (34.3%)	34 (41.0%)
Patients with at least 1 dose reduction due to adverse event	26 (54.2%)	23 (65.7%)	49 (59.0%)

Table 3 is a dose modification table. The spelling of 'Safey' in the table title is incorrect; it should be spelled 'Safety'. Additionally, the population denominator for Cohort 1 (*N* = 49) is inconsistent with the corresponding number in **Table 1**.

TABLE 4

Protocol 001 **(Page 1 of 1)**

Table 4
Study Drug-Related Treatment-Emergent Adverse Events by System Organ Class, and Preferred Term
(Safety Population)

System Organ Class Preferred Term	Cohort 1 (N=50)	Cohort 2 (N=35)	Total (N=85)
Number of patients with at least 1 study drug-related TEAE	46 (95.8%)	33 (94.3%)	79 (95.2%)

Table 4 is a safety population adverse event (AE) table. As with **Table 3**, the population denominator for Cohort 1 (*N* = 50) is inconsistent with the corresponding number in **Table 1.**

While we only focused on the utility of the python script for inconsistencies seen in the population denominators presented in the headers of the table, the python script is also capable of addressing inconsistencies in the values presented in the body of the table. For instance, the first row in this table, "Number of patients with at least 1 study drug-related TEAE", usually occurs in multiple AE tables. The python script can assess the values in this row, as the values should be consistent across all tables that report this row.

TABLE 5

Protocol 001 **(Page 1 of 1)**

Table 5
Treatment-Emergent Adverse Events Leading to Study Drug Dose Reduction by System Organ Class and Preferred Term
(Safety Population)

System Organ Class Preferred Term	Cohort 1 (N=48)	Cohort 2 (N=35)	Total (N=83)
Number of patients with at least 1 study drug-related TEAE	45 (93.8%)	33 (94.3%)	78 (94.0%)

Notice that the first row for **Table 5** is the same as that in Table 4.

EXCEL OUTPUT GENERATED BY THE PYTHON SCRIPT

The python script output for the five tables above is presented below. The output generated by the script extracts and presents the top portion of a given table in the different columns. Spelling errors or table numbering errors can be seen in the 'Table Title' column. Inconsistencies in the population denominators can be observed in the 'Header' column as it is easy to compare the numbers (N) as they are aligned one below the other.

Depending on the computer environment, the generation of the output by the python script takes a few seconds to a few minutes, which is faster and more efficient than the alternative of opening multiple outputs at the same time and visually checking for inconsistencies among the elements mentioned above.

The python script can also be enhanced to present only when inconsistencies occur. Additionally, it can extract elements from the body of a table, as mentioned in descriptions of Tables 4 and 5.

Project name	Table number	Total pages	Table title	Header
Protocol 001	Table 1	1	Table 1 Patient Populations	Cohort 1 (N=49) Cohort 2 (N=35) Total (N=84)
Protocol 001	Table 2	1	Table 2 Demographic and Baseline Characteristics (ITT Population)	Cohort 1 (N=49) Cohort 2 (N=34) Total (N=83)
Protocol 001	Table 3	1	Table 3 Dose Modifications Due to Adverse Events (Safey Population)	Cohort 1 (N=49) Cohort 2 (N=35) Total (N=84)
Protocol 001	Table 4	1	Table 4 Study Drug-Related Treatment-Emergent Adverse Events by System Organ Class, and Preferred Term (Safety Population)	Cohort 1 (N=50) Cohort 2 (N=35) Total (N=85)
Protocol 001	Table 5	2	Table 5 Treatment-Emergent Adverse Events Leading to Study Drug Dose Reduction by System Organ Class and Preferred Term (Safety Population)	Cohort 1 (N=48) Cohort 2 (N=35) Total (N=83)

Output 1. Excel output generated by the python script

CONCLUSION

The validation of datasets and statistical outputs produced by clinical SAS® programming teams in the pharmaceutical industry is important in ensuring the accuracy of the submission packet. In this paper, we focus on the current practice of table validation. While the body of a table is validated programmatically, there are some parts of the table that are validated indirectly or through visual checking. Additionally, there is no formal programmatic procedure in place to check for accuracy of information that is similar across tables. The python script we have created provides a more efficient and accurate programmatic solution for the aforementioned gaps in the table validation process. We hope that this additional step can help enhance the quality and increase the efficiency of the study package review process.

REFERENCES

Shilling, Brian C. 2010. *The 5 Most Important Clinical SAS Programming Validation Steps*. Wayne, PA. NESUG 2010.

Shostak, Jack. 2014. *SAS® Programming in the Pharmaceutical Industry, Second Edition*. Cary, NC: SAS Institute, Inc.

CONTACT INFORMATION

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

Varaprasad Ilapogu
Ephicacy Consultancy Group
prasad.ilapogu@pfizer.com

Janet Li
Pfizer Inc.
janet.li@pfizer.com