PharmaSUG 2013 - Paper IB09

Is Your Data Set Really Validated: Beware of "Blind-Fold" Validation

Neha Mohan, inVentiv Health Clinical, Mumbai, India
Gayatri Karkera, inVentiv Health Clinical, Mumbai, India

ABSTRACT

The aim of validation programming in clinical trial is to produce documented evidence that the trial data and reports meet the quality attributes as per clinical requirements. One of the basic and minimal necessities during validation is to match the numbers in both of the outputs, original as well as validation. In order to achieve this, some validation programmers often inadvertently perform what we term as "blind-fold" validation in this paper. We call it "blind-fold" validation because many a times the programmers have a tendency to,

- Follow the specifications "as is" - down to the punctuation
- Modify the validation output to match the original output incase any differences are noticed

This paper illustrates with simple examples of data sets validation from clinical trials how these practices can inadvertently lead to erroneous outputs passing validation. It then offers logical and meaningful validation approaches that hold valid beyond datasets as well; to help improve quality and reliability of the data using the "get it right the first time" principle.

WHAT IS EXPECTED IN VALIDATION?

Validation is a very crucial step in ensuring quality of the trial data and reports. It involves producing documented evidence which provides a high degree of assurance that a specific trial data or report will consistently meet its pre-determined specifications and quality attributes.

In clinical trials, the specifications for data sets, tables, listings and figures are usually created by the biostatistician, or by the lead programmer in consultation with the biostatistician. Based on these specifications, the original programmer writes the code to generate the outputs. As part of programming, the original programmer performs basic testing and releases the output for validation. An independent validation is then performed by a second programmer using validation procedures such as

- Independent programming
- Reviewing results (partial or complete)
- Visually verifying code

In case of data set validation, independent programming is usually performed to generate a data set identical to the original but independent of the original program. This validation data set is then compared (using PROC COMPARE) with the original/production data set.

COMMON TENDENCIES LEADING TO ERROR

When a data set is released for validation, many a times the validation programmer tends to presume that the specifications provided meet the requirements of the biostatisticians, and need to be followed as is. In addition to this, if there are any differences in the compare procedure, the validation programmer often tends to modify the validation data set to match the original with the aim of getting no unequal values in the PROC COMPARE output.

In order to illustrate the drawbacks of this approach, let us look at some cases involving validation of data sets from clinical trials

CASE 1: VALIDATION PROGRAM BEING MODIFIED TO MATCH THE ORIGINAL DATA SET

Suppose the Statistical Analysis Plan (SAP), defines classification of potential DILI case criteria as per Display 1.

A classification of potential Drug-Induced Liver Injury (DILI) cases has been established using the following critieria:

- *Total Bilirubin >2x ULN AND*
- *Direct Bilirubin/Indirect Bilirubin >1 AND*
- *ALT 2X baseline with a minimum increase of 100 IU/l*

Display 1: DILI case criteria

To facilitate the classification as per the criteria 'ALT 2X baseline with a minimum increase of 100 IU/l', the data set specifications are set up to define a variable ALT2BAS as listed in Table 1:

Variable name	Variable Label	Variable Derivation
ALT2BAS	ALT increased by 2 x Baseline	If any of the ALT lab test values is 2 times the baseline value then flag such records. The ALT values should be while on treatment.

Table 1: Variable ALT2BAS definition from specifications

The validation programmer interprets the aforementioned derivation as "flag any on treatment record that has a 2-fold increase in ALT results". In other words, if any of the on-treatment lab results is **at least** twice the baseline value, then flag such records.

With the aforementioned understanding, while comparing the validation dataset with the original dataset using Proc Compare, some differences were observed. On investigating, the validation programmer notices that the original dataset derives the variable to flag records where the on-treatment lab result value is **exactly** twice the baseline value. This leads the validation programmer to believe that he/she has misinterpreted the specs and the SAP. With this assumption, the validation code is changed to give a perfect match of values in the PROC COMPARE and in turn the data set is believed to pass validation.

When statistical analysis is performed on the corresponding endpoints, it is then noticed that the values appear inappropriate, which leads to the identification of the underlying cause in the variable derivation. Here the correct derivation should have rather been "at least twice" instead of "exactly twice".

In the first place the validation programmer interpreted the derivation to be "at least" increase. However, as the SAP does not specifically mention this, he/she assumed that the original programmer must have been right in their approach as the Stats team must have been consulted while setting up the original program. With so many "must haves", these assumptions lead to an incorrect data set passing validation.

In this scenario, the correct approach should have been for the validation programmer to check his/her understanding with the Stats team or the lead programmer in the first place, rather than modifying the code to match the original data set. This would have enabled them to get things right the first time at the dataset validation stage itself, instead of being noticed at the statistical analysis stage.

CASE 2: SPECIFICATIONS BEING FOLLOWED 'AS IS' - DOWN TO THE PUNCTUATION

2A - Leading to computational errors

Following is a "Percent duration" computation specified in the specification document:
*PERDUR = DUR*100/STOPDT – STARTDT*

For instance, if we are looking at variable values as in Table 2 below:

Start date (STARTDT)	Start date formatted	Stop date (STOPDT)	Stop date formatted	Event duration (DUR)
18775	28MAY2011	19004	12JAN2012	20

Table 2: Input variables for percent duration computation

If the aforementioned formula from the specifications is used in the program as is, SAS® will process it as follows:

PERDUR = DUR * **(**100 / STOPDT**)** – STARTDT = 20*(100/19004) – 18775= –18774.89476

Here we are looking at a resultant negative value of a duration variable, which is not logical as duration cannot be a negative value. This itself should be a trigger that something is not appropriate in the computation even before the actual validation is performed. If the validation programmer misses relating the computed value with the corresponding variable, they can inadvertently mark the data set to pass validation if there are no mismatches in the PROC COMAPRE. However, the resultant variable values computed will be incorrect and not logical.

In this case the variable should rather be computed as

PERDUR = **(**DUR*100**) / (**STOPDT – STARTDT**)** = (20*100)/(19004 - 18775) = 8.7336244541

Hence, following the specifications down to the punctuation is not always the best approach, and the derived values should always be cross-verified carefully. Also, as SAS programmers come from varied backgrounds ranging from biology to computer science, to statistics it is important that the specifications be followed, but with a logical and clear understanding of concepts. In case of any confusion, there should not be any hesitation to clarify with the lead programmer or the biostatistician.

2B - Leading to loss of data points

Consider a demographic data set that derives variable AGECAT (Age category) as per specifications from Table 3 below. Using this derivation the PROC COMPARE results in “no unequal” observations.

Variable	Derivation

AGECAT	if age<20 then agecat=1;
	if 20<age<40 then agecat =2;
	if age>=40 then agecat=3

Table 3: Derivation of variable AGECAT (Age Category)

If we look closely at the derivation, the data value of AGE = 20 is being excluded from the computation; as a result, all subjects with AGE = 20 will not get assigned to any AGECAT. This may be a case of a typographical error in the specs, but if neither the original programmer nor the validation programmer check the computations in the specs, subjects with AGE = 20 will get excluded from all age category analysis, leading to a loss of data points. Hence it is important to understand that specifications are created to assist programming and may need updates/corrections at the time of programming/validation. It is advisable that derivations in the specs be cross-verified with the study documents not only by the original programmers, but by the validation programmers as well.

CONCLUSION

The aim of independent validation should not be to simply match the results between original and validation data sets or outputs, but to catch any aspects that the original programmer might have missed while programming or even something that the statistician may have missed while creating the specifications.

It is important to check whether the values are logical and consistent with study requirements in addition to matching in value. There can be different interpretations of the specs, but one needs to make sure the correct one is being followed. Besides, there can be gaps due to oversight in the specs as well as the original output. It is essential to understand the requirements first and then follow them.

A validation programmer should not only identify the differences between the original and the validation outputs, but also provide logical reasoning as to why one finds a certain value correct over the other. Hence is it recommended to be cautious so as not to be inadvertently “blind-folded” while performing validations; there should not be any hesitance to question or clarify.

ACKNOWLEDGMENTS

We would like to thank Dr. Prashant Kirkire, Country Head, India at inVentiv Health Care, for his valuable guidance and our peers and colleagues for carefully reviewing the paper with comments and suggestions.

CONTACT INFORMATION

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

Name: Neha Mohan
Enterprise: inVentiv Health Clinical
Address: Ground Floor, Marwah Centre, Krishanlal Marwah Marg, Andheri (E)
City, State ZIP: Mumbai-400072, Maharashtra, India
Work Phone: +91-22-4095-7365
E-mail: neha.mohan@inventivhealth.com ; neha.nm@gmail.com

Name: Gayatri Karkera
Enterprise: inVentiv Health Clinical
Address: Ground Floor, Marwah Centre, Krishanlal Marwah Marg, Andheri (E)
City, State ZIP: Mumbai-400072, Maharashtra, India
Work Phone: +91-22-4095-7364
E-mail: gayatri.karkera@inventivhealth.com ; gayatri.karkera@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.

Other brand and product names are trademarks of their respective companies.