Introduction:

As a statistical programmer at a CRO, I have regularly encountered the following situation: I'm running quality control (or validation) on an analysis or standardized dataset and, in doing so, I find many discrepancies between the primary dataset and the validation dataset. In simple terms, one, or both, of the datasets are 'sick.' In addition to having a 'sick' dataset on your hands, it is usually accompanied by a countdown clock to a deadline, only to add an additional layer of pressure.

These high-pressure situations have led me to develop what I call my "doctor's bag" of programming techniques and efficiencies. That is, I take a certain approach to diagnosing and fixing issues when there are gaps between validation datasets and primary datasets (or those datasets that are used to produce the final table, listings, and figures). These techniques are neither hidden nor novel but can be highly effective in rooting out programming and data issues.

The following techniques are best used in cases of validation programming for SDTM, SDTM-like, ADaM or ADaM-like datasets but I have used them in other situations as well.

To illustrate these methods for validation, I will use LB (the SDTM 'Laboratory' dataset) as an example.

Establishing a Baseline

Creating a validation dataset is an easy way to check the contents of the primary dataset. It can be a time-consuming process up front, but it has become a gold standard for review of the primary dataset (especially if there are more than a handful of observations). This is what is considered "establishing a baseline." Standardized and analysis datasets are often accompanied by data specifications which can include a description of the dataset domain, all of the variables that are supposed to appear in the domain, and how the variables are populated (either from a source variable or derived through an algorithm). The data specifications should serve as a roadmap for programming the validation dataset.

Returning to our example, let's determine what baseline LB would look like. At the SDTM level, the laboratory analysis dataset can be one of the trickiest to program. Often times the source data is coming from many places including local labs, a central lab, or any other tests that were collected on the CRF. Issues can arise when the same laboratory data is collected in two places or if a subject is missing a large amount of data.

For the purposes of this paper, we will assume the validation dataset, LB, is 'correct.' It is important to note, however, that discrepancies between a primary dataset and validation dataset can often come from problems in the validation dataset. Do not assume the validation dataset is necessarily "correct."

Baseline Diagnostics

You have established a baseline dataset using the data specifications as a roadmap and it is time to compare it to the primary dataset.

Here is the short list of critical components to a dataset:

1. the variables that are included in the dataset (including whether they are character or numeric)
2. the value levels of these variables
3. the number of observations in the dataset

This is not an exhaustive list but it highlights the attributes that will have the most impact on further production.

Taking the Vitals

So where do we being? It is easy to start by throwing a number of SAS procedures at the primary dataset and seeing what comes out. Start by taking a look at the primary dataset. As much as this sounds like common sense, it is best to take the holistic route and get a total picture of the dataset. This also serves as a good opportunity to assess the dataset for visible issues, such as truncation of records and missing data.

Now you are ready to compare your validation dataset (LB in this case) to the primary LB dataset.

Reaching into the "Doctor's Bag"

The first tool that comes out of the 'doctor's bag' is PROC COMPARE. It can give you a vast amount of information without a lot of code.

The basic syntax for PROC COMPARE:

proc compare base=primary_dataset compare=validation_dataset;
run;

This code will take all common variables between the two datasets and compare them observation by observation. It is helpful to get a read for number of observations in each dataset and the variables in common.

This information is at the beginning of the PROC COMPARE output:

The COMPARE Procedure
Comparison of WORK.PRIMARY_DATASET with WORK.VALIDATION_DATASET
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.PRIMARY_DATASET	21JAN17:18:08:44	21JAN17:18:08:44	15	428
WORK.VALIDATION_DATASET	21JAN17:18:08:44	21JAN17:18:08:44	15	421

Variables Summary

Number of Variables in Common: 15.

Differences in Variables

Consider for a moment that there are variables that were described in the data specifications that are not present in the primary dataset. It is critical that the primary dataset has all applicable variables.

Furthermore, you notice that the validation dataset (which has all of the LB variables needed) has more variables than the primary dataset. Instead of visually comparing the datasets, a useful alternative is the "LISTVAR" option in the compare procedure.

proc compare base=primary_dataset compare=validation_dataset **LISTVAR**;
run;

The COMPARE Procedure
Comparison of WORK.PRIMARY_DATASET with WORK.VALIDATION_DATASET
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.PRIMARY_DATASET	21JAN17:18:09:29	21JAN17:18:09:29	15	428
WORK.VALIDATION_DATASET	21JAN17:18:09:29	21JAN17:18:09:29	16	421

Variables Summary

Number of Variables in Common: 15.
Number of Variables in WORK.VALIDATION_DATASET but not in WORK.PRIMARY_DATASET: 1.

Listing of Variables in WORK.VALIDATION_DATASET but not in WORK.PRIMARY_DATASET

Variable	Type	Length
LBORRES	Char	1

This option will list the variables that are in the base dataset and not in the compare dataset and vice versa. It is important to make sure all variables that are supposed to be in the primary dataset ARE in the

primary dataset. It is very easy to see a complete match in the PROC COMPARE output on only a subset of variables.

Differences in Observations

Once you have determined that the primary dataset has all the variables in the data specifications, the next part to check is the number of observations in the dataset. This is often times the most difficult part of the validation process, especially when there are thousands of observations in the dataset.

If you use the basic PROC COMPARE syntax, the output produced will be a comparison of the datasets based on observation number for the variables in common. Try to consider the variables in the dataset that make the records unique from one another. In SDTM datasets such as LB, the dataset structure will have core variables such as USUBJID, LBTEST, and LBTESTCD, and VISIT. In most cases, each subject will have a record for each laboratory test at each planned visit, making these records unique when the aforementioned variables are used. The expectation is that the primary and validation datasets have the same number of laboratory tests and values per visit per subject. It can be difficult to identify the source of difference between the number of observations in a primary and validation dataset. There are two helpful ways to diagnose these differences: PROC FREQ and PROC COMPARE.

Stepping away from PROC COMPARE for a moment, consider the commonly used PROC FREQ. The uses for this procedure are endless. A quick way to discover deviations between datasets is to compare the frequency of records between datasets based on the value-level counts of a particular variable. That is, how many records does a particular subject ID have (USUBJID) or, in the case of LB, what is the frequency of each unique LBTEST value?

```
Proc freq data=primary_dataset noprint;
        Tables LBTEST / out=check1;
Run;

Proc freq data=validation_dataset noprint;
        Tables LBTEST / out=check2;
Run;

*add in output to support code*
```

You can do a quick visual check of the frequency counts in each dataset or merge the two datasets (CHECK1 and CHECK2 in this case) on LBTEST and COUNT to see the differences. The same method can be used in comparing USUBJID. If there are systematically fewer subject records or LBTEST values in one of the datasets, it can be a consequence of excluding a laboratory source.

Alternatively in PROC COMPARE we can use the ID statement in conjunction with the LISTOBS option to mine down to the distinct subject record/LBTEST/visit observation that is causing a mismatch.

```
Proc sort data=primary_dataset ; by SUBJECTID LBTEST VISIT; run;
```

```
Proc sort data= validation_dataset; by SUBJECTID LBTEST VISIT; run;

Proc compare base= primary_dataset compare=validation_dataset noprint LISTOBS;
ID USUBJID LBTEST VISIT;
Run;
```

insert proc compare output

The ID statement will match up all records that have the same USUBJID LBTEST and VISIT values and then break down the differences in other variables for this matchup (such as LBORRES, or the lab result value). The ID statement can be used without the LISTOBS option but a word of caution: if the datasets do not match on the ID statement variables, the unmatched observations will be excluded from the comparison. A cautious programmer can avoid this by either using the LISTOBS option OR carefully checking the PROC COMPARE output for the number of observations in common based on the ID statement. The upside to using the LISTOBS option is that the output will identify which records are present in one dataset and not in the other. The more variables you add to the ID statement, the more details you will have in the summary of the differences. I find it easier to start with variables in the dataset that make an observation unique and then adjust from there.

Diagnosing with Instinct

The validation tools described in this paper are simple but useful in diagnosing dataset discrepancies. Although these are basic SAS tools, they can give a lot of detail without too much programming. This being said, it is easy to get stuck in the 'weeds' of programming. These are the small differences in detail that can make a programmer lose sight of the bigger picture. It is easy to become so focused on getting a perfect match between datasets that the programming can deviate from the overall purpose of the dataset. It is important to take a step back and consider the entirety of the dataset. What is the end purpose of this data? Who will be reviewing it and what are the critical aspects of this data to the whole project. Projects can evolve over time. Be open to revisiting the validation dataset and updating what is considered the 'baseline.' Finally, when communicating discrepancies with the programmer of the primary dataset, be as specific as possible when describing the issue and refer back to the source data if necessary.