Paper 10840-2016

How to Speed Up Your Validation Process Without Really Trying

Alice M. Cheng, Portola Pharmaceuticals, Inc.; Michael R. Wise, Experis;
Justina M. Flavin, SimulStat, Inc.

ABSTRACT

The COMPARE procedure in SAS® offers relatively simple, yet powerful way to perform validation. In this paper, the authors will not only provide a brief introduction of the COMPARE procedure, but will also share their techniques to speed up the validation process. These techniques include the standard use of options and the clever use of features in PROC COMPARE. Also introduced are the use of &SYSINFO which summarizes the result of the comparison by a single numeric number, the macros %QCDATA and %QCDIR to validate a single dataset and all the datasets in two directories, respectively.

KEYWORDS: PROC COMPARE, Validation, Without Automation, &SYSINFO, With Automation

INTRODUCTION

The Latin motto *Finis origine pendet* means "The end depends on the beginning". In order to have accurate results from statistical analysis, it is critical that the source data is accurate. In clinical studies, as well as, in other disciplines, double programming is often performed to ensure accuracy. A developer and a validator are given the same data specification and are asked to program independently to create the same dataset. The validator then compares the validation dataset against the production dataset until the 2 datasets are completely identical in terms of the number of variables and observations, the attributes of datasets, the attributes and values of the variables. Often programmers have to go through multiple rounds using the COMPARE procedure before the validation is completed. In this paper, the authors provide tips and techniques that will speed up the validation process. Below are the topics that will be covered in this paper.

- A brief introduction to the COMPARE procedure
- Standard Use of Features in PROC COMPARE to speed up the validation of a dataset
- Clever Use of Features in PROC COMPARE to speed up the validation of a dataset
- A macro named %QCDATA to speed up the validation of a single dataset
- Introduction to &SYSINFO generated by the COMPARE procedure
- A macro named %QCDIR to identify all datasets in the production and validation directories

OVERVIEW

PROC COMPARE is a powerful procedure in SAS® that allows users to compare two datasets, to compare variables against variables of the same dataset or variables against variables between two datasets. It is a relatively simple, yet powerful procedure to use once users have mastered its features. According to SAS® Procedures Guide 9.4, the basic syntax for the COMPARE procedure is as follows:

```
PROC COMPARE <option(s)>;
     BY <DESCENDING> variable-1
        <DESCENDING> variable-2 ...
        <NOTSORTED>;
     ID <DESCENDING> variable-1
        <DESCENDING> variable-2 ...
        <NOTSORTED>;
     VAR variable(s);
     WITH variable(s);
```

PROC COMPARE statement has numerous options. Some most commonly used options include the following:

- **BASE=** option specifies the dataset to use as the base dataset. In PROC COMPARE, BASE= option is synonymous with the **DATA=** option.

- **COMPARE=** option specifies the dataset to use for comparison, if applicable. This dataset is referred as the COMPARE dataset. COMPARE dataset may not exist when comparing variables against variables in the same dataset. In that case, the WITH statement needs to be used.
- **LISTVAR** lists all variables found in only one dataset.
- **LISTOBS** lists all observations that are found only in one dataset.
- **LISTALL** is equivalent to applying both LISTVAR and LISTOBS. . The author prefers to use LISTVAR and LISTOBS options because these two options are more self-explanatory.
- **LISTOBS** lists all observations that are found only in one dataset.
- **LISTEQUALVAR** lists variables that match in values.
- **ALLOBS** includes in the report of value comparison results the values, and, for numeric variables, the differences for all matching observations, even if they are judged equal.
- **MAXPRINT= *total* | *(per-variable, total*)** specifies the maximum number of differences to be printed.
- **NOPRINT** suppresses all printed output.
- **OUTBASE, OUTCOMP, OUTDIF, OUTPERCENT and OUT=** options generate output dataset with the observations from the BASE dataset, the COMPARE dataset and an observation to indicate the difference and the difference in percent between the 2 datasets for any pair of observations, if any.
- **OUTALL** is equivalent to these 4 options: OUTBASE, OUTCOMP, OUTDIF and OUTPERCENT.
- **OUTNOEQUAL** suppresses the writing of an observation to the output dataset when all values in the observation are judged to be equal.

BY statement allows BY group comparison. When BY statement is used, please make sure BASE and COMPARE datasets are sorted according to the variables stated in the BY statement.

ID statement helps one to easily identify the observation(s) with discrepancies. It also comes with a NOTSORTED feature, though it is highly recommended that your dataset(s) are sorted in the order of the ID variables. If you use ID statement without NOTSORTED feature, your input data has to be sorted according to the ID variables. If you apply the NOTSORTED feature, and the ID variables of the 2 datasets do not match, SAS *may* abort the comparison. The only exception is when the ID variables match in the first *n* observations, where *n* is the number of observations in the input dataset (i.e. either BASE dataset or COMPARE dataset) with lesser observations. Comparison will then be performed based on the first n observations only. The authors would highly recommend users to use ID statement without the NOTSORTED features.

VAR statement enables one to specify the variables to be compared.

WITH statement restricts the comparison of values of variables to the ones named in this statement.

Since this paper emphasizes on validating a dataset as a whole, WITH statements and BY statements are not applied. In addition, Output Delivery System (ODS) features are also available in PROC COMPARE and one can get output datasets by means of ODS. However, ODS in PROC COMPARE is beyond the scope of this paper.

For a comprehensive introduction to the features in PROC COMPARE, users are highly encouraged to consult SAS Procedures Guide, the COMPARE procedure and Christianna S. Williams' paper "PROC COMPARE – Worth Another Look!". The latter provides an excellent introduction to PROC COMPARE through ten examples.

A SIMPLE EXAMPLE USING PROC COMPARE

EXAMPLE 1 – NOT A MATCH

ADSL, the Subject-Level Analysis Dataset, is the first Analysis Data Model (ADaM) dataset to be created in a clinical study. All other ADaM datasets depend on the ADSL dataset to a certain extent. ADSL contains one record per subject and includes demographic data, population flag variables, first dose date/time, last dose date/time, date of birth and date of death, etc. Below is a simplified version of ADSL.

Figure 1: PROD.ADSL and QC.ADSL

PROD.ADSL

OBS	SUBJID	SEX	AGE	AGEU	COUNTRY	HEIGHTBL	WEIGHTBL	BMIBL
1	101	M	32	YEARS	CHILE	176	79.5	25.7
2	102	F	29	YEARS	CHILE	169	82.4	29.2
3	201	F	41	YEARS	FRANCE	183	85.6	26.3
4	202	M	36	YEARS	FRANCE	178	94.2	29.1
5	301	M	52	YEARS	United States of America	172	81.6	27.5
6	302	F	35	YEARS	USA	174	98.5	32.5
7	303	M	56	YEARS	USA	163	84.4	31.7
8	303	M	48	YEARS	USA	172	64.8	64.8

QC.ADSL

OBS	SUBJID	SEX	AGE	COUNTRY	HEIGHTBL	WEIGHTBL	BMIBL
1	101	M	32	CHILE	176	79.5	25.7
2	102	F	29	CHILE	169	82.4	29.2
3	201	F	41	FRANCE	183	85.6	25.6
4	202	M	36	FRANCE	185	94.2	27.5
5	301	M	52	USA	172	81.6	27.5
6	303	M	56	USA	163	84.4	31.7
7	304	M	48	USA	172	64.8	64.8

In this example, PROD.ADSL, the BASE dataset is compared against QC.ADSL, the COMPARE dataset. Cells with red text indicate a potential problem. A quick glance over these 2 datasets reveals the following discrepancies:

- PROD.ADSL has 8 variables and 8 observations while QC.ADSL has 7 variables and 7 observations.
- Variable AGEU exists *only* in PROD.ADSL.
- There are some discrepancies in values of variables HEIGHTBL, BMIBL and COUNTRY.
- Subject 302 only has an observation *only* in PROD.ADSL.
- Subject 303 has 2 observations in PROD.ADSL, even though ADSL is supposed to have only one observation per subject.
- Subject 304 exists only in QC.ADSL.

Now let us observe the PROC COMPARE result of these 2 datasets. Below is the SAS code.

```
PROC COMPARE BASE=PROD.ADSL COMPARE=QC.ADSL LISTVAR LISTOBS LISTEQUALVAR
    MAXPRINT=(20, 2400);
    ID SUBJID; RUN;
```

Note that since ID statement is used, one should sort the data to make sure both the BASE and COMPARE datasets are sorted according to the order of the ID variable(s). In this example, sorting has been skipped because both the BASE and COMPARE datasets have already been sorted by the ascending order of the variable SUBJID.

Now as stated earlier in this paper, **LISTVAR** option lists all variables found only in one dataset; **LISTOBS** lists all observations that are found only in one dataset. **LISTEQUALVAR** lists variables that match in values. **MAXPRINT= *(per-variable, total)*** specifies the maximum number of differences to be printed. Since both PROD.ADSL and QC.ADSL have very few observations, the MAXPRINT= option is not of much use in this example, but is included here for demonstration. With **MAXPRINT=(20, 2400)** option, SAS has been instructed to print up to 20 discrepancies per variable and the maximum total number of differences to be printed is 2400. And since the VAR statement is not applied here, all variables, except the ID variable (SUBJID) are subjected to comparison.

Now let us take a look at the output. Below are the 5 reports from PROC COMPARE. Other than 'Comparison of Results for Observations', the other 4 summary reports (Data Set Summary, Variables Summary, Observation Summary, Value Comparison Results for Variables) are default output.

- **DATA SET SUMMARY**

The COMPARE Procedure
Comparison of PROD.ADSL with QC.ADSL
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs	Label
PROD.ADSL	20MAR16:02:08:27	20MAR16:02:08:27	8	8	Subject Level Analysis Data
QC.ADSL	08MAR16:16:57:12	08MAR16:16:57:12	7	7	Subject Level Data

Note EXACT method is used for comparison since METHOD= option has not been used and METHOD=EXACT is the default.

Based on the Data Set Summary Report, there are discrepancies in the number of variables and the number of observations in the 2 datasets. This agrees with what you saw earlier. In addition, it also indicates that the dataset labels are not identical.

- **VARIABLES SUMMARY**

Variables Summary

Number of Variables in Common: 7.
Number of Variables in PROD.ADSL but not in QC.ADSL: 1.
Number of Variables with Differing Attributes: 2.
Number of ID Variables: 1.

Listing of Variables in PROD.ADSL but not in QC.ADSL

Variable	Type	Length	Format	Informat	Label
AGEU	Char	10	$10.	$10.	AGEU

Listing of Common Variables with Differing Attributes

Variable	Dataset	Type	Length	Format	Informat	Label
SUBJID	PROD.ADSL	Char	4	$4.	$4.	Subject Identifier
	QC.ADSL	Char	4	$4.	$4.	Subject ID
AGE	PROD.ADSL	Num	8	3.	3.	Age (years)
	QC.ADSL	Num	8	8.	8.	Age in Years

Glancing over the Variable Summary Report above, the result is as expected. Just by observing the datasets, one can see that the variable AGEU exists only in PROD.ADSL. However, without looking into the content of the datasets, one does not know until seeing this PROC COMPARE output that there are discrepancies in the label for SUBJID and AGE and discrepancies in format and informat for WEIGHTBL and BMIBL.

- **COMPARISON RESULTS FOR OBSERVATIONS**

Comparison Results for Observations

```
Observation 6 in PROD.ADSL not found in QC.ADSL: SUBJID=302.
WARNING: The data set PROD.ADSL contains a duplicate observation at observation number 8.
NOTE: At observation 8 the current and previous ID values are:
      SUBJID=303.
NOTE: Further warnings for duplicate observations in this data set will not be printed.
```

```
Observation 8 in PROD.ADSL not found in QC.ADSL: SUBJID=303.

Observation 7 in QC.ADSL not found in PROD.ADSL: SUBJID=304.
```

This report reflects exactly what one observed before. Observations for Subject 302 are missing from QC.ADSL. Two duplicated observations for Subject 303 are found in PROD.ADSL. QC.ADSL does not have duplicated records for Subject 303 and hence, the second duplicated record (Observation 8) is missing. And again, as stated before, Subject 304 only has an observation in QC.ADSL.

Perhaps what is more interesting is the NOTE message: "NOTE: Further warnings for duplicate observations in this dataset will not be printed". In other words, PROC COMPARE will only give you a warning when it first encounters duplication observations in BASE or COMPARE datasets. After that, it is up to the users to identify the duplicates!

- **OBSERVATION SUMMARY**

Observation Summary

Observation	Base	Compare	ID
First Obs	1	1	SUBJID=101
First Unequal	3	3	SUBJID=201
Last Unequal	5	5	SUBJID=301
Last Match	7	6	SUBJID=303
Last Obs	8		SUBJID=303
		7	SUBJID=304

```
Number of Observations in Common: 6.
Number of Observations in PROD.ADSL but not in QC.ADSL: 2.
Number of Observations in QC.ADSL but not in PROD.ADSL: 1.
Number of Duplicate Observations found in PROD.ADSL: 1.
```

```
Total Number of Observations Read from PROD.ADSL: 8.
Total Number of Observations Read from QC.ADSL: 7.

Number of Observations with Some Compared Variables Unequal: 3.
Number of Observations with All Compared Variables Equal: 3.
```

The Observation Summary Report can be easily explained. The 6 observations in common are those corresponding to SUBJID='101', '102', '201', '202', '301' and '303'. The 2 observations that are in PROD.ADSL but not in QC.ADSL are those with SUBJID = '302' and '303'. The single observation in QC.ADSL, but not in PROD.ADSL is the observation for SUBJID='304'. And yes, a pair of duplicates have been uncovered; they are related to SUBJID='303'! Observations with SUBJID='101', '102' and '303' have all values equal while observations with SUBJID='201', '202' and '301' have some discrepancies.

- ## VALUE COMPARISON RESULTS FOR VARIABLES

Values Comparison Summary

```
Number of Variables Compared with All Observations Equal: 3.
Number of Variables Compared with Some Observations Unequal: 3.
Total Number of Values which Compare Unequal: 5.
Maximum Difference: 7.
```

Variables with All Equal Values

Variable	Type	Len	Label	Compare Label
AGE	NUM	8	Age (years)	Age in Years
SEX	CHAR	1	Sex	Sex
WEIGHTBL	NUM	8	Weight at Baseline (kg)	Weight at Baseline (kg)

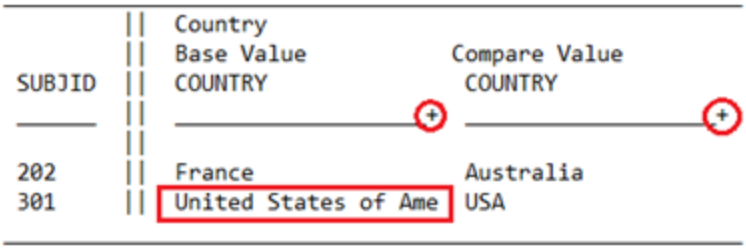
Variables with Unequal Values

Variable	Type	Len	Label	Ndif	MaxDif
HEIGHTBL	NUM	8	Height at Baseline (cm)	1	7.000
BMIBL	NUM	8	BMI at Baseline (kg/m**2)	2	1.600
COUNTRY	CHAR	30	Country	2	

Value Comparison Results for Variables

	Height at Baseline (cm)			
	Base	Compare		
SUBJID	HEIGHTBL	HEIGHTBL	Diff.	% Diff
202	178	185	7.0000	3.9326

	BMI at Baseline (kg/m**2)			
	Base	Compare		
SUBJID	BMIBL	BMIBL	Diff.	% Diff
201	26.3	25.6	-0.7000	-2.6616
202	29.1	27.5	-1.6000	-5.4983



SUBJID	Country Base Value COUNTRY	Compare Value COUNTRY
202	France	Australia
301	United States of Ame	USA

The Value Comparison Results for Variable Report displays the discrepancies in variables. Here, one sees discrepancies in HEIGHTBL, BMIBL and COUNTRY. Note that the value for COUNTRY='United States of America' has been truncated. This is because all variables PROC COMPARE output is confined to 20 characters, regardless if it is an ID variable or a variable to be compared. The PLUS symbol (+) in the line above are there to indicate the values of the variables are longer than what is displayed.

EXAMPLE 2 – PERFECT MATCH

As an example of a perfect match, PROD.ADSL was selected to compare with itself, as indicated in the code below:

```
PROC COMPARE BASE=PROD.ADSL COMPARE=PROD.ADSL LISTVAR LISTOBS LISTEQUALVAR;
   ID SUBJID; RUN;
```

Part of the output is shown below.

```
                    Observation Summary

              Observation      Base  Compare  ID

              First Obs           1        1  SUBJID=101
              Last  Obs           8        8  SUBJID=303

      Number of Observations in Common: 8.
      Number of Duplicate Observations found in PROD.ADSL: 1.
      Number of Duplicate Observations found in PROD.ADSL: 1.
      Total Number of Observations Read from PROD.ADSL: 8.
      Total Number of Observations Read from PROD.ADSL: 8.

      Number of Observations with Some Compared Variables Unequal: 0.
      Number of Observations with All Compared Variables Equal: 8.

      NOTE: No unequal values were found. All values compared are exactly equal.
```

"NOTE: No unequal values were found. All values compared are exactly equal."

This NOTE brings joy to a programmer's heart. "Eureka! We are done!" While it is true that this is a perfect match in this example, there are cases where the match is not perfect and still get this statement as demonstrated by Joshua Horstman and Roger Muller (2013). Moreover, even though this perfect match here, the dataset PROC.ADSL itself is not faultless. Remember this dataset has 2 observations with Subject 303; yet SUBJID is the unique identifier. If this is not the Holy Grail, what is then? This will be discussed in more detail later in this paper.

STANDARD USE OF FEATURES IN PROC COMPARE TO SPEED UP VALIDATION

- ### USE OF DIVIDE AND CONQUER TECHNIQUES

The example above is for illustration and hence, the number of observations in ADLS is relatively small. In reality, PROC COMPARE can generate a lot of output if the datasets have more observations that

have a lot of discrepancies. So to resolve the discrepancies, it is advisable to concentrate on one or a few discrepancies at a time!

Use WHERE statement. For instance, concentrate on one subject at a time. This allows you to easily see other values of the same subject, which can be helpful in resolving discrepancies. The value for change (CHG) can depend on the value of the current value (AVAL) and baseline value (BASE). So are the discrepancies in CHG a result of discrepancies in AVAL and/or BASE? To use WHERE statement in PROC COMPARE is the similar as what you use in other procedures. Below is a simple example.

```
PROC COMPARE BASE=PROD.ADSL COMPARE=QC.ADSL LISTVAR LISTOBS LISTEQUALVAR;
   ID SUBJID;
   WHERE SUBJID='303'; RUN;
```

Use VAR statement. Investigate one variable or several related variables at a time. For instance, in Example 1, there are discrepancies in BMIBL. Since BMI=(Weight in kg) / (Height in meter)^2, one may want to concentrate on all these 3 variables at the same time, as seen in the code below.

```
PROC COMPARE BASE=PROD.ADSL COMPARE=QC.ADSL LISTVAR LISTOBS LISTEQUALVAR;
   ID SUBJID;
   VAR HEIGHTBL WEIGHTBL BMIBL; RUN;
```

Apply DROP = option to *both* BASE and COMPARE datasets. One may drop the variables that have already been investigated and concentrate on the ones whose discrepancies have yet to be resolved. This can reduce the amount of output as well. Say, the discrepancies in HEIGHTBL, WEIGHTBL and BMIBL have already been investigated, but there are also a lot other variables to be looked into. Instead of listing all the remaining variables in a VAR statement, one may simply drop a few variables.

```
PROC COMPARE BASE=PROD.ADSL (drop = HEIGHTBL WEIGHTBL BMIBL)
  COMPARE=QC.ADSL (drop = HEIGHT WEIGHT BMIBL)
  LISTVAR LISTOBS LISTEQUALVAR;
  ID SUBJID; RUN;
```

Of course, one can use a combination of WHERE statement, VAR statement and DROP= option to achieve the best results. In fact, one is encouraged to use a combination of these techniques, as well as, techniques introduced later in this paper.

- ## BE PARSIMONIOUS IN ID VARIABLES SELECTION AND ID VARIABLE VALUES

Sometimes SDTM or ADaM data specification may have specified more key variables than needed. You may want to drop those variables since there are limited spaces to print out all these variable values in its Value Comparison Report. Moreover, you may want to shorten the values of variables, if possible to avoid truncation.

For example, the key identifier for Analysis of Lab Tests dataset, ADLB can be:

Study ID, Subject ID, Lab Category, Lab Parameter, Analysis Visit and Analysis Date

Now if this is only on single study in the dataset, Study ID can be dropped as part of the ID variables. Similarly, Lab Category does not give additional information because with some clinical knowledge, once one knows what the lab test it is, one knows what lab category it belongs. So the ID variables used in PROC COMPARE can be simplified to:

Subject ID, Lab Parameter, Analysis Visit and Analysis Date

Moreover, in some clinical studies, Subject ID can be lengthy, such as ABCD-1234-56-0000000001, where ABCD-1234-56 represents the Study ID and every Subject ID in the same study begins with ABCD-1234-56. In that case, one may want to exclude the Study ID part in the Subject ID because PROC COMPARE only displays the first 20 characters of a variable value in its report.

- ## REDUCE THE AMOUNT OF PRINTED OUTPUT

Use MAXPRINT= *total* | *(per-variable, total*).

As stated in the Procedures Guide for the COMPARE procedure, **MAXPRINT= *total* | *(per-variable, total)*** specifies the maximum number of differences to be printed.

Total is the maximum total number of differences to be printed. The default value is 500 unless you use ALLOBS option (or both the ALLVAR and TRANSPOSE options). In that case the default is 32000.

Per-variable is the maximum number of differences to be printed for each variable within BY group. The default value is 50 unless you use the ALLOBS option (or both ALLVAR and TRANSPOSE options). In that case, the default is 1000.

The MAXPRINT= option prevents the output from being extremely large when there are a lot of discrepancies between the BASE and COMPARE datasets.

Caveat: one needs to make sure that

(The Value Specified *for per-variables*) X (Number of Non-ID Variables and Non-BY Variables) <= *total*

Otherwise, some discrepancies would not be printed.

- ## OUTPUT RESULTS IN A DATASET, INSTEAD OF PRINTING

Use of OUTBASE, OUTCOMP, OUTDIF, OUTPERCENT, OUTNOEQUAL and OUT= options. Suppress printing by NOPRINT options.

Perhaps printing is not needed. Viewing an output dataset containing discrepancies is sufficient. The use **of OUTBASE, OUTCOMP,OUTDIF, OUTPERCENT** and **OUT=** options in the following code enables one to create an output dataset with values from BASE and COMPARE datasets, the difference and the percent difference. **OUTNOEQUAL** option instructs the COMPARE procedure to only output observations with discrepancies. One can even use **NOPRINT** options to suppress the printing completely. However, it is crucial that an output dataset is specified; otherwise, SAS will ignore the OUTBASE, OUTCOMP, OUTDIF, OUTPERCENT and OUTNOEQUAL options. Below is an example using PROD.ADSL and QC.ADSL.

```
PROC COMPARE BASE=PROD.ADSL COMPARE=QC.ADSL
   OUT=OUT_EX1 OUTBASE OUTCOMP OUTDIF OUTPERCENT OUTNOEQUAL NOPRINT;
   ID SUBJID; RUN;
```

Figure 2: OUT_EX1 generated by OUT=OUT_EX1, OUTBASE, OUTCOMP, OUTDIF and OUTPERCENT Options

	TYPE	_OBS_	SUBJID	AGE	SEX	HEIGHTBL	WEIGHTBL	BMIBL	COUNTRY
1	BASE	3	201	41	F	183	85.6	26.3	France
2	COMPARE	3	201	41	F	183	85.6	25.6	France
3	DIF	3	201	E		E	E	-0.7	.....
4	PERCENT	3	201	E		E	E	-2.7	.....
5	BASE	4	202	36	M	178	94.2	29.1	France
6	COMPARE	4	202	36	M	185	94.2	27.5	Australia
7	DIF	4	202	E		7	E	-1.6	XXXXXXXXX.....
8	PERCENT	4	202	E		4	E	-5.5	XXXXXXXXX.....
9	BASE	5	301	52	M	172	81.6	27.5	United States of America
10	COMPARE	5	301	52	M	172	81.6	27.5	USA
11	DIF	5	301	E		E	E	E	XXXXX.XXXXXX.XX.X...
12	PERCENT	5	301	E		E	E	E	XXXXX.XXXXXX.XX.X...
13	BASE	6	302	35	F	174	98.5	32.5	USA
14	BASE	8	303	48	M	172	64.8	21.9	USA
15	COMPARE	7	304	48	M	172	64.8	21.9	USA

OUT_EX1 contains only observations that are not a match. When the value of an ID variable exists in both the BASE and COMPARE datasets (as in the case for SUBJID='201'), the code above creates 4 observations of _TYPE_='BASE', 'COMPARE', 'DIF' and 'PERCENT'. _TYPE_='BASE' and _TYPE_='COMPARE' represent values from BASE and COMPARE datasets, respectively. _TYPE_='DIF' shows the difference and _TYPE_='PERCENT' shows the percent difference.

Because OUTNOEQUAL option is used, for numeric values, the special missing value '.E' is used to represent differences and percent differences for variables that are judged to be equal. For character variables, regardless of OUTNOEQUAL option, a missing value '.' is used for values judged to be equal and a 'X' symbol is used to represent differences and percent differences if there is a discrepancy.

Also, note that for ID variables that do not have a match only _TYPE_='BASE' or _TYPE_='COMPARE' are displayed, as seen in Observations 13-15 of OUT_EX1 dataset.

In addition to saving trees, viewing the output datasets has two BIG advantages over printing the output or viewing the PROC COMPARE reports: (1). In event there are a lot of ID variables, only some ID variables can be displayed in a PROC COMPARE report, but all these ID variables can be in an output dataset. (2). Values from variables are limited to 20 characters in PROC COMPARE report. Any values longer than that will be truncated as seen in Example 1. Output the dataset, instead of creating a report, eliminates these problems.

Knowing the values of the identifiers is critical so that one can quickly identify the problematic observations and the variable values; and hence, speed up the validation process.

Art Carpenter's Method to Output Data with Discrepancies

Perhaps it may be easier to review the data and its discrepancies if the OUT_EX1 dataset has been transposed. Art Carpenter developed an algorithm to transpose the output data from PROC COMPARE. This method creates a report that displays all discrepancies for different variables one subject at a time. Interested readers should consult Mr. Carpenter's book on 'Innovative SAS® Techniques' for details.

CLEVER USE OF FEATURES IN PROC COMPARE TO SPEED UP VALIDATION

Below are two clever methods that the author Cheng often uses to resolve discrepancies. These methods are handy because one can easily make PROC COMPARE display the source variable values instead of looking up these values in the BASE and COMPARE datasets. The author calls these methods 'clever' because she is actually tweaking PROC COMPARE to display the values she wants to see. Of course, this only works if the SOURCE variables are still in both the BASE and COMPARE datasets.

- ### ADD SOURCE VARIABLES FROM BASE AND COMPARE DATASETS TO THE ID STATEMENT

Recall when comparing PROD.ADSL against QC.ADSL, there were 2 discrepancies in BMIBL.

		BMI at Baseline (kg/m**2)			
		Base	Compare		
SUBJID		BMIBL	BMIBL	Diff.	% Diff
201		26.3	25.6	-0.7000	-2.6616
202		29.1	27.5	-1.6000	-5.4983

How would you resolve these discrepancies? Since

BMI = (Weight in kg) / (Height in meter)^2

it is logical to seek out the value of source variables for Weight and Height. Fortunately, these variables are both in PROD.ADSL and QC.ADSL. So instead of looking up these variables from BASE and COMPARE datasets, why not make PROC COMPARE do the job for you? Simply add the variable Weight in kg at Baseline (WEIGHTBL) and the variable for Height in cm at Baseline (HEIGHTBL) to the ID statement. Please make sure existing ID variables are listed first followed by WEIGHTBL and

HEIGHTBL. Note that both the BASE and COMPARE datasets need to be sorted to ensure the observations are ordered by original ID variables, WEIGHTBL and HEIGHTBL. Below is the code.

```
PROC SORT DATA=PROD.ADSL OUT=PROD_ADSL;
   BY SUBJID WEIGHTBL HEIGHTBL; RUN;

PROC SORT DATA=QC.ADSL OUT=QC_ADSL;
   BY SUBJID WEIGHTBL HEIGHTBL; RUN;

PROC COMPARE BASE=PROD_ADSL COMPARE=QC_ADSL LISTVAR LISTOBS LISTEQUALVAR;
   ID SUBJID WEIGHTBL HEIGHTBL; RUN;
```

From the output, the following information is helpful to resolve these discrepancies in BMIBL.

Comparison Results for Observations

```
Observation 4 in WORK.PROD_ADSL not found in WORK.QC_ADSL: SUBJID=202
 HEIGHTBL=178 WEIGHTBL=94.2.

Observation 4 in WORK.QC_ADSL not found in WORK.PROD_ADSL: SUBJID=202
 HEIGHTBL=185 WEIGHTBL=94.2.
```

Value Comparison Results for Variables

				BMI at Baseline (kg/m**2)			
				Base	Compare		
SUBJID	HEIGHTBL	WEIGHTBL		BMIBL	BMIBL	Diff.	% Diff
201	183	85.6		26.3	25.6	-0.7000	-2.6616

From Comparison Results for Observations Report, SUBJID='202' actually has a different value for Height at Baseline (HEIGHTBL). Chances are good that the discrepancies in BMIBL are due to different values for HEIGHTBL. On the other hand, based on Value Comparison Results for Variables, the values for HEIGHTBL and WEIGHTBL are identical. So most likely the discrepancy is a result of miscalculation. Indeed, in this example, the value of BMIBL from the BASE dataset is incorrect due to miscalculation!

- **ADD BOTH THE SOURCE VARIABLES AND THE VARIABLE TO BE COMPARED TO THE ID STATEMENT**

Another convenient way to see the values of the source variable and the variable to be compared is to add all these variables to the ID statement. The ID statement should be in this order: *key variables, source variables, variable to be compared.* This will ensure the results are lined up in the desirable order for easy viewing.

```
PROC SORT DATA=PROD.ADSL OUT=PROD_ADSL;
   BY SUBJID WEIGHTBL HEIGHTBL; RUN;

PROC SORT DATA=QC.ADSL OUT=QC_ADSL;
   BY SUBJID WEIGHTBL HEIGHTBL; RUN;

PROC COMPARE BASE=PROD_ADSL COMPARE=QC_ADSL LISTVAR LISTOBS LISTEQUALVAR;
   ID SUBJID WEIGHTBL HEIGHTBL BMIBL; RUN;
```

Below is a partial output from Comparison Results for Observations Report. Note that the 2 pairs of

observations with discrepancies in BMIBL have lined up nicely for our viewing. And in event the variables have longer than 20 characters, it will not be truncated here.

```
                    Comparison Results for Observations

Observation 3 in WORK.QC_ADSL not found in WORK.PROD_ADSL: SUBJID=201
 HEIGHTBL=183 WEIGHTBL=85.6 BMIBL=25.6.

Observation 3 in WORK.PROD_ADSL not found in WORK.QC_ADSL: SUBJID=201
 HEIGHTBL=183 WEIGHTBL=85.6 BMIBL=26.3.

Observation 4 in WORK.PROD_ADSL not found in WORK.QC_ADSL: SUBJID=202
 HEIGHTBL=178 WEIGHTBL=94.2 BMIBL=29.1.

Observation 4 in WORK.QC_ADSL not found in WORK.PROD_ADSL: SUBJID=202
 HEIGHTBL=185 WEIGHTBL=94.2 BMIBL=27.5.
```

One may not appreciate these clever techniques. But imagine the result value can be based on multiple source variables, having the result from the second clever method *is* a Godsend!

In addition, as stated previously, one can view the source variables and the variable to be compared by creating an output dataset using OUTBASE, OUTCOMP, OUTDIF, OUTPERCENT, OUT= option. But unless you limit the variables, you will be inundated with other variables as well. Moreover, even if you limit your variables, the resulting dataset may not be as viewer-friendly as the printout above.

Caveat: The second clever technique works assuming that you have variables, other than those in the ID statement in the BASE and COMPARE datasets. Otherwise, SAS will not perform the comparison before there is nothing to compare! Instead, you will find a statement like:

```
NOTE: Except for the 4 ID variables, the data sets WORK.PROD_ADSL and
WORK.QC_ADSL have no variables in common. There are no matching variables to
compare. Comparisons of data values not performed.
```

Of course, this can be easily resolved by creating a common dummy variable in BASE and COMPARE datasets.

%QCDATA TO SPEED UP VALIDATION OF A DATASET

Using the clever techniques described in the last section can speed up the validation process. Sometimes, one may need to change ID variables frequently because the source variables are likely to be different for a different variable. Alas! PROC COMPARE requires variables in ID statement to be sorted. (Yes, ID statement does have a NOTSORTED option, but that requires the ID variables to be matched in both BASE and COMPARE datasets, which may not be the case when your data still needs cleaning!) So far, only techniques without automation have been discussed. Wouldn't it be nice to have a macro that will sort these ID variables automatically? For that and other purposes, the authors have created a macro named **%QCDATA**. It is meant to be a simple macro, so complicated check for the validity of parameter input has not been incorporated in this macro.

%QCDATA achieves the following:

- Sort BASE and COMPARE datasets based on ID variables.
- Identify and print out duplicate observations of BASE and COMPARE datasets upon request, if applicable.
- Create Duplicate Observation Flag variables BASE and COMAPRE datasets.
- Allow users to specify ID variables to be used in PROC COMPARE.
- Allow users to specify variables to be compared.
- Allow users to drop variables not to be compared.
- Perform PROC COMPARE.

Based on request from users, generate PROC COMPARE report, output dataset and a dataset with the values of the flag variables, return code from PROC COMPARE and the status of the 16 conditions based on the return code %SYSINFO.

Below are the parameters based on the code from **%QCDATA** macro.

```
%macro QCDATA (
    BASE=         /* Enter the name of the BASE dataset. (Required)       */
   ,COMPARE=      /* Enter the name of the COMPARE dataset. (Required)    */
   ,IDVARS=       /* Enter a list of space-delimited ID variables.        */
                  /* (Optional, but Highly Recommended)                   */
   ,VARS=         /* Enter a list of space-delimited variables to be      */
                  /* compared. (Optional)                                 */
                  /* Note: If not specified, all variables other than the */
                  /* ID variables will be compared.                       */
  ,DROPBASEVARS= /* Enter a list of space-delimited variables from BASE  */
                  /* dataset to be dropped from comparison. (Optional)    */
  ,DROPCOMPVARS= /* Enter a list of space-delimited variables from       */
                  /* COMPARE dataset to be dropped from comparison.       */
                  /* (Optional)                                           */
   ,PRINTDUP=Y    /* Print Duplicate observations in BASE and COMPARE     */
                  /* datasets, if applicable.  (Optional)                 */
                  /* Values:  Y, N   Default: Y                           */
   ,COMPRPT=Y     /* Generate report from PROC COMPARE. (Optional)        */
                  /* Values:  Y, N   Default: Y                           */
   ,CREATEOUTDS=Y /* Create output dataset from PROC COMPARE. (Optional)  */
                  /* Values:  Y, N   Default: Y                           */
   ,OUTDATA=      /* Enter the name for the output dataset for PROC       */
                  /* COMPARE.  (Optional)                                 */
   ,OUTRC=        /* Enter the name of the dataset for return codes       */
                  /* and other flag variables.                            */
   ,REFRESH=N     /* Refresh the dataset specified in OUTRC= .            */
                   /* Values: Y, N   Default:  N                          */
)/des="To QC data, identify duplicate observations, output Return Codes and
       Duplicate Flags";
```

COMPRPT=Y and **CREATEOUTDS=Y** generate a typical PROC COMPARE report and datasets, respectively. The output layout/format has been discussed in the introduction section of this paper. For the complete definition of **%QCDATA**, please consult the appendix.

Perhaps, what is new here is **OUTRC=** parameter. It requests the output of return codes. What does that mean? Let us try to invoke this macro a several time and examine the output of return codes. Below is a simple driver program to invoke %QCDATA macro four times.

```
LIBNAME PROD  "C:\STUDY\ABC\ABC-CL-1001\DATA\PROD\ADAM" ACCESS=READONLY;
LIBNAME QC    "C:\STUDY\ABC\ABC-CL-1001\DATA\QC\ADAM" ACCESS=READONLY;
FILENAME MACROS "C:\STUDY\ABC\ABC-CL-1001\MACROS" ACCESS=READONLY;
OPTIONS SASAUTOS=(SASAUTOS, MACROS);

%QCDATA(BASE=PROD.ADSL,COMPARE=QC.ADSL,IDVARS=SUBJID,PRINTDUP=Y,
        COMPRPT=Y,OUTDATA=COMPARE1,OUTRC=RC,REFRESH=Y);
%QCDATA(BASE=PROD.ADSL,COMPARE=PROD.ADSL,IDVARS=SUBJID,PRINTDUP=Y,
         COMPRPT=Y,OUTDATA=COMPARE2,OUTRC=RC,REFRESH=N);
%QCDATA(BASE=QC.ADSL,COMPARE=QC.ADSL,IDVARS=SUBJID,PRINTDUP=Y,
         COMPRPT=Y,OUTDATA=COMPARE3,OUTRC=RC,REFRESH=N);
%QCDATA(BASE=PROD.ADAE,COMPARE=QC.ADAE,
        IDVARS=SUBJID AESTDTC AEENDTC AEDECOD AETERM,PRINTDUP=Y,
         COMPRPT=Y,OUTDATA=COMPARE4,OUTRC=RC,REFRESH=N);
```

Note that %QCDATA has been invoked four times to perform four comparisons: PROD.ADSL vs. QC.ADSL, PROD.ADSL vs. PROD.ADSL, QC.ADSL vs. QC.ADSL and PROD.ADAE vs. QC.ADAE. Each time a PROC COMPARE printout and an output dataset containing the result of BASE, COMPARE, the difference of BASE vs. COMPARE and the percent difference of BASE vs COMPARE were generated. Also generated is a dataset named RC. This dataset RC has been refreshed at the first invocation of %QCDATA and return code of each comparison has been appended to a dataset named RC. Let us take a look at dataset RC.

Figure 3: Dataset RC from %QCDATA

RC										
OBS	**BASE**	**COMPARE**	**IDVARS**	**DUPB**	**DUPC**	**RC**	**RC_0**	**RC_1**	**…**	**RC_16**
1	PROD.ADSL	QC.ADSL	SUBJID	Y	N	5357		Y		
2	PROD.ADSL	PROD.ADSL	SUBJID	Y	Y	0	Y			
3	QC.ADSL	QC.ADSL	SUBJID	N	N	0	Y			
4	PROD.ADAE	QC.ADAE	SUBJID AESTDTC AEENDTC AEDECOD AETERM	N	N	48				

Obviously, the variables **BASE** and **COMPARE** hold the value of the BASE and COMPARE datasets, respectively. **IDVARS** contains a list of ID variables that are supposed to uniquely identify an observation in both BASE and COMPARE datasets. **DUPB=Y** when there are duplicated records in BASE dataset based on the ID variables; **DUPC=Y** serves the same purpose for COMPARE dataset.

Now what are RC, RC_0, RC_1, …, RC_16 ? **RC** contains the value of the return code &SYSINFO just after PROC COMPARE has been executed. **RC_1, … , RC_16** represents the 16 conditions after PROC COMPARE has been executed. **RC_0** was created by the authors to indicate a perfect match. These values are derived from the value of RC. Further explanation is provided in the next session.

INTRODUCTION OF &SYSINFO GENERATED BY PROC COMPARE

After the execution of the COMPARE procedure, a return code value will be generated and stored in the macro variable **&SYSINFO**. This code indicates the result of comparison of the BASE dataset against the COMPARE dataset. It is critical that one retrieves this value *immediately* after the COMPARE procedure because its value will change once when SAS encounters another data step or procedure. On the following page is a table indicating all the 16 conditions that SAS will check. This check is a modified version of the one in the SAS Procedures Guide, the COMPARE procedure.

Based on this table, there are 16 conditions representing the 16 possible errors one gets from PROC COMPARE. The RC dataset in Figure 3 will have a 'Y' value to indicate the corresponding error condition has met. For instance, RC_1='Y' means the first error condition has met; RC_2='Y' means the second error condition has met and so on. Now it is possible a comparison has met one or more of these conditions. As seen in this table, each condition has a corresponding code. The value of **&SYSINFO** is the sum of these codes if the corresponding conditions are met. These code values are actually nicely chosen so that the **code value = 2 ** (Condition Number-1**). So the first condition has a value of 2**(1-1)=1; the second condition has a value of 2**(2-1)=2 and so forth. Putting this in binary representation, it corresponds to a 1 in a position in binary representation. Looking at the values of these codes carefully, one can note that each of these conditions represents a value of '1' in different position when put in binary 16. format. (See Figure 4.) *In fact, these codes are so well chosen that knowing the sum of all the codes is sufficient to identify the codes that add up to this sum. In other words, knowing the sum of all the codes*

enables one to identify the error conditions. And the sum of the codes is none other than the return value from PROC COMPARE stored in &SYSINFO, which was saved in the variable RC in Figure 3.

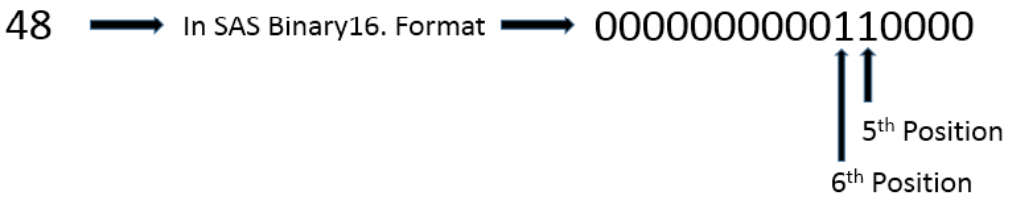
Also note that based on Figure 4, if the value of &SYSINFO is less than 64, the discrepancies are a result of the difference in attributes. On the other hand, if the value of &SYSINFO is 64 or higher, then there are more serious discrepancies. In fact, the higher the value of &SYSINFO from PROC COMPARE, the more serious the discrepancies are! Now earlier in this paper, it has been commented that the message in PROC COMPARE output:

```
"NOTE:  No unequal values were found.  All values compared are exactly equal."
```

Is not the Holy Grail. Horstman and Muller (2013) have provided 4 nice examples warning users not to be blind-sided by this message. So, what is the *Holy Grail*? As SAS Technical Support representative, Jane Eslinger has confirmed, **the Holy Grail is *&SYSINFO=0*.** *When none of the 16 conditions are met, the sum of &SYSINFO=0, indicating a perfect match!* Here in Figure 3, the authors have set RC_0='Y' when there is a perfect match. And since RC contains the value directly from &SYSINFO, RC=0 indicates a perfect match!

How to Interpret the value of &SYSINFO

As an example, let us consider Figure 3. PROD.ADAE vs QC.ADAE comparison results in a return code of 48. Since 48 < 64, the discrepancies must be in attributes. But what are the problematic attributes? Let's put 48 in binary format.



The 5th and 6th positions from the right have a value of 1 meaning that the 5th and 6th conditions stated in Figure 4 have been violated. In other words, when comparing PROD.ADAE against QC.ADAE, there is a variable with a different length and different label.

One can use this method to identify which (if any) of the 16 conditions have been violated. However, it would be much easier to use Bit Map Testing to check for the value of 1 in a position, and hence, decide if the corresponding conditions have been violated. Below is an extraction of the code for Bit Map Testing in **%QCDATA**. Note if RC='1….'b and RC='1…..'b check if the 5th and 6th positions are 1, respectively.

Code for Bit Map Testing

```
/* Test for Data Set Label. */
if RC='1'b then do;
   RC_1='Y';  put '<<< Data set labels differ.'; end;
/* Test for Data Set Types. */
if RC='1.'b then do;
   RC_2='Y';  put '<<< Data set types differs'; end;
    ... more codes ...
/* Test if there is a fatal error: comparison not done. */
if RC='1..............'b then do;
    RC_16='Y'; put '<<< There is a fatal error: comparison not done.'; end;
```

%QCDIR TO SPEED UP VALIDATION OF ALL DATASETS IN PRODUCTION DIRECTORY AGAINST ALL DATASETS IN QC DIRECTORY

In the last section, %QCDATA was invoked 4 times to compare 4 pairs of datasets. Consider the datasets in both PROD and QC directories, where the QC directory is supposed to contain the datasets for validating their namesakes in PROD directory. In the example below, ADaM datasets ADAE, ADIE, ADLB, ADSL and ADVS exist in both PROD and QC directories, while ADCM and ADMH exist in only QC and PROD directories, respectively.

Figure 4: Macro Return Codes

Bit	Condition	Code	Description	Value Used in binary 16. Format	Value Used in Bit-Testing
1	DSLABEL	$1=2^0$	Data set labels differ.	0000000000000001	‘1’b
2	DSTYPE	$2=2^1$	Data set types differ.	0000000000000010	‘1.’b
3	INFORMAT	$4=2^2$	Variable has different informat.	0000000000000100	‘1..’b
4	FORMAT	$8=2^3$	Variable has different format.	0000000000001000	‘1…’b
5	LENGTH	$16=2^4$	Variable has different length.	0000000000010000	‘1….’b
6	LABEL	$32=2^5$	Variable has different label.	0000000000100000	‘1…..’b
7	BASEOBS	$64=2^6$	Base data set has observation not in comparison.	0000000001000000	‘1……’b
8	COMPOBS	$128=2^7$	Comparison data set has observations not in base.	0000000010000000	‘1…….’b
9	BASEBY	$256=2^8$	Base dataset has BY group not in comparison.	0000000100000000	‘1……..’b
10	COMPBY	$512=2^9$	Comparison dataset has BY group not in base.	0000001000000000	‘1.....’b
11	BASEVAR	$1024=2^{10}$	Base dataset has variable not in comparison.	0000010000000000	‘1.....’b
12	COMPVAR	$2048=2^{11}$	Comparison dataset has variable not in base.	0000100000000000	‘1……..’b
13	VALUE	$4096=2^{12}$	A value comparison was unequal.	0001000000000000	‘1……’b
14	TYPE	$8192=2^{13}$	Conflicting variables types.	0010000000000000	‘1……’b
15	BYVAR	$16384=2^{14}$	BY variables do not match.	0100000000000000	‘1……..’b
16	ERROR	$32768=2^{15}$	Fatal error: comparison not done.	1000000000000000	‘1……’b

PROD	QC
ADAE	ADAE
	ADCM
ADIE	ADIE
ADLB	ADLB
ADMH	
ADSL	ADSL
ADVS	ADVS

Now create a dataset named **DICT.IDVARS** with all ID variables for each dataset as shown below.

DICT.IDVARS	
DSNAME	**IDVARS**
ADAE	SUBJID AESTDTC AEENDTC AEDECOD AETERM
ADCM	SUBJID PARCAT2N CMSTDTC CMENTDTC CMDECOD CMTRT CMINDC CMDOSTXT COMPSID
ADIE	SUBJID IECAT IETESTCD
ADLB	SUBJID PARCAT1N PARCAT2N PARAMN LBTYPE AVISITN ADTM VISITNUM LBREFID
ADMH	SUBJID PARCAT1N MHSTDTC MHENDTC MHTERM MHSPID
ADSL	SUBJID

Note that DICT.IDVARS does not contain the IDVARS for ADVS. It is intentionally left out to see what the result will be when the ID variables for a dataset has not been specified. Now by means of the macro **%QCDIR**, one can easily compare 2 datasets of the same name in PROD and QC directories using the ID variables provided in DICT.VARS, when available.

Below are the parameters based on the code from **%QCDIR** macro. Code of this macro can be found in the appendix.

```
%macro QCDIR (
  LIB_BASE=    /* Enter the LIBREF of the PRODUCTION datasets. (Required)  */
 ,LIB_COMPARE= /* Enter the LIBREF of the COMPARE datasets. (Required)     */
 ,IDVARS_DICT= /* Enter the name of the datasets containing data names     */
               /* and the key variables.(Optional, but Highly Recommended.)*/
 ,RC_REPORT=   /* Enter the name of the report containing all duplicate    */
               /* flags and RC flags.  If this parameter is blank,         */
               /* no report will be generated. (Optional, but recommended) */
)/des="To QC data, identify duplicate observations, output Return Codes and
       Duplicate Flags for all datasets in PROD and QC directories.";
```

Within **%QCDIR**, PROC CONTENTS is applied to find out all the dataset names in PROD and QC

directories. Combining these dataset names with data from **DICT.IDVARS**, a dataset named **ALLDS** can be produced.

One can then use the **ALLDS** dataset to invoke **%QCDATA** by means of **CALL EXECUTE** to check for duplications, perform comparison and generate comparison report for each pair of namesake datasets in the production and QC directories. Below are the code excerpted from **%QCDIR**. Moreover, **%QCDIR** can generate a summary report of the validation findings like the one seen below.

WORK.ALLDS

OBS	BASE	COMPARE	DSNAME	IDVARS
1	BASE.ADAE	QC.ADAE	ADAE	SUBJID AESTDTC AEENDTC AEDECOD AETERM
2		QC.ADCM	ADCM	SUBJID PARCAT2N CMSTDTC CMENTDTC CMDECOD CMTRT CMINDC CMDOSTXT COMPSID
3	BASE.ADIE	QC.ADIE	ADIE	SUBJID IECAT IETESTCD
4	BASE.ADLB	QC.ADLB	ADLB	SUBJID PARCAT1N PARCAT2N PARAMN LBTYPE AVISITN ADTM VISITNUM LBREFID
5	BASE.ADMH		ADMH	SUBJID PARCAT1N MHSTDTC MHENDTC MHTERM MHSPID
6	BASE.ADSL	QC.ADSL	ADSL	SUBJID
7	BASE.ADVS	QC.ADVS	ADVS	

Code Excerpted from %QCDIR

```
data _null_;
  retain refresh_flag;
  set ALLDS;
  *--- Only refresh the ALLRC in the first observation. ---*;
  if _n_=1 then refresh_flag=0;
  if refresh_flag=0 then do;
     refresh_flag=1;
     CALL EXECUTE('%NRSTR(%QCDATA(BASE='||BASE||',COMPARE='||COMPARE||
            ',IDVARS='||IDVARS||',OUTRC=ALLRC,REFRESH=Y));'); end;
  else if refresh_flag=1 then do;
     CALL EXECUTE('%NRSTR(%QCDATA(BASE='||BASE||',COMPARE='||COMPARE||
                  ',IDVARS='||IDVARS||',OUTRC=ALLRC,REFRESH=N);'); end;
  run;
```

One very important thing that worth pointing out is that the authors have used CALL EXECUTE (%NRSTR(…)), instead of simply CALL EXECUTE(…). This is because CALL EXECUTE executes immediately and in this example, the program will run with simply CALL EXECUTE, but incorrect results will be generated. CALL EXECUTE(%NRSTR(…)) delays the execution and produces the correct result. Interested readers can consult H. Ian Whitlock's SUGI 22 paper.

In addition to the typical PROC COMPARE output, %QCDIR also produces a summary as in Figure 5. It is a report that summarizes the comparison reason of all datasets in production and QC directories. Note that in this example, the program will have an error because there is a fatal error in PROD.ADLB vs QC.ADLB comparison. Based on the report, one knows that ADLB comparison has a fatal error (RC_16='Y') and because RC_14='Y', one knows that the error is a result of conflicting variable types. In fact, because it is a fatal error, that variable with conflicting type has to be an ID variable!

Comments: Traffic-lighting has been applied in the summary report to indicate the degree of seriousness

of the discrepancies. For instance, GREEN for 100% matched (i.e. RC=0); YELLOW when for minor discrepancies, such as discrepancy in attributes (i.e., RC < 64); ORANGE for more significant discrepancies, such as values not match in some variables (i.e., 65 <= RC < 32768); RED for Fatal Error (i.e., RC >= 32768); PURPLE when a dataset of that name exists in only one dataset and hence, comparison is not possible. This technique of traffic lighting to report PROC COMPARE results has been discussed in Conover (2013) and Carpenter (2007).

Figure 5: Comparison of Datasets in Production vs. Validation Directories

Base	Compare	IDVARS	DUPB	DUPC
	QC.ADCM	SUBJID PARCAT1N CMSTDTC CMENDTC CMDECOD CMTRT CMINDC CMDOSTXT CMSPID	NA	N
PROD.ADAE	QC.ADAE	SUBJID AESTDTC AEENDTC AEDECOD AETERM	N	N
PROD.ADIE	QC.ADIE	SUBJID IECAT IETESTCD	N	N
PROD.ADLB	PROD.ADLB	SUBJID PARCAT1N PARCAT2N PARAMN LBTYPE AVISITN ADTM VISITNUM LBREFID	N	N
PROD.ADMH		SUBJID PARCAT1N MHSTDTC MHENDTC MHTERM MHSPID	N	NA
PROD.ADSL	QC.ADSL	SUBJID	Y	N
PROD.ADVS	QC.ADVS		NA	NA

Base	Compare	RC	RC_0	RC_1	RC_2	RC_3	RC_4	RC_5	RC_6	RC_7	RC_8	RC_9	RC_10
	QC.ADCM	NA											
PROD.ADAE	QC.ADAE	48						Y	Y				
PROD.ADIE	QC.ADIE	0	Y										
PROD.ADLB	PROD.ADLB	40960											
PROD.ADMH		NA											
PROD.ADSL	QC.ADSL	5357		Y		Y	Y		Y	Y	Y		
PROD.ADVS	QC.ADVS	5164				Y	Y		Y				

Base	Compare	RC_11	RC_12	RC_13	RC_14	RC_15	RC_16
	QC.ADCM						
PROD.ADAE	QC.ADAE						
PROD.ADIE	QC.ADIE						
PROD.ADLB	PROD.ADLB				Y		Y
PROD.ADMH							
PROD.ADSL	QC.ADSL	Y		Y			
PROD.ADVS	QC.ADVS	Y		Y			

CONCLUSION

In this paper, the authors have provided a brief introduction to the COMPARE procedure. They have introduced tips and techniques to speed up the validation process. The paper can be actually divided into 2 parts: without automation and with automation. In the without automation sections, standard and clever techniques have been introduced. These techniques help users to quickly identify the source of the discrepancies. In the with automation portion of this paper, the authors demonstrate the use of %QCDATA to easily check for duplicated observations based on ID variables and from the return code (&SYSINFO) generated by the COMPARE procedure, confirm if the comparison is a 100% match (&SYSINFO=0) or if the comparison has one or more of the 16 possible problematic conditions from PROC COMPARE. They then further extended the automation process by means of the macro %QCDIR to automatically compare datasets from two different directories. *"How to Speed Up Your Validation Process Without Really Trying"* Ok, maybe users do need to give comparison a try, but hopefully, armed with these techniques, the trying part has reduced and the validation process does speed up! Last but not the least, to borrow Christianna Williams' last words in her 2010 paper, the authors wish you all *Happy Compare-ing*!

REFERENCES

Carpenter, Art, 2004, "Carpenter's Complete Guide to SAS9® Macro Language, 2nd Edition", SAS Institute Inc., Cary, NC, USA.

Carpenter, Art, 2007 "Advanced PROC REPORT: Traffic Lighting-Controlling Cell Attributes With Your Data", PharmaSUG 2007. http://www.sascommunity.org/wiki/Advanced_PROC_REPORT:_Traffic_Lighting_-_Controlling_Cell_Attributes_With_Your_Data

Carpenter, Art, 2012, "Carpenter's Guide to Innovative SAS® Techniques", SAS Institute Inc., Cary, NC, USA.

Conover, William, "An Abbreviated PROC COMPARE with Traffic Lighting", WUSS 2013. http://wuss.org/Proceedings13/91_Paper.pdf

Horstman, Joshua and Muller, Roger, "Don't Get Blindsided by PROC COMPARE", PharmaSUG 2013. http://www.pharmasug.org/proceedings/2013/CC/PharmaSUG-2013-CC36.pdf

SAS Institute Inc. KNOWLEDGE BASE/SAMPLES & SAS Notes. Sample *45011:* Using return codes from automatic macro variable &SYSINFO with PROC COMPARE. http://support.sas.com/kb/45/011.html

SAS Institute Inc. 2015. The COMPARE Procedure. Base SAS® 9.4 Procedures Guide. Cary, NC: SAS Institute Inc. http://support.sas.com/documentation/cdl/en/proc/68954/HTML/default/viewer.htm#n0c1y14wyd3u7yn1dmfcpaejllsn.htm

Whitlock, H. Ian, "Call Execute: How and Why", SUGI 22. http://www2.sas.com/proceedings/sugi22/CODERS/PAPER70.PDF

Williams, Christianna S., "PROC COMPARE – Worth Another Look! ", SAS Global Forum 2010. http://support.sas.com/resources/papers/proceedings10/149-2010.pdf

ACKNOWLEDGMENTS

The author, Alice Cheng would like to express her sincere gratitude to Mr. Art Carpenter of CALOXY, Ms. Jane Eslinger, Mr. Kevin Russell and Kathryn McLawhorn of SAS® Technical Support and Mr. Tom Brotz of Chiltern International for sharing their expertise in SAS and validation in clinical studies. She would like to thank you her manager, Carey Smoak of Portola Pharmaceuticals, Inc. for his encouragement and her former manager, Mr. Boyd Roloff of Chiltern International for his guidance and support, as well as, the numerous validation assignments, without which this paper would not have come into fruition. The authors, Michael R. Wise and Justina M. Flavin would like to thank their respective companies, Experis and SimulStat, Inc. for their support.

CONTACT INFORMATION

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

Alice M. Cheng
Portola Pharmaceuticals, South San Francisco, CA
E-mail: acheng@portala.com , alice_m_cheng@yahoo.com
Web: https://www.linkedin.com/in/alicemcheng

Michael R. Wise
Experis, San Diego, CA
E-mail: mike.wise@experis.com

Justina Flavin
SimulStat, Inc., San Diego, CA
E-mail: Justina.flavin@gmail.com

SLIDE PRESENTATION

Slide presentation of this paper will be made available in https://www.linkedin.com/in/alicemcheng or http://www.sascommunity.org/wiki/Presentations:Alicemcheng_Papers_and_Presentations

APPENDIX

The definition of %QCDATA, %QCDIR and %NOBS (which is invoked by %QCDATA) can be found in the following Dropbox links: https://www.dropbox.com/home?d=1&preview=Validation+Macros_v4.pdf or http://www.sascommunity.org/wiki/Presentations:Alicemcheng_Papers_and_Presentations