Paper PP19

A way out of the PROC COMPARE labyrinth

Judith Kanz, mainanalytics GmbH, Sulzbach, Germany

ABSTRACT

Outputs for the various analyses need to be programmed while the trial is ongoing. But validation cannot wait until final data is available either. Double programming of the input dataset for the reporting procedure usually is the method of choice as validation can easily be repeated on updated data by executing the available validation programs.

This poster shows not only how to facilitate programmatic comparison but also how to digest all information provided by the PROC COMPARE procedure step by step.

INTRODUCTION

Since I started in the pharmaceutical industry 30 years ago, the significance of validation has seen a remarkable increase. This heightened awareness has led to validation being given a very high priority in the creation process of analysis datasets and outputs. The time between the availability of final data and delivery of final outputs is usually rather short, the validation cannot wait for any database lock and should start at an early state. Moreover, draft outputs will probably be requested by statisticians or medical writers a long time before final delivery. The tables, listings and graphs delivered at this timepoint should already be in a good shape.

As the validation process is repeated very often, with limited time for the final check, it is recommended to do this by double programming. The programmatic comparison of output files is difficult and even impossible for graphs, besides that it would take an enormous effort to match the table layout completely. A practical way is the double programming of the input dataset used in the reporting procedure. In general, the comparison is performed with PROC COMPARE. The output of this procedure can be overwhelming and often it is not easy to find the source of the discrepancies and figure out what is essential for the underlying output.

This poster shall show my daily work validation approach, how to go through a PROC COMPARE example step by step and how the process can be supported by a macro.

EXAMPLE OUTPUT

Example outputs of the PROC COMPARE will be shown for the validation of an adverse event frequency table. The first page of this table looks as follows:

mainanalytics GmbH
Table 6.2: Adverse event frequencies by subgroup
Subgroup Sex: Male

System organ class/ Preferred term	Placebo N	%	95% CI	Time at risk [patient years]	Incidence rate/100 patient years	Xanomeline N	%	95% CI	Time at risk [patient years]	Incidence rate/100 patient years
Number of patients	33	100.0		14.5		78	100.0		20.4	
Number of patients with TEAE	25	75.8	[59.0, 87.2]	5.8	432.6	73	93.6	[85.9, 97.2]	4.6	1602.4
Cardiac disorders	3	9.1	[3.1, 23.6]	13.6	22.0	16	20.5	[13.0, 30.8]	18.7	85.4
ATRIAL FIBRILLATION	0	0.0	[0.0, 10.4]	14.5	0.0	2	2.6	[0.7, 8.9]	20.4	9.8
ATRIAL FLUTTER	0	0.0	[0.0, 10.4]	14.5	0.0	2	2.6	[0.7, 8.9]	20.3	9.9
ATRIAL HYPERTROPHY	1	3.0	[0.5, 15.3]	14.1	7.1	0	0.0	[0.0, 4.7]	20.4	0.0
ATRIOVENTRICULAR BLOCK FIRST DEGREE	0	0.0	[0.0, 10.4]	14.5	0.0	1	1.3	[0.2, 6.9]	20.2	5.0
ATRIOVENTRICULAR BLOCK SECOND DEGREE	1	3.0	[0.5, 15.3]	14.5	6.9	0	0.0	[0.0, 4.7]	20.4	0.0
BRADYCARDIA	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
BUNDLE BRANCH BLOCK LEFT	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
BUNDLE BRANCH BLOCK RIGHT	0	0.0	[0.0, 10.4]	14.5	0.0	1	1.3	[0.2, 6.9]	20.4	4.9
CARDIAC DISORDER	0	0.0	[0.0, 10.4]	14.5	0.0	1	1.3	[0.2, 6.9]	20.4	4.9
CARDIAC FAILURE CONGESTIVE	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
MYOCARDIAL INFARCTION	0	0.0	[0.0, 10.4]	14.5	0.0	3	3.8	[1.3, 10.7]	19.8	15.1
PALPITATIONS	0	0.0	[0.0, 10.4]	14.5	0.0	1	1.3	[0.2, 6.9]	20.4	4.9
SINUS ARRHYTHMIA	1	3.0	[0.5, 15.3]	14.1	7.1	0	0.0	[0.0, 4.7]	20.4	0.0
SINUS BRADYCARDIA	0	0.0	[0.0, 10.4]	14.5	0.0	10	12.8	[7.1, 22.0]	19.8	50.6
SUPRAVENTRICULAR EXTRASYSTOLES	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
SUPRAVENTRICULAR TACHYCARDIA	0	0.0	[0.0, 10.4]	14.5	0.0	1	1.3	[0.2, 6.9]	20.2	5.0
TACHYCARDIA	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
VENTRICULAR EXTRASYSTOLES	0	0.0	[0.0, 10.4]	14.5	0.0	2	2.6	[0.7, 8.9]	20.1	10.0
VENTRICULAR HYPERTROPHY	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
WOLFF-PARKINSON-WHITE SYNDROME	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
Congenital, familial and genetic disorders	0	0.0	[0.0, 10.4]	14.5	0.0	2	2.6	[0.7, 8.9]	20.3	9.8
VENTRICULAR SEPTAL DEFECT	0	0.0	[0.0, 10.4]	14.5	0.0	2	2.6	[0.7, 8.9]	20.3	9.8
Ear and labyrinth disorders	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
CERUMEN IMPACTION	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
EAR PAIN	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0
VERTIGO	0	0.0	[0.0, 10.4]	14.5	0.0	0	0.0	[0.0, 4.7]	20.4	0.0

Source data: CDISC Pilot
Date/time of run: 12NOV2024 17:30 t-aefreq.sas / t_002_aefreq.lst

Figure 1: Example table

The dataset that was used for PROC REPORT is saved as a permanent file called PRODDS in this paper. To enable as many automatic checks as possible the labels and formats used for the procedure shall be already assigned to the variables in this dataset rather than defining them in the PROC REPORT define statement.
The table above was created with the following PROC REPORT:

```
proc report data=dstlf.&outid nowd headline headskip missing split="$" spacing=2;
  column subgroupn subgroup order aesoc aedecod text treat,
         (freq perc ci trisk rate) dummy;
  define subgroupn / group id order=data noprint;
  define subgroup / group id order=data noprint;
  define order / group id order=data noprint;
  define aesoc / group id order=data noprint;
  define aedecod / group id order=data noprint;
  define text / group id order=data width=55 flow spacing=0;
  define treat / " " order=data across;
  define freq / display;
  define perc / display;
  define trisk / display;
  define rate /  display;
  define dummy / noprint;
  break after aesoc / skip;
  compute before _page_;
    line @1 subgroup $50.;
  endcomp;
run;
```

Figure 2: PROC REPORT statement to produce example table

VALIDATION STEPS

- Before starting with the validation process make sure that the output is created with the latest program version and on the most current data.
- Check if the log file of the main program is clear as it is not reasonable to compare datasets which may not be created properly.
- An output review shall be done in the beginning to make sure that the required information is available and displayed correctly.
- In addition, a source code review of the reporting procedure is necessary as this step is beyond the part that can be handled by double programming.
- Once all the above steps were successfully performed create a dataset matching the input dataset of the reporting procedure. This validation dataset is called QCDS.

CONSIDERATIONS

For output validation generally it is not necessary to create a completely matching QCDS. It is sufficient to focus on what is really needed for the respective output:

- Which variables included in PRODDS are used to produce the output? This can either be variables where the values are displayed, or variables used for sorting. Often the dataset also contains variables used in the derivation process but not needed in the end.
- Are variable labels or formats used directly for the output?
- How many digits are shown in the output? Does it make sense to restrict the comparison that way?
- Are indentations or other spaces relevant for PROC COMPARE or would it be sufficient to check this per output review taking indentations or spaces into account
- Furthermore, it should be checked if there are any identifying variables in PRODDS which can be used as ID variables in the PROC COMPARE procedure. This helps to find the location of discrepancies.

SUPPORTING MACRO

For the comparison between PRODDS and QCDS a macro will be used to perform the PROC COMPARE directly in the validation program, taking the considerations mentioned above into account.
The macro is defined with the following macro parameters:

Macro QC_COMPARE

Parameter	Default	Description
OUTID		Output ID for TLF comparison
PRODDS		ADS name for ADS comparison
QCDS	out	Name of QC dataset
KEY	_none_	Variable names for PROC COMPARE ID (also possible: _none_)
KEEP	_all_	Variable names taken into account for PROC COMPARE (also possible: _all_)
DROP	_none_	Variable names removed for PROC COMPARE (also possible: _none_)
INDENT	_none_	Variable names for comparison of indentations (also possible: _all_/_none_)
COMPRESS	_none_	Variable names to be compressed before PROC COMPARE (also possible: _all_/_none_)
CRITERION	0.000001	Value of CRITERION option used for PROC COMPARE
LABELS	_all_	Variable names for comparison of labels (also possible: _all_/_none_)
FORMATS	_all_	Variable names for comparison of formats (also possible: _all_/_none_)
MODIFY		Dataset statements for PRODDS modifications

The macro creates temporary datasets from PRODDS and QCDS with the required modifications. Both datasets contain only variables as specified with the parameters KEEP and DROP and only relevant labels and formats are kept. If the check for indentations or blanks shall not be performed, or further changes are required, the respective variables are modified accordingly. Both datasets are sorted according to the variable list specified with macro parameter KEY. The PROC COMPARE is conducted with the option LISTALL, the option CRITERION with the value as specified in parameter CRITERION and with the key variables in the ID statement, if applicable.

It is recommended to set up the environment in a way that the output file name and the PRODDS name can automatically be retrieved from the output ID. This can be for example handled by a dataset containing the meta information of all outputs including the IDs and the output file names, and so the macro can also check if the output file is available. As a timestamp check of the output file is no guarantee that the output is created by the last run, this check is not implemented. The best way to avoid outdated files being kept is to remove all existing output files before running the main programs.

In addition to the PROC COMPARE performance the macro creates two temporary datasets to support the search for the sources of discrepancies.

Temporary dataset 1: MCCOMP_CHECKKEY to check for discrepancies in ID variables
This dataset supports the check of non-matching IDs. It contains all ID variables plus two additional flag variables PROD and QC to identify from which dataset the record is coming from. This dataset can be helpful to find a pattern in the discrepancies. The dataset contains all original records from both datasets. It might be useful to see all values together or restrict the focus to specific entries only to find a pattern in the discrepancies.
This dataset can for example be used for discrepancies in very large lab outputs as it is easier to see here if a parameter is dropped completely or for some visits only than in the PROC COMPARE output itself.

Temporary dataset 2: MCCOMP_CHECKVARS to check for discrepancies in non-ID variables
Again, all records are included. This dataset contains all ID variables first, and afterwards all non-ID variables from PRODDS and QCDS side by side with prefixes PROD_ and QC_. Moreover, the flag variables PROD and QC are added, and in addition the variable DIFF_PROD_QC is created. This variable flags records with any discrepancy and is created with respect to the value in macro parameter CRITERION, so only the number of relevant digits is taken into account.
This dataset could for example be helpful for PROC COMPAREs with more ID variables than printed in the PROC COMPARE output. Furthermore, it might help to find a pattern in discrepancies if only specific entries are affected.

All issues found by the macro are also printed in the log file with the keywords ERROR or WARNING as shown in the screenshot below. If a re-run is done due to new data being available it is not necessary to check all PROC COMPARE outputs manually again, but search for these keywords in the log files, which is something that should be done anyway. All messages contain either the output or the dataset name, which makes it possible to identify the problematic output in case a program produces multiple outputs.

```
ERROR: output /data/projects/sandbox/sandbox_1/05-Prog-Env/jkanz/outtlf/t_002_aefreq.lst does not exist
ERROR: QC of DSTLF.T_002_AEFREQ - Dataset DSTLF.T_002_AEFREQ does not exist

WARNING: QC of DSTLF.T_002_AEFREQ - variable DUMMY in PROD dataset only
WARNING: QC of DSTLF.T_002_AEFREQ - variable ORDER in PROD dataset only

WARNING: QC of DSTLF.T_002_AEFREQ - variable PERC with conflicting types

WARNING: QC of DSTLF.T_002_AEFREQ - variable CI with different labels
WARNING: QC of DSTLF.T_002_AEFREQ - variable FREQ with different labels
WARNING: QC of DSTLF.T_002_AEFREQ - variable PERC with different labels
WARNING: QC of DSTLF.T_002_AEFREQ - variable RATE with different labels
WARNING: QC of DSTLF.T_002_AEFREQ - variable TREAT with different labels
WARNING: QC of DSTLF.T_002_AEFREQ - variable TRISK with different labels

WARNING: QC of DSTLF.T_002_AEFREQ - values for key variables differ

WARNING: QC of DSTLF.T_002_AEFREQ - values differ
```

Figure 3: Log file messages

Here the usage of the macro for an output validation is presented only. The validation of analysis datasets (ADS) is more straight forward than for outputs, as the QC approach is to create a complete duplicate version of the dataset. In this case the macro can be supportive as well, but with different rules. It is not possible to restrict the variables by using parameters KEEP and DROP as the variables must match completely. Accordingly, suppressing the label or format check as well as data modifications with parameters INDENT, COMPRESS or MODIFY are disabled. A key must be provided. For ADS comparisons two additional checks are conducted. The macro checks if the ADS is sorted according to the key variables, and if the order of the variables in the datasets is matching. It is assumed that the QCDS is created according to the dataset specifications and the comparison shall be done with the key variables provided there.

PROC COMPARE OUTPUTS

This section will show some examples of PROC COMPARE outputs and explain how to handle these with the help of the macro.

EXAMPLE 1

```
Listing of Variables in WORK.MCCOMP_PRODDS but not in WORK.MCCOMP_QCDS

Variable  Type  Length

ORDER     Num         8
DUMMY     Num         8
```

Figure 4: PROC COMPARE output example 1 - listing of variables not in both datasets

The first part of the PROC COMPARE output shows variables that are only included in one of the datasets. The use of these variables in the output procedure should be checked via source code review to decide if they need to be compared or not. In this case both variables are included in the PROC REPORT statement, but not printed directly. Variable ORDER controls the order of the rows in the output, so it is recommended to include it in the comparison. Variable DUMMY is often added in PROC REPORT when ACROSS is used to control the layout of the output. As this variable is not printed it might be skipped for the comparison.
The list of variables used for PROC COMPARE can be controlled by the macro parameters KEEP and DROP.

EXAMPLE 2

```
Listing of Common Variables with Conflicting Types

Variable  Dataset              Type  Length  Format  Label

PERC      WORK.MCCOMP_PRODDS   Num        8  5.1     %
          WORK.MCCOMP_QCDS     Char       5
```

Figure 5: PROC COMPARE output example 2 - listing of common variables with conflicting types

The second part of the PROC COMPARE output shows variables with conflicting types, so the same variable is numeric in one dataset and character in the other. This means the values of these variables cannot be compared. Usually, it makes sense to update the validation program.

EXAMPLE 3

Listing of Common Variables with Differing Attributes

Variable	Dataset	Type	Length	Format	Label
SUBGROUP	WORK.MCCOMP_PRODDS	Char	45		
	WORK.MCCOMP_QCDS	Char	50		
TREAT	WORK.MCCOMP_PRODDS	Char	100		Treatment
	WORK.MCCOMP_QCDS	Char	100		
TRISK	WORK.MCCOMP_PRODDS	Num	8	8.1	Time at risk [patient years]
	WORK.MCCOMP_QCDS	Num	8		the sum, RISKYEARS
CI	WORK.MCCOMP_PRODDS	Char	15		95% CI
	WORK.MCCOMP_QCDS	Char	15		
FREQ	WORK.MCCOMP_PRODDS	Num	8	3.	N
	WORK.MCCOMP_QCDS	Num	8		
RATE	WORK.MCCOMP_PRODDS	Num	8	9.1	Incidence rate/100 patient years
	WORK.MCCOMP_QCDS	Num	8		
TEXT	WORK.MCCOMP_PRODDS	Char	200		System organ class/$ Preferred term
	WORK.MCCOMP_QCDS	Char	250		System organ class/$ Preferred term

Figure 6: PROC COMPARE output example 3 - listing of common variables with differing attributes

The third part of the output lists variables with differences in the attributes label, format, informat and length.

It is important to compare labels within the PROC COMPARE if they are directly used by the output procedure, otherwise this comparison might be skipped. This can be controlled by the macro parameter LABELS.

In case a variable is attributed with a permanent format that is used in the procedure it should also be included in the comparison. This is handled by macro parameter FORMATS. If a relevant temporary format is assigned, the definition of the format has to be checked per source code review. In addition, it would be good to have a decode variable in PRODDS to allow automated checks, even if it is not used for output creation.

Informats are not relevant for the output procedure and are dropped completely from the comparison.

The macro does not provide the possibility to suppress discrepancies in length statements. For some variables it makes sense to have a greater length in QCDS to ensure that values are not truncated. A workaround to get rid of this message is to create a longer variable in the QC program first and check if the length of the variable is sufficient. If this is not the case create a warning message in the log file. Afterwards create a variable with the name and length as used in PRODDS.

EXAMPLE 4

```
Comparison Results for Observations

Observation 25 in WORK.MCCOMP_QCDS not found in WORK.MCCOMP_PRODDS:
SUBGROUPN=11 TEXT=  APPLICATION SITE DESQUAMATION TREAT=_Placebo_.

Observation 26 in WORK.MCCOMP_QCDS not found in WORK.MCCOMP_PRODDS:
SUBGROUPN=11 TEXT=  APPLICATION SITE DESQUAMATION TREAT=_Xanomeline_.

Observation 37 in WORK.MCCOMP_QCDS not found in WORK.MCCOMP_PRODDS:
SUBGROUPN=11 TEXT=  APPLICATION SITE PERSPIRATION TREAT=_Placebo_.

Observation 38 in WORK.MCCOMP_QCDS not found in WORK.MCCOMP_PRODDS:
SUBGROUPN=11 TEXT=  APPLICATION SITE PERSPIRATION TREAT=_Xanomeline_.

Observation 47 in WORK.MCCOMP_PRODDS not found in WORK.MCCOMP_QCDS:
SUBGROUPN=11 TEXT=  APPLICATION SITE$   DESQUAMATION TREAT=_Placebo_.

Observation 48 in WORK.MCCOMP_PRODDS not found in WORK.MCCOMP_QCDS:
```

Figure 7: PROC COMPARE output example 4 - comparison results for observations

The next output part is about observations with non-matching ID variables. They are shown if the PROC COMPARE option LISTALL is specified, but as they are printed in order of appearance the records from PRODDS and QCDS are mixed. Therefore, it can be difficult to find a pattern, especially in very large outputs or for a lot of ID variables or if there are multiple reasons leading to mismatches.

To facilitate the check of non-matching IDs the first temporary dataset MCCOMP_CHECKKEY can be used.

WORK.MCCOMP_CHECKKEY

View: Column names | Filter: prod ne qc

Columns: Select all, SUBGROUPN, TREAT, TEXT, PROD, QC

Property | Value: Label

Total rows: 510 Total columns: 5 Filtered rows: 180 Rows 1-180

	SUBGROUPN	TREAT	TEXT	PROD	QC
1	11	_Placebo_	APPLICATION SITE DESQUAMATION		1
2	11	_Xanomeline_	APPLICATION SITE DESQUAMATION		1
3	11	_Placebo_	APPLICATION SITE PERSPIRATION		1
4	11	_Xanomeline_	APPLICATION SITE PERSPIRATION		1
5	11	_Placebo_	APPLICATION SITE$ DESQUAMATION	1	
6	11	_Xanomeline_	APPLICATION SITE$ DESQUAMATION	1	
7	11	_Placebo_	APPLICATION SITE$ PERSPIRATION	1	
8	11	_Xanomeline_	APPLICATION SITE$ PERSPIRATION	1	
9	11	_Placebo_	ATRIOVENTRICULAR BLOCK SECOND$ DEGREE		1
10	11	_Xanomeline_	ATRIOVENTRICULAR BLOCK SECOND$ DEGREE		1
11	11	_Placebo_	ATRIOVENTRICULAR BLOCK$ SECOND DEGREE	1	

Figure 8: temporary dataset MCCOMP_CHECKKEY to show differences in ID variables

The dataset was filtered for entries with discrepancies, this means records that are either included in PRODDS or in QCDS only were selected. It shows that the split characters and additional blanks to force the indentation of the following line in the table are not done at the same place. As the datasets provides the chance to see the respective entries one below the other it is easier to find the correct algorithm done on the production side.

In case it turns out that the search for the root cause is very time consuming another solution to handle this discrepancy can be chosen. The macro provides also the possibility to modify entries in PRODDS using the macro parameter MODIFY. With this option it is possible to remove split characters and check the correct appearance per output review.

EXAMPLE 5

Value Comparison Results for Variables

SUBGROUPN	TEXT	TREAT	Base Value SUBGROUP	Compare Value SUBGROUP
36	ABDOMINAL DISCOMFO	_Xanomeline_	Subgroup Race: Ameri	Subgroup Race: Ameri
36	ABDOMINAL PAIN	_Xanomeline_	Subgroup Race: Ameri	Subgroup Race: Ameri
36	ACROCHORDON EXCISI	_Xanomeline_	Subgroup Race: Ameri	Subgroup Race: Ameri
36	ACTINIC KERATOSIS	_Xanomeline_	Subgroup Race: Ameri	Subgroup Race: Ameri
36	AGITATION	_Xanomeline_	Subgroup Race: Ameri	Subgroup Race: Ameri
36	ALCOHOL USE	_Xanomeline_	Subgroup Race: Ameri	Subgroup Race: Ameri

Figure 9: PROC COMPARE output example 5 - value comparison results for variable SUBGROUP

The last part of the output covers discrepancies in non-ID variables. If mismatches occur after the twentieth character, they are not visible in the PROC COMPARE output itself. In this case the second temporary dataset MCCOMP_CHECKVARS can be used. Both variables can be seen side by side, containing the complete contents.

WORK.MCCOMP_CHECKVARS

View: Column names | Filter: prod=1 and qc=1 and prod_subgroup ne qc_subgroup

Columns: Select all, SUBGROUPN, TEXT, TREAT, PROD_SUBGROUP, QC_SUBGROUP, PROD_TREATN, QC_TREATN

Total rows: 3915 Total columns: 32 Filtered rows: 249 Rows 1-2

	PROD_SUBGROUP	QC_SUBGROUP
1	Subgroup Race: American Indian or Alaska Nati	Subgroup Race: American Indian or Alaska Native
2	Subgroup Race: American Indian or Alaska Nati	Subgroup Race: American Indian or Alaska Native
3	Subgroup Race: American Indian or Alaska Nati	Subgroup Race: American Indian or Alaska Native
4	Subgroup Race: American Indian or Alaska Nati	Subgroup Race: American Indian or Alaska Native
5	Subgroup Race: American Indian or Alaska Nati	Subgroup Race: American Indian or Alaska Native
6	Subgroup Race: American Indian or Alaska Nati	Subgroup Race: American Indian or Alaska Native
7	Subgroup Race: American Indian or Alaska Nati	Subgroup Race: American Indian or Alaska Native

Figure 10: temporary dataset MCCOMP_CHECKVARS to show differences in variable SUBGROUP

EXAMPLE 6

Value Comparison Results for Variables

SUBGROUPN	TEXT	TREAT	95% CI Base Value CI	Compare Value CI
11	ABDOMINAL DISCOMFO	_Placebo_	[0.0, 10.4]	[0.0, 10.4]
11	ABDOMINAL DISCOMFO	_Xanomeline_	[0.2, 6.9]	[0.2, 6.9]
11	ABDOMINAL PAIN	_Placebo_	[0.0, 10.4]	[0.0, 10.4]
11	ABDOMINAL PAIN	_Xanomeline_	[0.2, 6.9]	[0.2, 6.9]
11	ACROCHORDON EXCISI	_Placebo_	[0.0, 10.4]	[0.0, 10.4]
11	ACROCHORDON EXCISI	_Xanomeline_	[0.0, 4.7]	[0.0, 4.7]
11	ACTINIC KERATOSIS	_Placebo_	[0.0, 10.4]	[0.0, 10.4]
11	ACTINIC KERATOSIS	_Xanomeline_	[0.2, 6.9]	[0.2, 6.9]
11	AGITATION	_Placebo_	[0.0, 10.4]	[0.0, 10.4]
11	AGITATION	_Xanomeline_	[0.0, 4.7]	[0.0, 4.7]

Figure 11: PROC COMPARE output example 6 - value comparison results for variable CI

The above example is about discrepancies in confidence intervals. It appears the discrepancies are simply additional blanks within the interval in the QC variable. Therefore, it is a reasonable option to remove these spaces from the compare and only check them during the output review. If the check concentrates on the values only by using the macro parameter COMPRESS it is possible to even use a bigger format on the values in QCDS and so it can be checked if the format used in PRODDS is sufficient.

EXAMPLE 7

All Variables Compared have Unequal Values

Variable	Type	Len	Label	Compare Label	Ndif	MaxDif
TRISK	NUM	8	Time at risk [patient years]	the sum, RISKYEARS	39	0.0082
RATE	NUM	8	Incidence rate/100 patient years		39	1565

Figure 12: PROC COMPARE output example 7 - listing of variables with unequal values

This example shows discrepancies in variables for time at risk and for the incidence rate. In this case it makes sense to first check for dependencies between variables. As time at risk is used to calculate the incidence rate it is reasonable to focus on the mismatches in time at risk first. It might be that the discrepancies for the incidence rate disappear automatically once time at risk matches, but for sure there is no chance to get a match if the underlying variables differ.

Value Comparison Results for Variables

			Time at risk [patient years] the sum, RISKYEARS			
SUBGROUPN	TEXT	TREAT	Base TRISK	Compare TRISK	Diff.	% Diff
11	COUGH	_Xanomeline_	19.2	19.2498	0.008214	0.0427
11	ERYTHEMA	_Xanomeline_	16.1	16.1205	0.008214	0.0510
11	NASAL CONGESTION	_Xanomeline_	19.3	19.3155	0.008214	0.0425
11	NAUSEA	_Xanomeline_	19.9	19.9042	0.008214	0.0413
11	PAROSMIA	_Xanomeline_	20.2	20.1725	0.008214	0.0407
11	PRURITUS	_Xanomeline_	15.5	15.4935	0.008214	0.0530
11	SALIVARY HYPERSECR	_Xanomeline_	19.7	19.6660	0.008214	0.0418
11	VOMITING	_Xanomeline_	18.9	18.9131	0.008214	0.0434
11	Gastrointestinal dis	_Xanomeline_	16.4	16.3888	0.008214	0.0501
11	Nervous system disor	_Xanomeline_	16.1	16.0903	0.008214	0.0511
11	Number of patients w	_Xanomeline_	4.6	4.5640	0.008214	0.1803
11	Respiratory, thoraci	Xanomeline	18.1	18.1410	0.008214	0.0453

Figure 13: PROC COMPARE output example 7 - value comparison results for variable TRISK

On first sight the source of the difference cannot be detected, and it is necessary to have a closer look at the data. To find a starting point for the search for the root cause the second temporary dataset MCCOMP_CHECKVARS can be used.

WORK.MCCOMP_CHECKVARS

View: Column names | Filter: diff_prod_qc=1 and substr(text,1,1)=" "

Columns: Select all, SUBGROUPN, TEXT, TREAT, PROD_SUBGROUP, QC_SUBGROUP, PROD_TREATN, QC_TREATN, PROD_AESOC, QC_AESOC, PROD_AEDECOD, QC_AEDECOD, PROD_COUNT, QC_COUNT

Total rows: 3825 Total columns: 32 Filtered rows: 24 | Rows 1-24

	SUBGROUPN	TEXT	TREAT	PROD_COUNT	DIFF_PROD_QC
1	11	COUGH	_Xanomeline_	6	1
2	11	ERYTHEMA	_Xanomeline_	13	1
3	11	NASAL CONGESTION	_Xanomeline_	4	1
4	11	NAUSEA	_Xanomeline_	3	1
5	11	PAROSMIA	_Xanomeline_	1	1
6	11	PRURITUS	_Xanomeline_	25	1
7	11	SALIVARY HYPERSECRETION	_Xanomeline_	3	1
8	11	VOMITING	_Xanomeline_	7	1
9	21	COUGH	_Xanomeline_	2	1
10	21	ERYTHEMA	_Xanomeline_	4	1
11	21	NASAL CONGESTION	_Xanomeline_	2	1
12	21	NAUSEA	_Xanomeline_	1	1
13	21	PAROSMIA	_Xanomeline_	1	1
14	21	PRURITUS	_Xanomeline_	5	1

Figure 14: temporary dataset MCCOMP_CHECKVARS prepared to find the source of differences in variable TRISK

Records with discrepancies were selected, and in addition entries for preferred terms were kept. As the dataset contains all variables, the variable containing the number of patients with event can be checked. Some events with one occurrence only are included in the selection. So one of these events can be picked and the corresponding event can easily be selected from the underlying ADaM dataset. A manual check of the results can be performed.

EXAMPLE CALL

During the validation process the macro call for comparison is adapted several times according to the current results. For the example above, in the end the call could look like this:

```
%qc_compare(outid     = t_002_aefreq
           ,key       = subgroupn text treat
           ,drop      = dummy
           ,compress  = ci
           ,labels    = text freq perc ci trisk rate treat
           ,formats   = text freq perc ci trisk rate treat
           ,modify    = %str(text=tranwrd(text,"$   "," ");)
           );
```

Figure 15: Example call of macro QC_COMPARE

FURTHER INVESTIGATIONS

If there are still discrepancies left the common approach to identify the source needs to be followed. One possibility is to perform a source code review. In case the program contains a lot of macros a log file review can be done using options MPRINT, MLOGIC or SYMBOLGEN. Another option is to run the main program interactively and check temporary datasets. Alternatively, to avoid looking into the production program the programmer can be asked to provide a dataset with the respective calculations on patient level that are done in the main program and compare this one against the corresponding dataset on QC level.

CONCLUSION

You can ease your daily work. The macro cannot provide a solution to get rid of all discrepancies, but it can help you to get closer to finding a pattern or a hint for the source of mismatches and can speed things up.

CONTACT INFORMATION

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

Author Name: Judith Kanz
Company: mainanalytics GmbH
Address: Otto-Volger-Strasse 3c, 65843 Sulzbach/Taunus, Germany
Work Phone: +49 (6196) 766 84 28
Email: Judith.kanz@mainanalytics.de
Website: www.mainanalytics.de