

FAST AND HIGH QUALITY VALIDATION PROCESS

Alessia Sacco, Valos, Genova, Italy
Dmitri Petratchenko, Valos, Genova, Italy
Andrea Parodi, Valos, Genova, Italy

INTRODUCTION

The purpose of a validation plan is to describe the validation activities performed for checking errors in databases, tables, listings and figures produced. All the objects (datasets and outputs) that require validation have to be listed in it. Three levels of validation exist:

- Most Critical (MC): double programming by an independent statistical programmer will be performed.
- Critical (C): validation by an independent statistical programmer or statistician will be performed. Validation does not have to be double programmed.
- Non-Critical (NC): validation in the form of manual review by an independent statistical programmer or statistician will be performed.

Once the validation level of objects has been set, a tracking sheet describing the activity of the team working on the project needs to be compiled. This document usually includes for each object:

- validation level
- priority
- main programmer
- status (ongoing / ready for validation / final)
- date linked to the status
- program used for creating the object
- validation programmer
- validation status (fail / ongoing / validated)
- date linked to the validation status
- program used for validating the object
- comments

Both main and validation programmers keep this documents updated. When the objects to be compared have validation level MC, once the double programming has been performed several procedures exist for comparing results produced, but most of them does not take in account all aspects they should.

COMPARING TWO DATASETS

The macro %dbcompare we are using has many advantages that can be summarized by steps. Firstly, with PROC CONTENTS, this macro allows the user to check if the datasets in input are comparable. A list of variables present in one but not in the other dataset are listed. Anyway, these differences do not stop the macro that goes through comparing the format and length of variables. Once the differences in types are fixed we can go on with the last step of the macro. This consists in showing in a dataset the observations without an equal counterpart in both datasets, specifying from what dataset they come from.

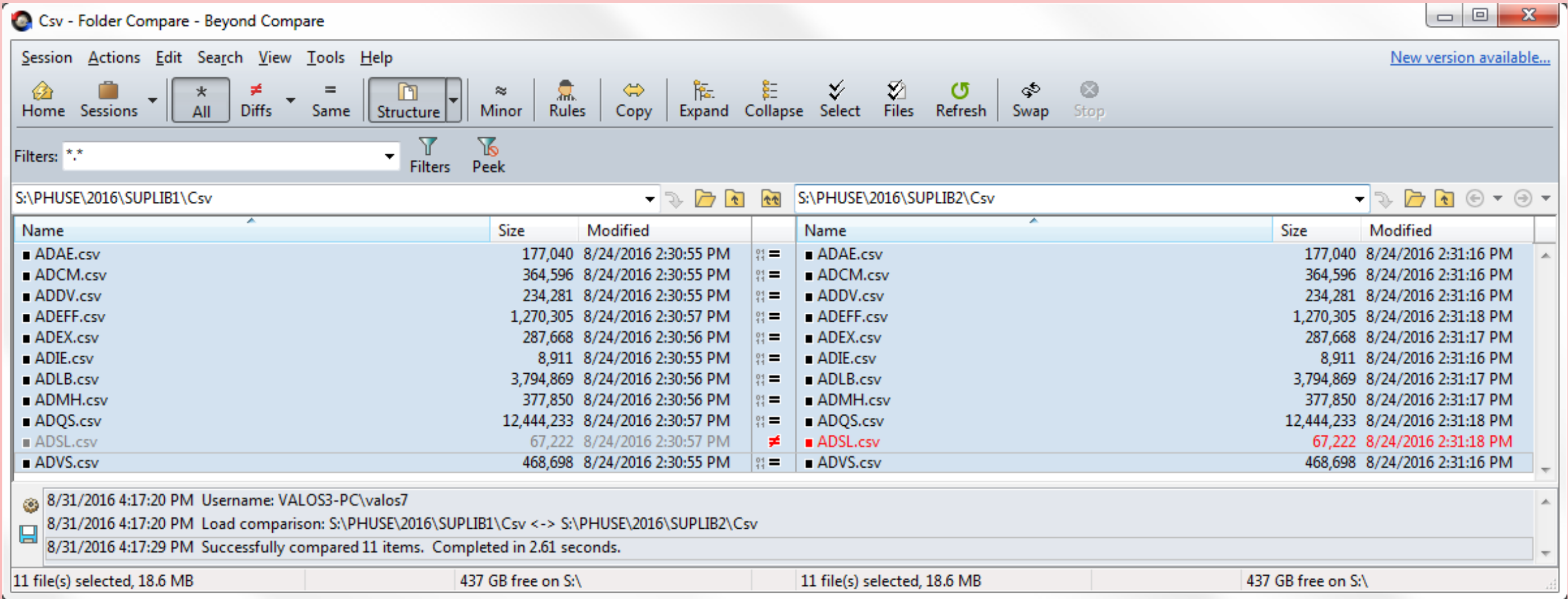
	UNIQUE SUBJECT IDENTIFIER	COUNTRY	AGE	SEX	HEIGHT AT BASELINE (CM)	WEIGHT AT BASELINE (KG)	BODY MASS INDEX AT BASELINE	PLANNED TREATMENT	_from
1	S01N01P01	AUS	31	F	160	66	25.78125	test TRT1	qc dataset
2	S01N01P01	ITA	31	F	160	66	26.296875	test TRT1	main dataset
3	S01N02P03	ITA	32	F	165	53	19.467401286	test TRT2	qc dataset
4	S01N02P03	ITA	32	F	165	53	19.856749312	test TRT2	main dataset

Moreover, this macro has two really important skills.

To avoid a lot of differences in the output dataset, the user can input the variables to be excluded from comparison. For example, once we find a difference in the calculation of a score, it is useful to remove the differences generated by this and check for other differences one cannot notice because of too many errors in output.

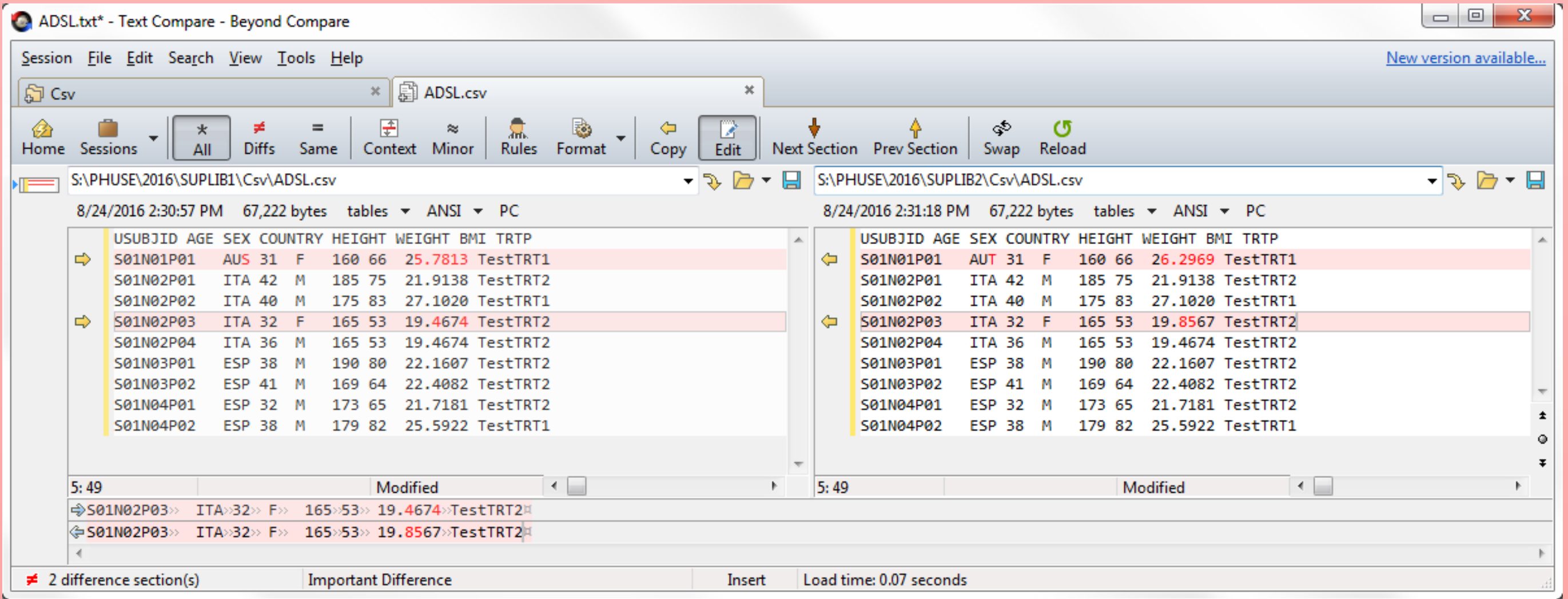
USING BEYOND COMPARE®

Beyond Compare® allows the user to perform directly comparison between folders, text and other types of file. Once we exported the datasets to be compared we can compare these with this program, then click on tables found different.



Three symbols are shown:

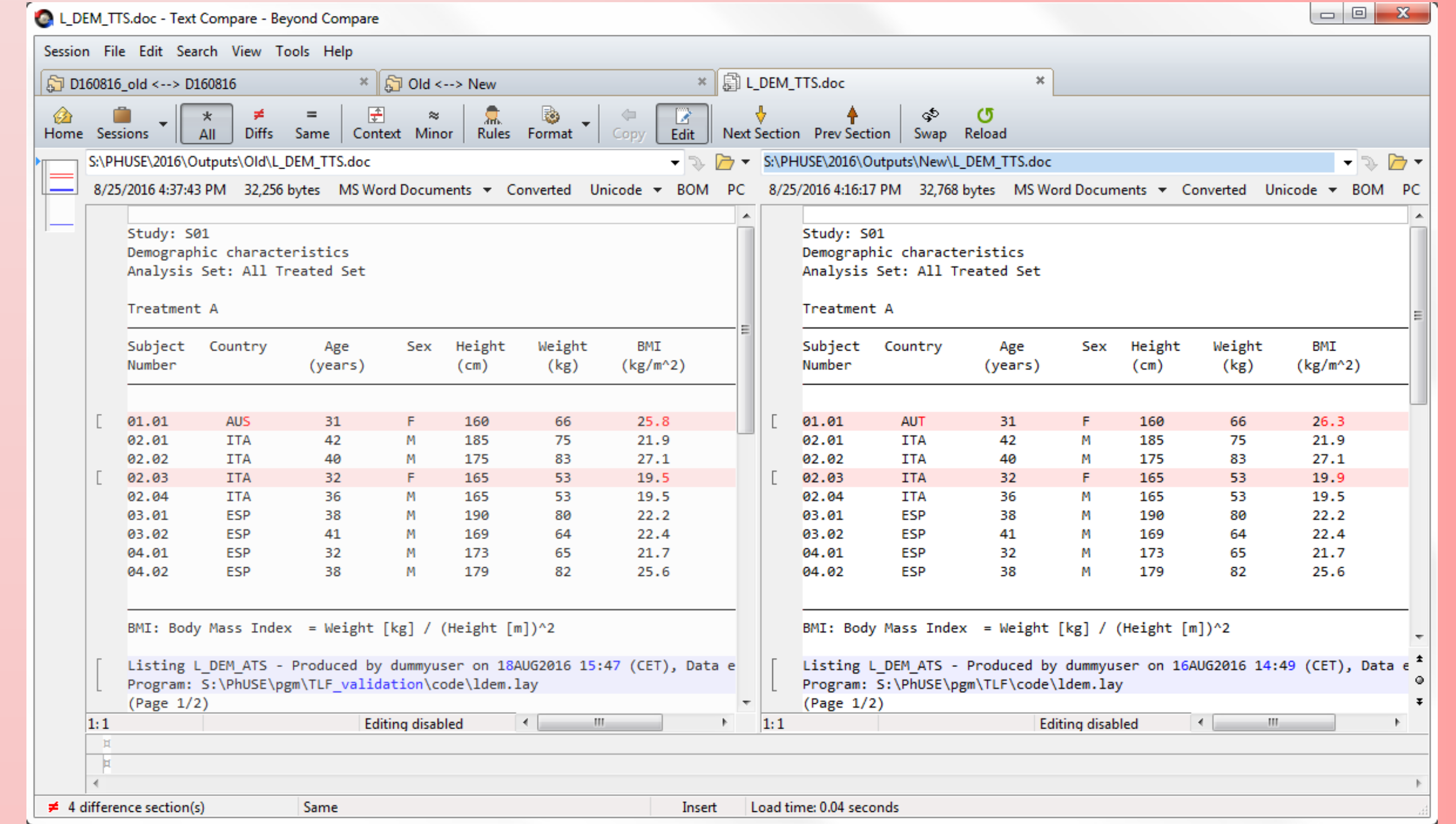
- = means that outputs are identical
- ≈ means that outputs are identical not considering the exceptions we put in the program
- ≠ means that outputs are different



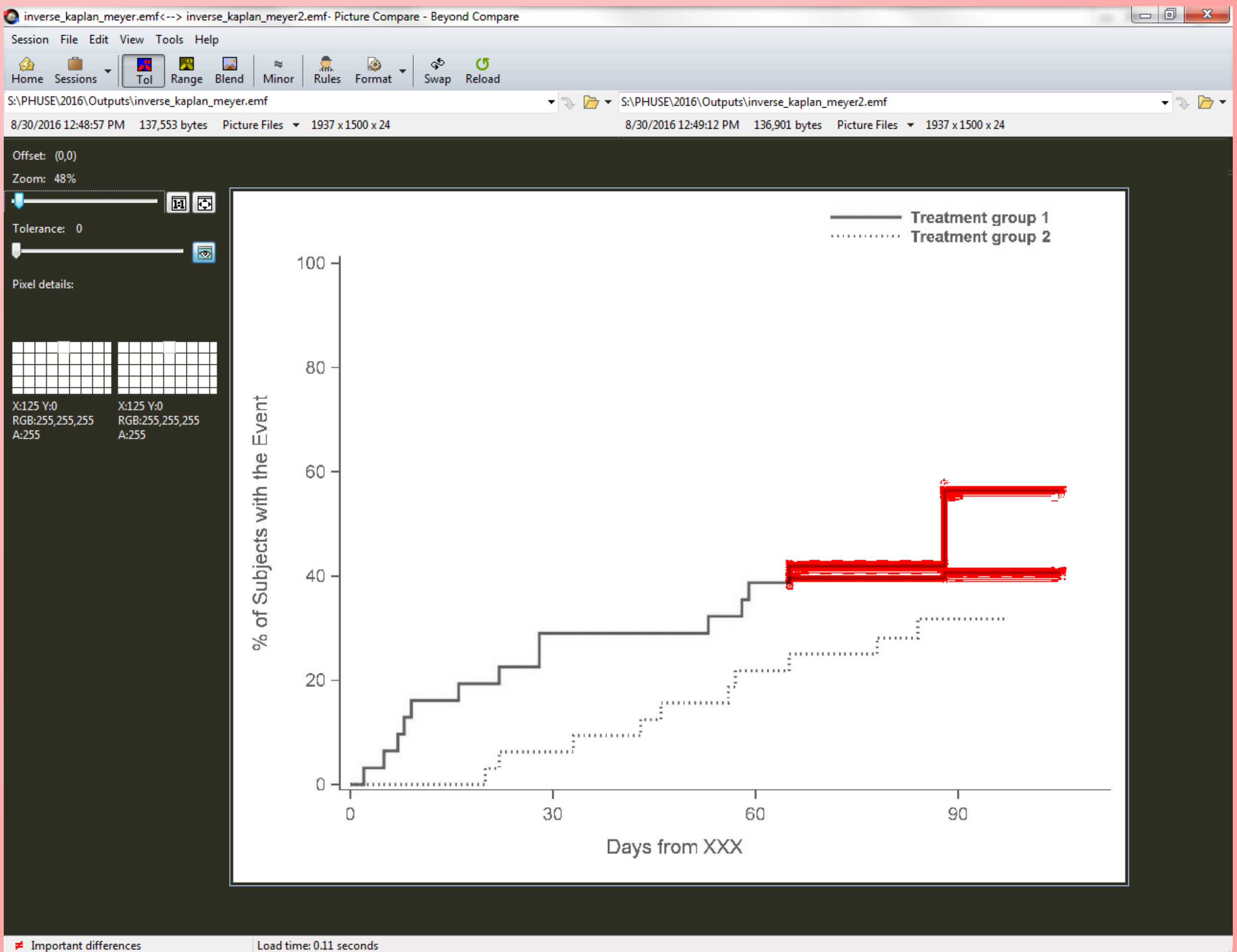
COMPARING OUTPUTS WITH BEYOND COMPARE®

Depending on the format tables and listings are produced, different ways of comparing these outputs exist. Not to mention the countless errors one cannot find comparing tables and listings simply looking at them, we think the best way for perform the validation is to produce tables and listings in Microsoft Word® or Rich Text formats and check them with a program that permits to

compare all the outputs automatically and mark only the outputs with different values.



At the top left of the screen there is a sort of summary indicating the differences in the rows: red rows indicate the differences, blue rows indicate the exceptions (a useful facility for avoiding the program to output as errors all that difference that we don't want to consider).



Moreover, Beyond Compare® processes the graphs and outputs a new graph with the differences marked in red.

CONCLUSION

Programmers in charge of validating datasets and outputs must be aware that they not only have to focus on the outputs to be produced, but also on the way they have to compare them to the main outputs. This way must be as much as possible free of errors. The best and less time-consuming method for us is the one described in this poster.