Paper SM01

Let's Compare Them All – TFL Comparisons

Katja Glaß, Bayer AG, Berlin, Germany

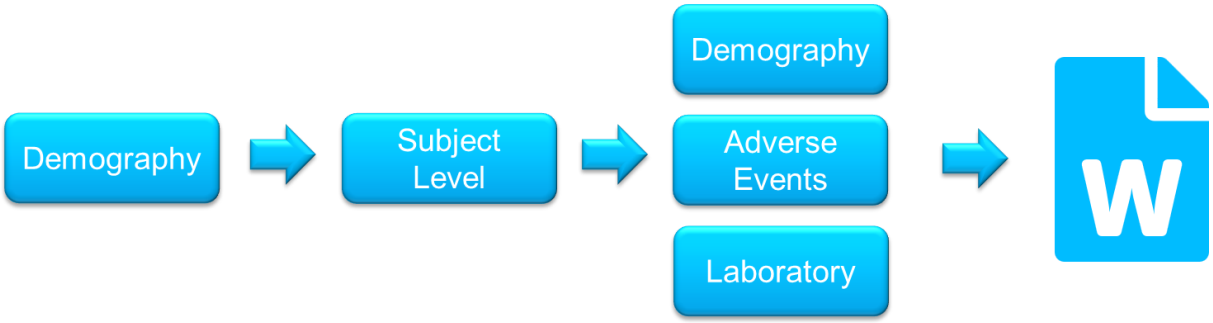
ABSTRACT

In our daily work, even minor changes can sometimes have unexpected consequences. To ensure nothing unexpected has changed, we often rely on comparison methods. This paper will focus on the comparison of outputs such as tables, listings and figures. We will also explore comparison options and provide helpful hints on how to perform these comparisons effectively.

INTRODUCTION

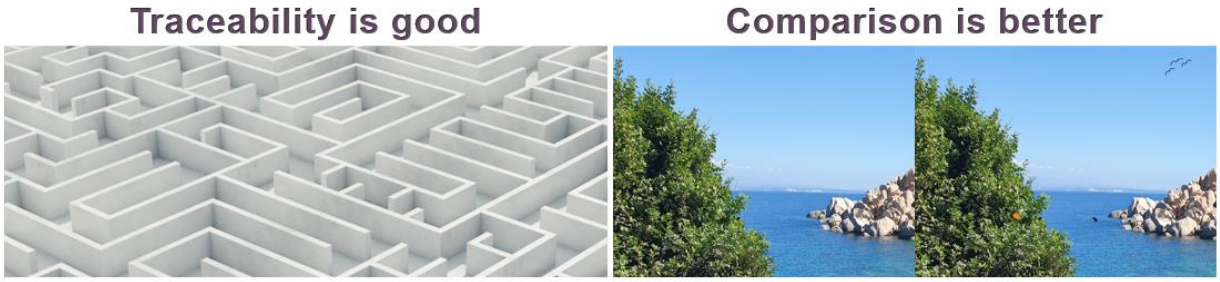
Most of us are well aware that minor changes can have a major impact, especially when complex processes or dependencies are involved. Even though there might be just one value changed in a database, it can lead to unexpected alterations in the output table that you might not have been aware of.

A possible solution could be traceability! Then we could follow along what will change and directly see the impact. Unluckily often we do not have traceability on a granular level. Let's assume the following example.



In the example, we do have a change in the Demography data. According to traceability, we know that this dataset is used to derive some content for the Subject Level dataset (ADSL). This dataset might also be used to merge information into all other datasets. But does this mean that all values in all datasets, and consequently all output tables, have changed? Depending on the nature of the change, it is highly unlikely. It is possible that the value in the Demography dataset has not been used to derive anything in the subject level metadata, which means the other datasets might not have changed at all. However, if the change is in a calculated age group, for example, it could have an impact on the other datasets and outputs, as it might result in shifting a subject from one group to another. Sometimes it is hard to predict the impact of small changes.

While traceability can be a solution to investigate what might have changed, depending on the level of detail, it might not be sufficient. To obtain the detailed changes in the final outputs, the easiest way is to compare these final deliverables. In Bayer, we deliver Word documents where an automated PDF-rendered file is created and submitted. The Word format allows us much flexibility to read tables and images smoothly.

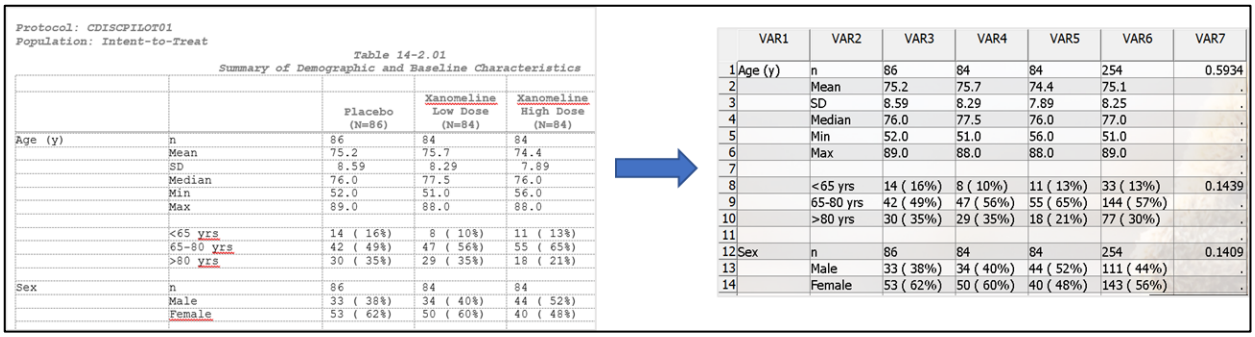


When we want to perform comparison we should consider the purpose and type of document. Depending on the document type, whether this is about the tables and figures or listing section, single table RTF files, SAPs, Narrative documents, specifications or what ever, different things might be of interest.

COMPARE TEXTS

Usually, the need to compare texts arises when we have already delivered a finalized table and figure, or listing package and subsequently make changes that require re-running the programs. Although we don't typically expect any modifications, it is often necessary to perform a quick comparison.

To facilitate comprehensive and efficient text comparisons, we can import Word documents into SAS datasets using PROC GROOVY, a SAS procedure designed for this purpose. The process and accompanying SAS macro source code for achieving this are elaborated in the paper titled "Let's get Groovy – Word and PDF processing in SAS"[1].



Protocol: CDISCPILOT01
Population: Intent-to-Treat

Table 14-2.01
Summary of Demographic and Baseline Characteristics

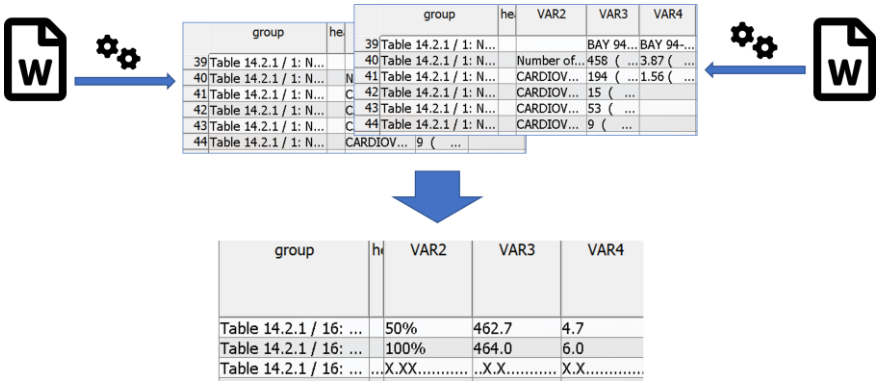
		Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)
Age (y)	n	86	84	84
	Mean	75.2	75.7	74.4
	SD	8.59	8.29	7.89
	Median	76.0	77.5	76.0
	Min	52.0	51.0	56.0
	Max	89.0	88.0	88.0
	<65 yrs	14 (16%)	8 (10%)	11 (13%)
	65-80 yrs	42 (49%)	47 (56%)	55 (65%)
	>80 yrs	30 (35%)	29 (35%)	18 (21%)
Sex	n	86	84	84
	Male	33 (38%)	34 (40%)	44 (52%)
	Female	53 (62%)	50 (60%)	40 (48%)

	VAR1	VAR2	VAR3	VAR4	VAR5	VAR6	VAR7
1	Age (y)	n	86	84	84	254	0.5934
2		Mean	75.2	75.7	74.4	75.1	.
3		SD	8.59	8.29	7.89	8.25	.
4		Median	76.0	77.5	76.0	77.0	.
5		Min	52.0	51.0	56.0	51.0	.
6		Max	89.0	88.0	88.0	89.0	.
7							.
8		<65 yrs	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
9		65-80 yrs	42 (49%)	47 (56%)	55 (65%)	144 (57%)	.
10		>80 yrs	30 (35%)	29 (35%)	18 (21%)	77 (30%)	.
11							.
12	Sex	n	86	84	84	254	0.1409
13		Male	33 (38%)	34 (40%)	44 (52%)	111 (44%)	.
14		Female	53 (62%)	50 (60%)	40 (48%)	143 (56%)	.

Once the documents are read into SAS datasets, the next step is to compare these datasets. One approach to accomplish this is by utilizing PROC COMPARE. By summarizing the findings in the log, we can conveniently identify the presence of any differences and promptly address them.

The overall processing steps involve:

- Importing the documents to SAS
- Replace timestamps
- Utilizing PROC COMPARE for a simple comparison
- Printing relevant information to the log



	group	he			
39	Table 14.2.1 / 1: N...				
40	Table 14.2.1 / 1: N...	N			
41	Table 14.2.1 / 1: N...	C			
42	Table 14.2.1 / 1: N...	C			
43	Table 14.2.1 / 1: N...	C			
44	Table 14.2.1 / 1: N...		CARDIOV...	9 (...	

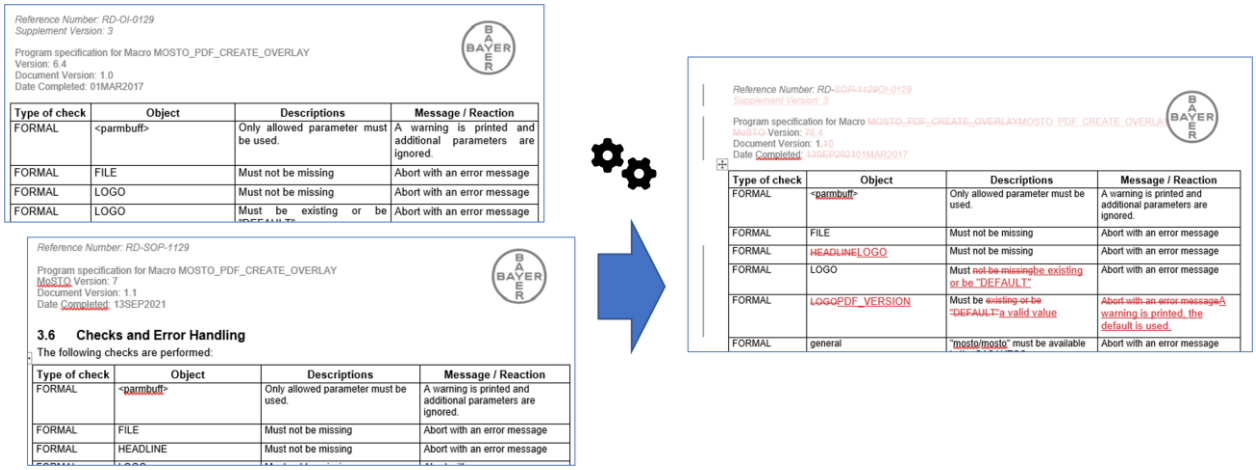
	group	he	VAR2	VAR3	VAR4
39	Table 14.2.1 / 1: N...			BAY 94...	BAY 94-...
40	Table 14.2.1 / 1: N...		Number of...	458 (...	3.87 (...
41	Table 14.2.1 / 1: N...		CARDIOV...	194 (...	1.56 (...
42	Table 14.2.1 / 1: N...		CARDIOV...	15 (...	
43	Table 14.2.1 / 1: N...		CARDIOV...	53 (...	
44	Table 14.2.1 / 1: N...		CARDIOV...	9 (...	

group	he	VAR2	VAR3	VAR4
Table 14.2.1 / 16: ...		50%	462.7	4.7
Table 14.2.1 / 16: ...		100%	464.0	6.0
Table 14.2.1 / 16: ...		X.XX.....	X.X.....	X.X.....

By utilizing PROC GROOVY, we can achieve rapid comparisons. For instance, in our environment, we successfully compared a massive 80MB listing file consisting of 8500 pages in just 41 seconds. However, if there are numerous changes, such as added lines, the comparison on observations might take longer than anticipated since it involves a line-by-line evaluation. Additionally, interpreting the results within the dataset needs to establish a connection between each dataset observation and its corresponding location in the Word document. Nonetheless, this efficient change detection capability proves to be quite suitable for a wide range of practical scenarios.

COMPARING WITH TRACK CHANGES

To streamline the process of identifying changes within Word documents, it is highly beneficial to utilize the track changes visibility feature in Microsoft Word.



Implementing such functionality can be achieved through Visual Basic for Applications (VBA), in the form of a Word macro. At Bayer, we developed a tool with these capabilities back in 2013. As the size of Word documents grow, it has become more efficient to utilize other programming languages. For instance, at Bayer, we leverage a commercial Java library called "Aspose for Word," although there may be other alternative tools available.

The processing involves:

- Importing the documents
- Creating "track-changes" outputs to visualize differences (if any)
- Printing a summary in the log indicating whether changes are present
- Optionally, removing timestamps and similar elements before performing the comparison

This solution provides a highly user-friendly view of the changes, making it intuitive to review the document. To enhance the clarity of the comparison, it is advisable to remove changes that are predictable or expected, such as footnotes containing timestamps. However, depending on the number of changes present, the provided level of information may still be insufficient. For example, if a table is shifted from one location to another, it may appear as if it was deleted and subsequently added. Additionally, when dealing with exceptionally large files, there is a possibility of encountering out-of-memory issues. In our environment, we experienced an out-of-memory error when attempting to compare an 80MB file with 8500 pages. Nonetheless, this solution works remarkably well for "smaller" yet still substantial files. For instance, a 30MB file containing 990 pages only took 2 minutes and 14 seconds to compare in our environment.

DETAILED TABLE COMPARE

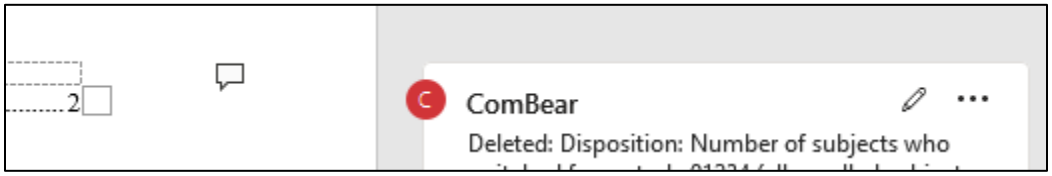
In certain cases, a more comprehensive examination of changes is necessary, especially when it comes to tables. Specific details that might be of interest include:

- Changes in table titles
- Addition or deletion of tables
- Addition or deletion of rows within a table
- Addition or deletion of columns within a table
- Changes in individual table cells
- Modifications in the footer (excluding identifying information)

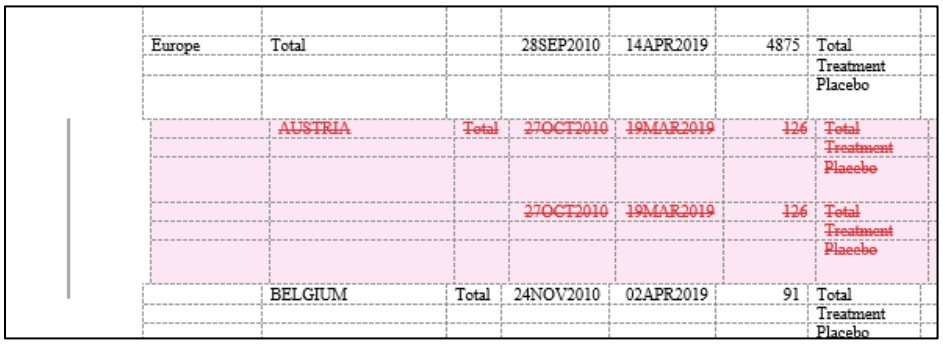
It would be ideal to visualize these changes within a Word document, utilizing the "track changes" view for modifications and incorporating comments for general information.

Implementing this functionality requires more complex programming logic. At Bayer, we achieved this using Java, which enables fast processing. First, we read all the tables into distinct objects and match them based on potential changes in their location. Subsequently, we perform various checks, such as identifying changes in the title or detecting added and deleted tables. We also compare row counts to determine which rows have been added or removed, and we apply the same process for columns. In addition, we examine the content of individual cells alongside other investigations. Based on the differences identified, we create a visual representation of the changes using the "track changes" feature, accompanied by relevant comments.

A deleted table is indicated in the first line of the document as a comment:



Removed rows will include next to the comments also a detailed track changes view:



Europe	Total		28SEP2010	14APR2019	4875	Total Treatment Placebo
~~AUSTRIA~~	~~Total~~		~~27OCT2010~~	~~19MAR2019~~	~~126~~	~~Total~~ ~~Treatment~~ ~~Placebo~~
			~~27OCT2010~~	~~19MAR2019~~	~~126~~	~~Total~~ ~~Treatment~~ ~~Placebo~~
BELGIUM	Total		24NOV2010	02APR2019	91	Total Treatment Placebo

Additionally we collect all information and provide these back to SAS, to include such information into the log like the following:

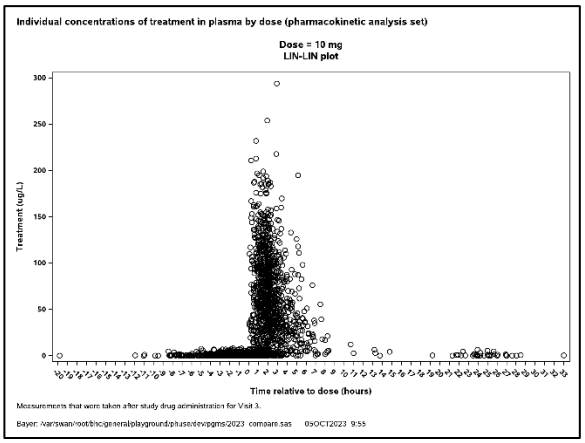
```
COMBEAR - -------------- Summary -----------------
COMBEAR - 6 out of 13 compared tables changed.
COMBEAR - 1 tables are added.
COMBEAR - 1 tables are removed.
COMBEAR - Table changes detected in the document.
```

This detailed table comparison approach allows for intuitive analysis of all changes. It facilitates a comprehensive review of the complete document, including comments and change revisions. Moreover, this method accommodates highly complex scenarios. By leveraging Java instead of VBA as the programming language, the performance is significantly improved. In our environment, for a 30MB file with 990 pages, the application only took 2 minutes and 9 seconds to complete the comparison.

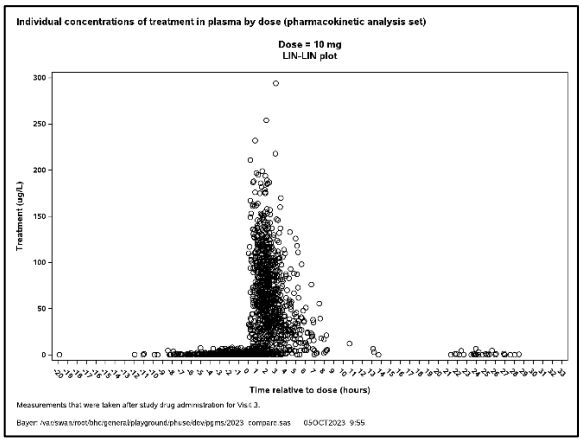
COMPARE IMAGES

In addition to tables and listing texts, our final deliverables often include figures. Comparing images visually can be challenging, but implementing an image comparison algorithm can prove beneficial.

Old Image



New Image



When comparing two images, it is typically desirable to determine whether any changes have occurred. Users typically hope that the image remains unchanged, but if changes are present, they might be interested in understanding the extent of the modifications and visualizing them.

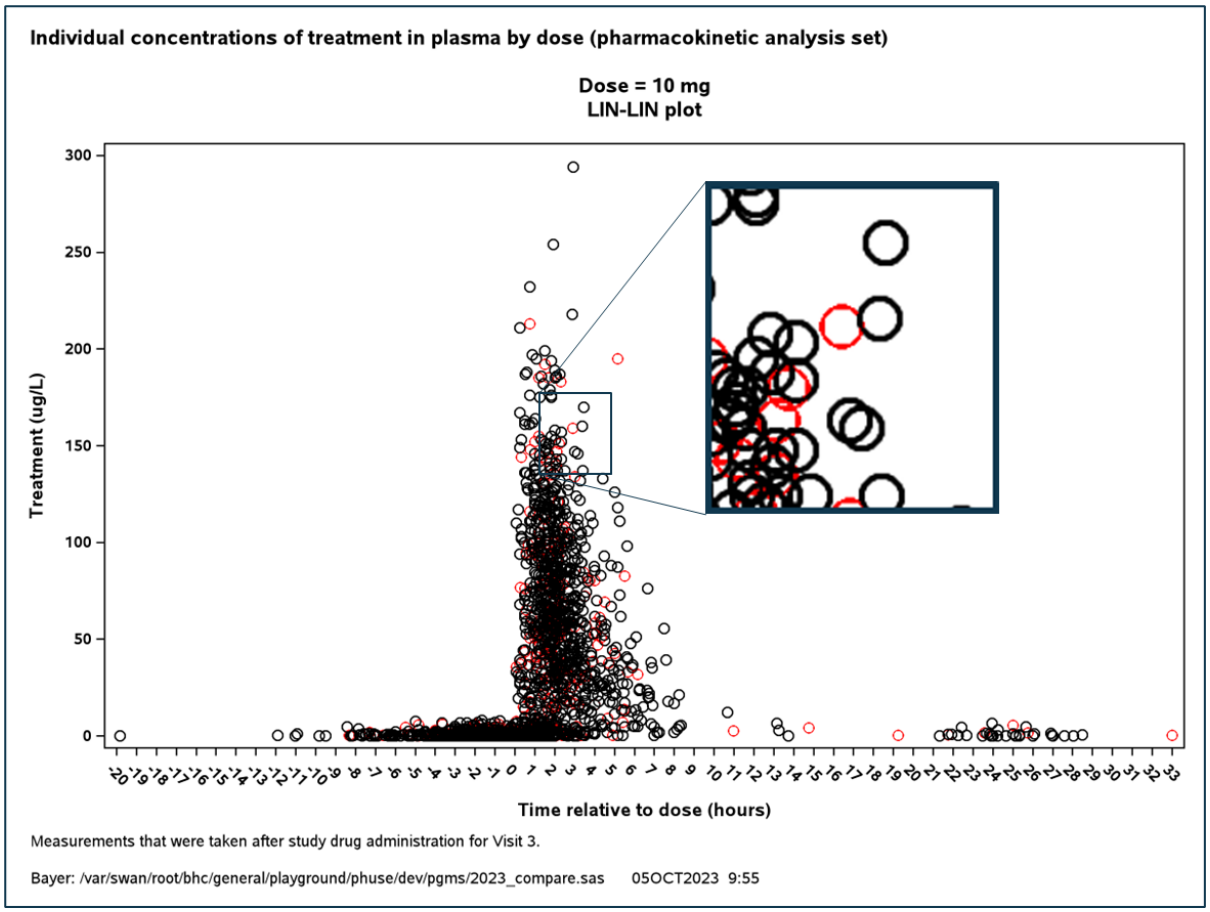
We can leverage basic functionality from Java to work with the images. SAS provides support for Java through various methods, such as the PROC GROOVY procedure[1], enabling straightforward integration of this functionality.

Here is a simplified example of how to compare images using a pixel-based algorithm in Java:

```
BufferedImage oldImage = ImageIO.read(oldFile);
BufferedImage newImage = ImageIO.read(newFile);
int pixelDiff = 0;
if (oldImage.getWidth() == newImage.getWidth() &&
    oldImage.getHeight() == newImage.getHeight()) {
    for (int x = 0; x < oldImage.getWidth(); x++) {
        for (int y = 0; y < oldImage.getHeight(); y++) {
            if (oldImage.getRGB(x, y) != newImage.getRGB(x, y)) {
                pixelDiff++;
                newImage.setRGB(x, y, RED_COLOR);
            }
        }
    }
}
```

In this code snippet, we read the old and new images into BufferedImage Java objects, allowing programmatic access to the images. We then compare the pixels between the two images. If there is a difference in color, we increment the pixel difference count and set the corresponding pixel in the new image to red for visualization.

Once the comparison is complete, we can log messages to indicate differences, such as issuing a warning when changes are present. Additionally, the modified image containing red pixels highlighting the changes can be saved.



The user can easily review this image to visually identify the actual changes, complementing the notification in the log that changes are present. A corresponding SAS macro utilizing Java in PROC GROOVY is available as an attachment to this paper. Furthermore, the program can be modified to exclude the footer, which might contain a timestamp, if desired.

CONSIDERATIONS

As we have explored, different problems require different solutions. Depending on the specific problem at hand, certain comparison methods may be more suitable than others. Additionally, it is possible to combine different comparison techniques. For example, combining a detailed table comparison with image comparison can be particularly effective when dealing with documents that contain both tables and figures.

When developing a solution, it is crucial to consider the scope and the specific needs of the user. Understand the type of information the user is seeking. In many cases, simply knowing whether any changes have occurred is sufficient. By understanding the use case thoroughly, you can provide the most appropriate and tailored solution.

SUMMARY

In summary, we have discovered that while traceability is a valuable aspect, comparison provides even greater insights into the changes that occur in output files when certain data points are updated. Users often require a quick overview of changes, making it logical to utilize the log for easy access to summarized information. Additionally, quick results in form of SAS datasets might be sufficient. When it is necessary to closely examine detailed changes, the track changes functionality in word, along with comments and a visual comparison of images highlighting the changes, proves to be highly useful.

Throughout this paper, we have learned how to implement simple text comparisons, track-changes comparisons, special processing techniques, and image comparisons. It is essential to recognize that even minor changes can have a major impact, and performing comparisons offers a reliable approach to detecting changes effectively and with confidence.

REFERENCES

[1] “Let's get Groovy - Word and PDF processing in SAS”, PHUSE 2022 EU Connect paper by Katja Glaß (https://phuse.s3.eu-central-1.amazonaws.com/Archive/2022/Connect/EU/Belfast/PAP_SM04.pdf)

CONTACT INFORMATION

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

Katja Glaß
Bayer AG
Müllerstraße 178
13353 Berlin, Germany
Email: katja.glass@bayer.com

ATTACHMENT

The following source code uses the unlicensed license and can be used without any reference or similar.

```
%MACRO compare_figure(infile1=, infile2=, outfile=);
    FILENAME cmd8295 TEMP;
    DATA _NULL_;
        FILE cmd8295 LRECL=500;
        PUT 'PROC GROOVY;                                                                                    ';
        PUT '    SUBMIT "&infile1" "&infile2" "&outfile";                                                    ';
        PUT '        import java.awt.image.BufferedImage;                                                    ';
        PUT '        import java.awt.Color;                                                                  ';
        PUT '        import javax.imageio.ImageIO;                                                           ';
        PUT '                                                                                                ';
        PUT '        final int RED_COLOR = new Color(255, 0, 0).getRGB();                                    ';
        PUT '                                                                                                ';
        PUT '        String image1 = args[0];                                                                ';
        PUT '        String image2 = args[1];                                                                ';
        PUT '        String out    = args[2];                                                                ';
        PUT '                                                                                                ';
        PUT '        System.out.println("Compare picture " + image1);                                       ';
        PUT '                                                                                                ';
        PUT '        File fileImage1 = new File(image1);                                                     ';
        PUT '        File fileImage2 = new File(image2);                                                     ';
        PUT '        File fileOut = new File(out);                                                           ';
        PUT '                                                                                                ';
        PUT '        BufferedImage img1=null;                                                                ';
        PUT '        BufferedImage img2=null;                                                                ';
        PUT '        img1 = ImageIO.read(fileImage1);                                                        ';
        PUT '        img2 = ImageIO.read(fileImage2);                                                        ';
        PUT '                                                                                                ';
        PUT '        int pixelDiff = 0;                                                                      ';
        PUT '        if (img1.getWidth() == img2.getWidth() && img1.getHeight() == img2.getHeight()) {       ';
        PUT '            for (int x = 0; x < img1.getWidth(); x++) {                                         ';
        PUT '                for (int y = 0; y < img1.getHeight(); y++) {                                    ';
        PUT '                    if (img1.getRGB(x, y) != img2.getRGB(x, y)) {                               ';
        PUT '                        pixelDiff++;                                                            ';
        PUT '                        img2.setRGB(x, y, RED_COLOR);                                           ';
        PUT '                    }                                                                           ';
        PUT '                }                                                                               ';
        PUT '            }                                                                                   ';
        PUT '        } else {                                                                                ';
        PUT '            System.out.println("ERROR: Height and/or Weight for both images are not equal."); ';
        PUT '            return;                                                                             ';
        PUT '        }                                                                                       ';
        PUT '        if (pixelDiff != 0) {                                                                   ';
        PUT '            System.out.println("WARNING: " + pixelDiff + " changes found for " + image1);     ';
        PUT '            ImageIO.write(img2, "png", fileOut);                                                ';
        PUT '        } else {                                                                                ';
        PUT '            System.out.println("No changes found: " + image1);                                ';
        PUT '        }                                                                                       ';
        PUT '    ENDSUBMIT;                                                                                  ';
        PUT 'QUIT;                                                                                           ';
    RUN;
    %INCLUDE cmd8295;
    FILENAME cmd8295;
%MEND compare_figure;


%compare_figure(
    infile1        = &dir/&filename1..png
  , infile2        = &dir/&filename2..png
  , outfile        = &dir/&filename_out..png)
```