PharmaSUG 2025 - Paper PO-311

Statistical Programming Outputs Comparison and Reporting Using Python

Vinayak Mane, Tanusree Bhattacharyya, Adel Solanki, Anindita Bhattacharjee, Nibedita Hazra
Inference Inc.

ABSTRACT

This paper showcases the functionality and workflow of an automated Python tool designed to compare different versions of clinical trial outputs, particularly in the context of statistical programming. This tool, named the "Outputs Comparison Tool," is a user-friendly executable (.exe) application that streamlines the process of comparing and analyzing the differences in original and revised outputs.

Iterations of data sets, tables, listings, and figures (TLFs) are common in the programming workflow, often resulting from updates in work scope or analysis requirements. These iterations lead to multiple intermediate submissions for review. Ensuring consistency across outputs in these iterations is a challenging task for statistical programmers, typically performed manually, making the process both tedious and error-prone.

The "Outputs Comparison Tool" automates this process and offers the following key features:

- **Count of outputs** - the total number of files from the two versions are compared and reported.
- **Table of Contents comparison** - the presence of each output is checked in the two versions.
- **Content differences** - updates in the output contents are highlighted across versions.
- **Consolidated Comparison Report** - This report generates a comprehensive list of compared outputs, along with a reference to the corresponding pages with changes.

This lightweight and robust application empowers statistical programmers to enhance productivity and maintain precision in output review processes. Designed with simplicity in mind, the tool does not require any programming expertise and supports standard file formats like .rtf, .txt, .doc, .docx, and .pdf, commonly used in statistical reporting workflows, making it a powerful resource.

INTRODUCTION

This tool presents an automated Python-based solution for comparing documents. The solution leverages Python libraries and Microsoft Word's Component Object Model Application Programming Interface (COM API) at the back-end of the tool to streamline the extraction of content, document comparison, and generation of detailed reports. By automating these processes, the tool reduces human effort, minimizes errors, and enhances productivity.

Primary Objectives of the Comparison Tool

1. **Streamline the process of reports comparison**

   In the workflow of statistical programmers' activities, the iterations of data sets and TLFs are unavoidable. Maintaining the quality at every iteration can be a challenge when things are checked manually. There are several items to be compared when TLFs get updated, including the number of reports, the file names, and the content within these reports. This tool will do all of these in a defined flow and generate a summary at each step. This is explained elaborately in the sections Count of Outputs, Table of Contents Comparison, and Content Differences.

2. **Improve quality**

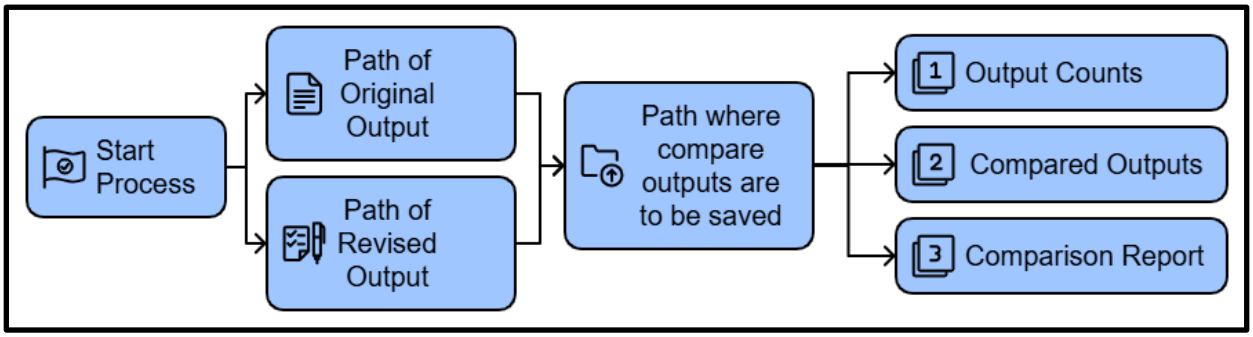
   A Python-based executable tool, the 'Output Comparison Tool' ensures coherent application of predefined rules and formatting, thus enhancing the quality of deliverables. Built-in validation steps within the tool ensure that outputs are monitored across version changes consistently.

3. **Save time**

   Repeating the task for outputs comparison is tedious and, when done manually, can take several days to complete, depending on the complexity and volume of data. The output comparison tool sums up all the changes and makes navigation centralized through the outputs. This leads to a significant reduction in time and effort in every iteration. Refer to the section Consolidated Comparison Report for a detailed description of the consolidated report.

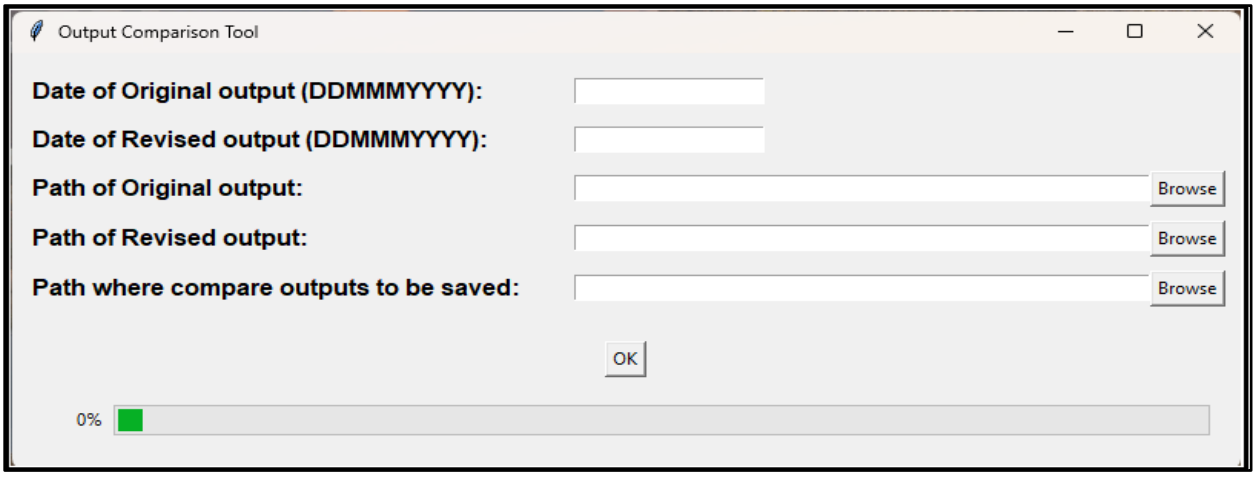
By addressing these objectives, the tool eases meeting deadlines, maintains data quality, and achieves compliance in the highly regulated clinical research environment.

WORKFLOW OF THE "OUTPUT COMPARISON TOOL"



Flowchart 1

STEP 1 – DEFINE OUTPUT PATHS



Display 1. Main Interface

As a first step, the tool receives the locations of the two versions of the outputs from the user through the **Display 1** User Interface.

```
    def process_files_to_excel(original_path, revised_path, output_folder,
old_version_date, new_version_date):
    original_files = {f for f in os.listdir(original_path) if f.endswith('.rtf') and not
f.startswith('~$')}
    revised_files = {f for f in os.listdir(revised_path) if f.endswith('.rtf') and not
f.startswith('~$')}
    excel_folder, _, _ = create_output_folders(output_folder, original_files)
    workbook = openpyxl.Workbook()
    summary_sheet = workbook.active
    summary_sheet.title = "Summary of Output counts"
    ...
    # Process files and fill in workbook with details
    workbook.save(output_file)
    return extracted_titles
```

Program 1. process_files_to_excel

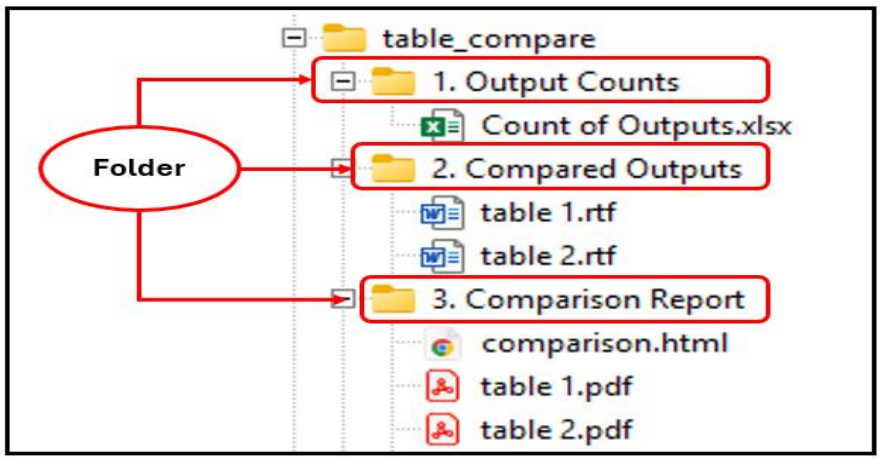
Program 1 is the back-end Python code snippet for step 1, it processes files from two directories (original and revised), compares their contents, and writes the differences into an Excel workbook. It tracks the presence of tables and listings, extracts titles, and calculates the page count for both versions of the documents. A summary sheet is created with the counts of each output type (table, listings), and a detailed sheet contains information on discrepancies.

Function Breakdown of Program 1:

1. Initialization:
    - Retrieves a list of files from both the original and revised directories.
    - Calls create_output_folders() to generate the output folder structure as per the **Display 2** image.
    - Initializes a new workbook with two sheets: one for a summary of counts and one for details of differences.

STEP 2 – FOLDER STRUCTURE

After executing "Display 1", the **Display 2** folder structure gets generated:



Display 2. Folder Structure

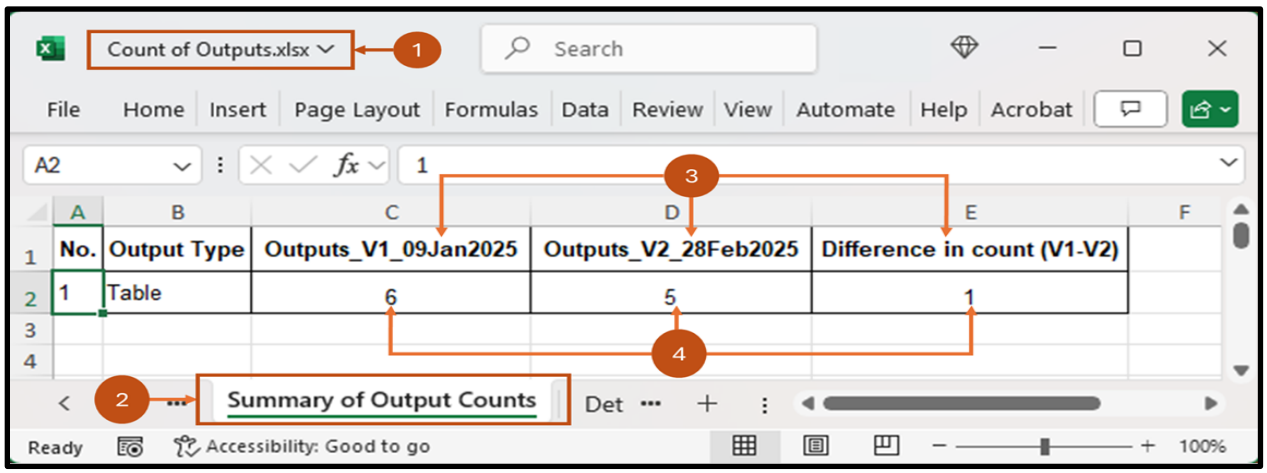
STEP 3 – EXCEL FILE

The tool generates a folder named "Output Counts". Within these folders, an Excel file titled "Count of Outputs.xlsx" is created. This file contains **Tab 1** and **Tab 2**.

Tab 1 – "Summary of Output Counts"

This tab summarizes the count of outputs across the versions and highlights the difference in count to bring the programmer's attention to these differences.

The **Display 3** screenshot explains the "Count of Outputs" in detail:



Display 3. Count of Outputs

A. Content of Count of Outputs Tab

1. The first pointer shows Excel files saved as **"Count of Outputs"** as shown in the Display 3.

2. The second pointer shows a tab named **"Summary of Output Counts"**. This sheet provides an overall summary of the counts of different types of output (e.g., tables and listings) in the original and revised versions.

3. Third pointer shows,

   3.1 Outputs_V1_[old_version_date] - Count of the outputs in the original version files.

   3.2 Outputs_V2_[new_version_date] - Count of the outputs in the revised version files.

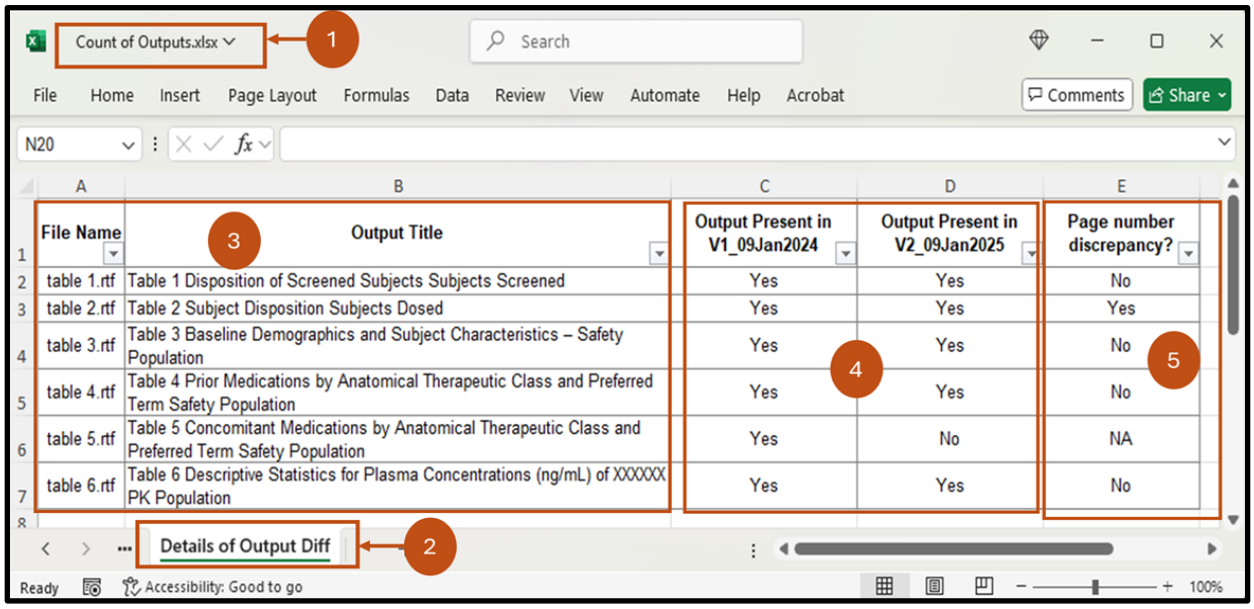
   3.3 Difference in count (V1-V2) - The difference in counts between the original and revised files.

4. The fourth pointer shows, there are 6 files in the original folder and 5 in revised folder, so the difference is 1. In this example, there is 1 extra file in the original folder which is not present in the revised folder.

Tab 2 – “Details of Output Diff”

“Details of Output Diff” - This tab lists the presence of each output across the versions. It further gives a quick peek into the change in number of pages across these outputs from the two versions.

The **Display 4** screenshot explains the “File Discrepancies” in details



File Name	Output Title	Output Present in V1_09Jan2024	Output Present in V2_09Jan2025	Page number discrepancy?
table 1.rtf	Table 1 Disposition of Screened Subjects Subjects Screened	Yes	Yes	No
table 2.rtf	Table 2 Subject Disposition Subjects Dosed	Yes	Yes	Yes
table 3.rtf	Table 3 Baseline Demographics and Subject Characteristics – Safety Population	Yes	Yes	No
table 4.rtf	Table 4 Prior Medications by Anatomical Therapeutic Class and Preferred Term Safety Population	Yes	Yes	No
table 5.rtf	Table 5 Concomitant Medications by Anatomical Therapeutic Class and Preferred Term Safety Population	Yes	No	NA
table 6.rtf	Table 6 Descriptive Statistics for Plasma Concentrations (ng/mL) of XXXXXX PK Population	Yes	Yes	No

Display 4. Details of File Discrepancies

B. Content of File Discrepancies Tab

1. The first pointer shows Excel files saved as “**Count of Outputs**” as shown in the Display 4.

2. The second pointer is about the sheet that provides detailed information about each output file, including its title, presence in the original and revised versions, and mismatch in page counts.

3. Third pointer is about File Name & Output Title – **‘File Name’** is the name of the file being processed. And the second column, **‘Output Title’** is the title extracted from the file content.

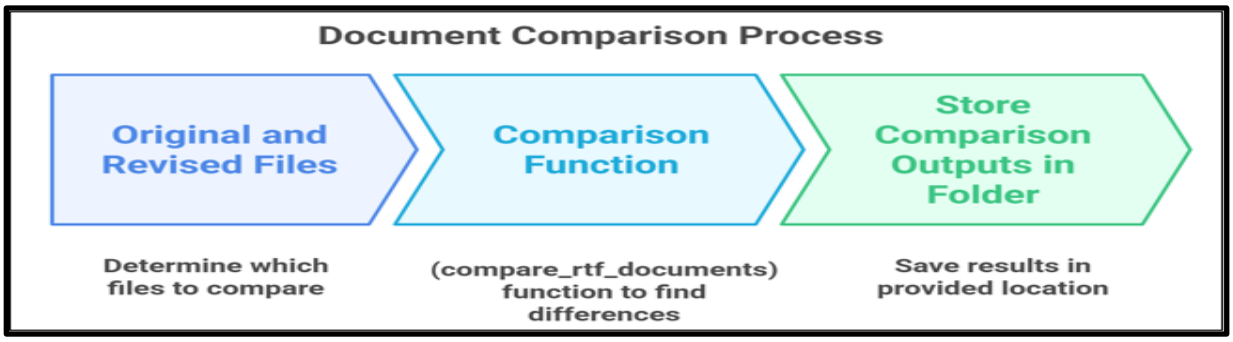
4. The fourth pointer is about **‘Output Present in 'V1_[old_version_date]’** & **‘Output Present in V2_[new_version_date]’** – These columns refer to whether the file exists in the original version and the revised version ("**Yes**" or "**No**").

5. The fifth pointer is about **“Page number discrepancy?”** - Additional notes, such as whether there’s a mismatch in the page counts or if specific content is missing.

STEP 4 – CONTENT DIFFERENCES

In the folder structure in step 2, referring to the folder named ‘Compared Outputs”, the tool generates the compared output files.

In this step, the tool compares files from the original and revised directories using Microsoft Word's built-in comparison functionality. It saves the comparison results in a specified output folder. A comparison is made to each file present in both directories, and the differences are highlighted in a new document.



Flowchart 2

Program 2 is the back-end Python code snippet for the document comparison functionality –

```
    def compare_documents(original_path, revised_path, compared_path):
     original_files = {f for f in os.listdir(original_path) if f.endswith('.rtf') and not
f.startswith('~$')}
     revised_files = {f for f in os.listdir(revised_path) if f.endswith('.rtf') and not
f.startswith('~$')}
     _, compare_folder, _ = create_output_folders(compared_path, original_files)
     word = initialize_word()
     for file_name in tqdm(original_files, desc="Comparing documents"):
         if file_name in revised_files:
             original_file = os.path.join(original_path, file_name)
             revised_file = os.path.join(revised_path, file_name)
             compared_file = os.path.join(compare_folder, file_name)
             odoc = word.Documents.Open(original_file)
             rdoc = word.Documents.Open(revised_file)
             ndoc = word.CompareDocuments(OriginalDocument=odoc, RevisedDocument=rdoc,
Destination=2)
             ndoc.SaveAs2(compared_file, FileFormat=6)
             ...
```

Program 2. compare_documents

Function Breakdown of Program 2:

1. Initialization:
    - Retrieves lists of files from the original and revised directories.
    - Calls create_output_folders() to create the necessary output folder structure.

2. File Comparison:
    - Iterates through the files in the original directory and checks for corresponding files in the revised directory.
    - Uses the Compare Documents method in Word to compare the two documents.
    - Saves the comparison result as a new file in the specified output folder.

3. Output:
    - The compare files are saved in the comparison folder.

Table 3 Baseline Demographics and Subject Characteristics – Safety Population			
Characteristic Statistic	**Drug A N = 9**	**Drug B N = 9**	**Total N = 18**
Sex			
Female	~~0~~2 (22.2%)	0	~~0~~2 (11.11%)
Male	~~9 (100~~7 (77.8%)	9 (100%)	~~18 (100~~16 (88.89%)

N = Number of subjects in Cohort; n = Number of subjects with demographic and baseline data;
Calculation of percentages is based on N.
SD = Standard Deviation.

Display 5. Details of comparison

The 'compare_documents' function identifies files in the specified directories. It compares files with matching names from the original and revised directories, leveraging Word's 'Compare Documents' method to generate documents that highlight the differences between the two versions. The resulting compared files are saved in the specified output directory (compared_path). The function includes robust error handling to catch exceptions during the comparison process, ensuring that errors do not interrupt the overall execution and allowing the function to proceed with the remaining files. Refer to the example provided **Display 5** for a comparison result that includes track changes.

STEP 5 – CONSOLIDATED COMPARISON REPORT

In the folder structure in step 2, refer to the folder named 'Comparison Report'; the tool generates the following files:

1) Individual PDF file for compared output
2) HTML Report

The flow of this step is as follows:

Flowchart 3

This function extracts tracked changes from files and converts them to PDF. It also generates an HTML report summarizing the changes, with clickable links to the pages where changes occur.

Program 3 is the Python code snippet for this functionality:

```
    def extract_tracked_changes_and_convert_to_pdf(input_folder, report_folder,
extracted_titles):
    output_file_name = "comparison.html"
    html_output = """<html><head><title>Tracked Changes Summary</title>...</html>"""
    word = initialize_word()
    for file_name in tqdm(rtf_files, desc="Processing files"):
        rtf_path = os.path.join(input_folder, file_name)
        pdf_name = os.path.splitext(file_name)[0] + ".pdf"
        pdf_path = os.path.join(report_folder, pdf_name)
        # Extract changes and create HTML report

        html_output +=
f"<tr><td>{file_name}</td><td>{title}</td><td>{status}</td><td>{page_links}</td></tr>"
    html_output += "</table></body></html>"
```

Program 3. extract_tracked_changes_and_convert_to_pdf

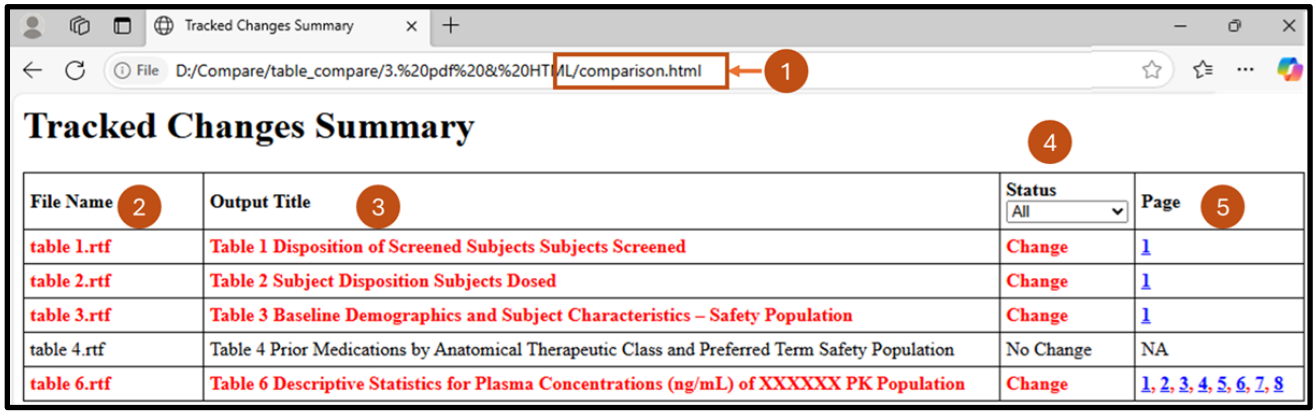
Function Breakdown of Program 3:

1. Initialization:
    - Retrieves the list of files from the input folder and prepares the HTML header for the report.

2. Processing Files:
    - For each file, it opens the document in Word, saves it as a PDF, and checks for tracked changes.
    - If changes are present, the function records the page numbers where changes occurred and adds them to the HTML output.

3. HTML Report Generation:
    - The function creates an HTML table summarizing the files, their titles, change status (**Change/ No Change**), and the page numbers of the changes.

4. Output:
    - Saves the HTML file and the corresponding PDF files in the specified report folder.



Display 6. PDF Files & Details of HTML Report

HTML Report Generation: Populates the report with details for each processed file:

1. The first pointer is about the saved comparison report.
2. The second pointer is about the File Name: Name of the processed file.
3. The third pointer is about the Output Title: extract the title from the ‘extracted_titles’ dictionary or a fallback message.
4. The fourth pointer is about the status: Indicates whether the file was "Updated" (has tracked changes) or "No Change". Users can use the dropdown option to select specific files where Change or No Change.
5. The fifth pointer is about the Page Links: Lists page numbers where tracked changes are present, linking directly to the corresponding pages in the PDF. NA stands for Not Applicable where the file has no differences.

PDF Conversion:

- Opens each file using Microsoft Word.
- Saves it as a .pdf in the specified report_folder using the Word file format for PDFs.

Tracked Changes Detection:

- Uses Word’s Revisions property to detect if the document contains tracked changes.
- Extracts page numbers where the changes occur and maps them to clickable links in the HTML generated report.

RESULTS AND DISCUSSION

The significance of this tool lies in its ability to provide a scalable and reliable framework for managing document workflows. It not only eliminates the limitations of manual processing but also enhances the accuracy and efficiency of document comparison tasks, making it a valuable tool for professionals in our industry.

The ‘Output Comparison Tool’ is a comprehensive tool designed to process, compare, and report differences between files efficiently. Its functionality spans across creating detailed Excel reports, comparing documents for differences, and generating summarized reports in HTML and PDF formats.

LIMITATIONS

Performance can also be an issue when processing a large number of files, as the sequential nature of the operations and the reliance on Word’s COM interface can result in slower processing times.

CONCLUSION

This tool provides significant value to statistical programmers and clinical research organizations (CROs) but also to pharmaceutical companies, regulatory agencies, and other industries requiring accurate and efficient comparison of outputs:

1. **Streamline Document Review:** Minimize the time and effort spent on manual comparisons of outputs by automating the identification of differences between original and revised documents.
2. **Enhance Accuracy and Consistency:** Reduce the likelihood of human error in detecting discrepancies across versions of tables, listings, and other critical outputs.
3. **Improve Traceability:** Generate detailed comparison reports, including Excel summaries and tracked change documents, to maintain clear documentation of revisions for audits and submissions.
4. **Boost Efficiency:** Enable statistical programmers to focus on core analytical tasks by simplifying routine document comparisons and automating report generation.
5. **Facilitate Collaboration:** Provide comprehensive output summaries and interactive HTML reports that can be easily shared with team members, sponsors, and stakeholders.
6. **Support Decision-Making:** Highlight critical differences in clinical outputs to assist in identifying significant changes and ensuring the integrity of reported data.

Numerous outputs are produced for a clinical study report, and there are instances when re-evaluation of these outputs is necessary. To minimize the requirement for the study team to review all outputs again, the method outlined in this paper efficiently and precisely identifies which outputs have changed and which have not. This tool demonstrated both accuracy and speed, capable of conducting an initial comparison of all outputs from a substantial Phase III study in a few minutes. For context, even opening two versions of a large listing can take longer.

REFERENCES

Python Official Documentation
Python Software Foundation. (2025). *Python Documentation*. Retrieved from https://docs.python.org/3/

pythoncom & win32com for Word Automation
Hammond, M., & Robinson, A. (2000). *Python Programming on Win32: Help for Windows Programmers*. O'Reilly Media.

OpenPyXL for Excel Manipulation
OpenPyXL Developers. (2025). *OpenPyXL Documentation*. Retrieved from https://openpyxl.readthedocs.io/en/stable/

TQDM for Progress Bars
Caswell, J., & Contributors. (2025). *TQDM: A Fast, Extensible Progress Bar for Python and CLI*. Retrieved from https://tqdm.github.io/

RTF Handling in Python
Various RTF-to-text conversion techniques based on:

- UnRTF: https://www.gnu.org/software/unrtf/
- StripRTF: https://pypi.org/project/striprtf/

Microsoft Word Automation using COM Objects
Microsoft Corporation. (2025). *Microsoft Office VBA Reference*. Retrieved from https://docs.microsoft.com/en-us/office/vba/api/overview/

CONTACT INFORMATION

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

Vinayak Mane
vinayak.mane@inferenceinc.com,
https://inferenceinc.com

Tanusree Bhattacharyya
tanusree@inferenceinc.com,
https://inferenceinc.com

Adel Solanki
adel.solanki@inferenceinc.com,
https://inferenceinc.com

Anindita Bhattacharjee
anindita@inferenceinc.com,
https://inferenceinc.com

Nibedita Hazra
nibedita.hazra@inferenceinc.com
https://inferenceinc.com

APPENDIX - DEMONSTRATION VIDEO

To provide a clear understanding of the functionalities and workflow of automated Python tool for outputs comparison and reporting, we have prepared a demonstration video. This video showcases the step-by-step process, including input handling, comparison execution, and output generation. To watch the demonstration video, please visit:

https://drive.google.com/file/d/1Byd7HtuH6A4KR4E-RG4RU4KofHeOs6Gv/view?usp=sharing