Paper SM01

Optimizing Output Review: A Targeted and Scalable RTF Validation Framework

Siddharth Kumar, Navitas Data Sciences, Pottstown, USA

ABSTRACT

Clinical Programming and Biostatistics teams often work under compressed timelines while delivering submission-ready Tables, Listings, and Figures (TFLs). Independent double programming remains the gold standard for validating analysis datasets; however, the review of final Rich Text Format (RTF) outputs is still largely manual. This becomes particularly challenging during CRO-to-sponsor handoffs, where extensive TFL packages must be reviewed within just a few days. Traditional tertiary QC methods, such as converting RTF files back into SAS datasets and performing PROC COMPARE, demand considerable effort and still fail to identify formatting and presentation-level discrepancies.

This paper introduces a scalable RTF Validation Framework that integrates SAS, R, and Python to evaluate RTF outputs directly and detect differences across deliveries. The framework automates key review tasks, including change detection, validation of population counts (Big N), enforcement of decimal precision rules, and consistency checks for titles and footnotes. Rather than requiring reviewers to manually inspect every output, the system highlights only meaningful discrepancies, allowing teams to focus their attention on priority tasks.

To further enhance transparency and collaboration, validation results are automatically published to interactive dashboards through Smartsheet API integration. This enables centralized visibility, structured issue tracking, and near real-time alignment between Programming and Biostatistics teams. By combining automated validation with dashboard-driven reporting, the framework creates a more efficient, traceable, and audit-ready approach to output review in regulated clinical environments.

INTRODUCTION

Delivering high-quality Tables, Listings, and Figures (TFLs) is one of the most visible responsibilities of Programming team. These outputs ultimately represent the trial’s story to regulators, reviewers, and stakeholders. While statistical rigor is built into dataset validation through independent double programming, the final presentation layer the RTF outputs - often depends on careful manual review.

In practice, reviewing hundreds of RTF outputs is time-consuming and mentally demanding. Programmers must confirm population counts, check decimal precision against the Statistical Analysis Plan (SAP), review alignment and formatting, verify titles and footnotes, and ensure that no macro variables remain unresolved. These checks are critical, yet they are repetitive and susceptible to human oversight, especially under tight timelines.

The challenge becomes even more evident during CRO-to-sponsor handoffs. Sponsors are frequently given only a few days to perform a comprehensive review of an entire TFL package. Under such constraints, teams may rely on selective QC approaches, prioritizing certain outputs while accepting residual risk elsewhere. While practical, this approach does not provide a consistent or scalable validation strategy.

To address this gap, a framework is needed that validates RTF outputs directly, minimizes manual effort, and provides structured, traceable documentation without disrupting established statistical workflows.

CHALLENGES IN RTF OUTPUT REVIEW

Traditional QC processes focus on dataset-level accuracy. Independent programming and PROC COMPARE ensure that derived variables and statistical summaries are correct. However, many presentation-related risks arise after datasets are finalized during PROC REPORT execution and RTF generation.

Common challenges include incorrect assignment of treatment population counts (Big N), misaligned columns, inconsistent decimal precision, missing or duplicated footnotes, unresolved macro variables, and unintended formatting changes between deliveries. These issues may not affect the underlying data but can significantly impact interpretation, credibility, and regulatory perception.

Version management presents another difficulty. With multiple data cuts and iterative deliveries, reviewers must identify what has changed between versions. Manual side-by-side comparison of RTF files is inefficient and increases the likelihood of missed differences. Without structured comparison tools, teams spend valuable time reviewing unchanged outputs.

These operational realities highlight the need for automation that operates directly at the RTF level while complementing, not replacing, statistical validation.

OBJECTIVES OF THE FRAMEWORK

The RTF Validation Framework was designed with practical implementation in mind. Its primary goals are to detect meaningful differences between TFL deliveries, automate high-risk presentation checks, reduce manual review burden, and generate structured, audit-ready documentation. The framework must integrate smoothly with existing SAS-based production environments while leveraging modern tools such as R and Python for enhanced flexibility.

Equally important, the system must support collaboration. Validation findings should not remain isolated in individual spreadsheets but instead be shared transparently across Programming and Biostatistics teams through a centralized and controlled reporting mechanism.

OVERVIEW OF THE RTF VALIDATION FRAMEWORK

The framework integrates three primary components: Python for automated RTF comparison, R for structured validation checks and application logic, and Smartsheet API integration for dashboard-driven reporting. SAS remains the production engine for statistical outputs, preserving standard development practices.

The workflow begins by reading RTF files from two delivery folders, typically representing a previous and current version. A Python-based comparison engine extracts readable text from each RTF file and performs a systematic line-level comparison. Differences are identified, structured, and categorized.

The structured output is then processed through R-based validation routines. These routines perform targeted checks such as verifying population counts, confirming decimal precision compliance, and evaluating titles and footnotes. The resulting findings are compiled into a structured dataset.

Finally, the validation dataset is transmitted via the Smartsheet API to populate interactive dashboards. This integration provides centralized visibility, structured issue tracking, and near real-time collaboration.

This layered approach allows teams to move from manual inspection to automated detection, while preserving reviewer oversight where it matters most.

PYTHON-BASED VERSION COMPARISON

One of the most impactful components of the framework is automated version comparison. Instead of visually comparing two RTF files side by side, the system extracts textual content and evaluates differences programmatically.

For each file in a prior delivery folder, the corresponding file in the current delivery is identified. Text is extracted using RTF parsing libraries, and a line-by-line comparison is performed. Differences are captured, classified, and summarized into a structured report.

This process enables reviewers to focus exclusively on outputs where changes occurred. In large studies with hundreds of tables, this significantly reduces review time and improves consistency. Importantly, the comparison operates directly on CRO-delivered files, eliminating the need to recreate datasets solely for tertiary comparison.

R-BASED STRUCTURED VALIDATION

Beyond change detection, the framework performs targeted validation checks within R to address high-risk presentation elements.

For population count validation, the system parses header sections of RTF outputs to extract treatment labels and corresponding Big N values. These values are compared to expected counts derived from validated datasets. Any discrepancy is flagged automatically.

Decimal precision validation ensures that numerical values conform to SAP-defined rules. The system evaluates detected decimal places and compares them to predefined standards. Inconsistencies are logged systematically, reducing reliance on manual spot-checking.

Title and footnote validation focuses on clarity and completeness. The system scans for unresolved macro variables, performs spell checks, and evaluates structural consistency across outputs. These checks help ensure professional presentation and reduce downstream reviewer comments.

Together, these automated checks replace repetitive manual tasks with structured, repeatable logic.

SMARTSHEET API INTEGRATION AND DASHBOARD REPORTING

A key strength of the framework lies in its ability to communicate results effectively. Rather than producing static spreadsheets that must be manually distributed, validation findings are transmitted programmatically to Smartsheet via secure API integration.

Each discrepancy is recorded with relevant metadata, including output name, issue type, detected difference, and validation category. The dashboard allows users to filter by table, issue classification, or delivery version. Severity tagging enables prioritization, and structured tracking supports resolution workflows.

This approach promotes transparency and shared ownership of findings. Programming and Biostatistics teams can view the same information in real time, reducing back-and-forth communication and aligning expectations. Additionally, the dashboard provides structured documentation suitable for audit and inspection purposes.

BENEFITS AND PRACTICAL IMPACT

The framework is designed to complement existing validation standards. It does not replace independent double programming or statistical QC; instead, it strengthens presentation-level controls.

In practical use, the framework reduces the number of outputs requiring manual review, accelerates identification of meaningful differences, and standardizes validation findings across reviewers. Teams gain confidence that presentation-level risks are systematically evaluated rather than selectively reviewed.

The approach scales effectively to large TFL packages and multiple data cuts. It supports CRO-to-sponsor transitions, integrated summaries, and submission-readiness activities without requiring fundamental workflow disruption.

Most importantly, the framework allows programmers and statisticians to focus on interpretation and analysis rather than repetitive inspection tasks.

FUTURE ENHANCEMENTS

While the current framework substantially improves efficiency and consistency in RTF validation, several enhancements could further expand its impact and scalability.

A natural next step is the introduction of intelligent anomaly detection. Moving beyond rule-based checks, future iterations could leverage machine learning techniques to identify unusual patterns in outputs, such as unexpected shifts in summary statistics, structural layout changes, or formatting anomalies that fall outside predefined rules. By

learning from historical deliveries, such models could flag deviations that warrant closer human review.

Another advancement involves semantic change classification. Not all differences between TFL versions carry equal significance. Future enhancements could categorize detected changes based on clinical or operational impact, distinguishing formatting-only updates from content-level statistical changes. This would enable reviewers to prioritize high-impact modifications and further streamline review cycles.

Dashboard capabilities could also evolve. Incorporating automated assignment workflows, comment tracking, and resolution status updates within the Smartsheet environment would transform the dashboard from a reporting interface into a fully managed review platform.

Finally, exploration of Generative AI may offer meaningful support. AI-driven summarization of detected differences or automated narrative explanations of changes could help reviewers quickly understand the context of updates. Any implementation in regulated environments, however, must prioritize data security, controlled access, validation standards, and compliance with governance policies.

CONCLUSION

As clinical trials grow in complexity and timelines continue to compress, efficient and reliable output validation becomes increasingly important. While dataset-level validation remains essential, presentation-level review of RTF outputs demands additional attention.

The RTF Validation Framework described in this paper integrates SAS, R, Python, and Smartsheet API-driven dashboards to automate change detection and structured presentation checks. By highlighting meaningful discrepancies and centralizing results within interactive dashboards, the framework transforms output review from a manual, time-intensive process into a scalable and collaborative system.

This approach enhances efficiency, strengthens traceability, and supports audit readiness allowing Clinical Programming and Biostatistics teams to deliver high-quality, submission-ready outputs with greater confidence.

REFERENCES

Kumar S. Efficient TFL Review within Statistical Programming – Using R and SAS. PHUSE Paper ETD02.

R: A language and environment for Statistical Computing

Chang W, Cheng J, Allaire J, Sievert C, Schloerke B, Xie Y, Allen J, McPherson J, Dipert A, Borges B (2024). _shiny: Web Application Framework for R_. R package version 1.10.0, https://CRAN.R-project.org/package=shiny

Mori K (2023). _striprtf: Extract Text from RTF File_. R package version 0.6.0, https://CRAN.R- project.org/package=striprtf