Paper OS14

From SAS to R for TLGs: A Case Study on Implementing an 'R-First, SAS-QC' Workflow for Time-to-Event Analysis

Adler Moisés Fallas Rodríguez, Intego Clinical, San José, Costa Rica

ABSTRACT

For decades, SAS has been the primary language for generating clinical trial Tables, Listings, and Graphs (TLGs). With the increasing adoption of R in the pharmaceutical industry, our team faced the challenge of transitioning to open-source tools while maintaining compliance and quality standards. We implemented a workflow where R is the first-line tool for TLG generation, with SAS serving as an independent benchmark for quality control (QC). This dual-programming approach leveraged R's flexibility and transparency while relying on SAS for validation and regulatory alignment. As an example, we illustrate a Time-to-Event summary TLG. In R, we applied packages supporting metadata-driven table construction and automated formatting. The same key derivations —patient counts, survival estimates, and hazard ratios—were independently replicated in SAS, providing validation. By adopting this "R-first, SAS-QC" strategy, we accelerated TLG production for survival analysis without compromising confidence or regulatory reassurance, showing that organizations can innovate with R while preserving compliance.

INTRODUCTION

For many years, SAS has been the standard to produce clinical trial Tables, Listings, and Graphs (TLGs). Its widespread adoption, long regulatory history, and well-established validation practices have made it the default choice for clinical reporting. At the same time, the use of R has grown rapidly across the pharmaceutical industry, particularly for statistical analysis, exploratory work, and increasingly for production-grade outputs. This shift is driven by the flexibility of open-source tools, improved reproducibility, and the availability of mature R packages designed specifically for clinical trial reporting.

Despite this adoption, full transition of TLG production to R remains challenging. The primary obstacle is not statistical capability, as R readily supports standard clinical methods such as Kaplan–Meier estimation and Cox proportional hazards modeling. Rather, the challenge lies in operational confidence: ensuring that outputs generated in R meet the same quality expectations, traceability, and validation standards traditionally associated with SAS-based workflows. In many organizations, this concern has limited R to an exploratory or secondary role, while SAS remains the primary engine for final deliverables.

Dual programming has long been an accepted quality control (QC) strategy in clinical reporting. However, in practice, dual programming is often implemented within the same language or within tightly coupled frameworks, which can reduce its effectiveness as independent verification. When R is introduced into production workflows, a key design question emerges: how can R be used as the primary tool for TLG generation while maintaining a robust, regulatorily defensible QC process?

This paper presents a practical solution to this problem through an "R-first, SAS-QC" workflow. In this approach, R serves as the primary production engine for TLGs, while SAS is used independently to replicate key derivations and validate results. The intent is not to replace SAS entirely, but to reposition it as a dedicated QC tool that provides confidence through independent statistical re-derivation rather than through formatting or visual comparison alone.

To illustrate this approach, we focus on a Time-to-Event summary TLG, a class of output that is both statistically rich and operationally sensitive. Time-to-Event tables typically combine multiple components, including patient counts, event classifications, Kaplan–Meier summaries, Cox model estimates, and time-point survival rates. These outputs therefore provide an ideal case study for demonstrating how complex statistical content can be produced in R and independently validated in SAS.

The scope of this paper is intentionally practical. Rather than comparing statistical methods or advocating for a specific software ecosystem, we describe the architecture of a real production workflow, including metadata-driven processing in R, independent statistical replication in SAS, and a structured comparison strategy to assess concordance between systems. By grounding the discussion in an operational case study, this paper aims to demonstrate that an R-first approach to TLG production can be implemented in a controlled, auditable, and scalable manner while preserving the level of confidence expected in clinical reporting environments.

SYSTEM OVERVIEW: R-FIRST, SAS-QC WORKFLOW

Overall Architecture

The proposed workflow is built around a clear separation between production and validation activities. R is used as the primary engine for generating clinical trial TLGs, while SAS is used exclusively for independent quality control. This architectural separation is intentional and ensures that statistical results are derived independently using different software environments, reducing the risk of shared implementation errors.

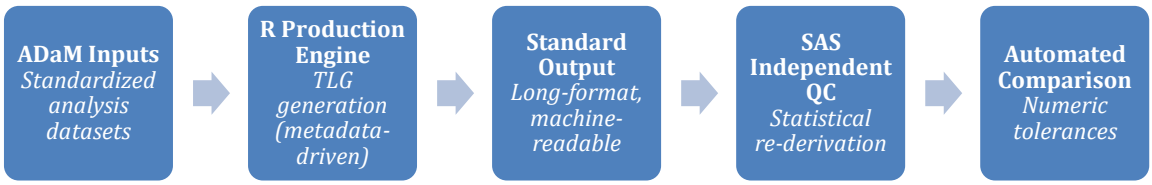

Figure 1. R-first, SAS-QC workflow for Time-to-Event TLG production and validation. Standardized ADaM inputs are processed in R to generate TLGs and a standard long-format output, which is independently re-derived and validated in SAS.

Standardized ADaM datasets serve as the common input to both systems. The R pipeline produces the full, presentation-ready TLG, while the SAS pipeline independently re-derives the same statistical quantities and assembles them into a structure suitable for automated comparison. The workflow is designed to be fully script-driven and compatible with automated orchestration.

Role of R in TLG Production

Within this framework, R functions as the production reporting engine. Input ADaM datasets are processed using metadata that define analysis populations, parameter-specific filters, formatting rules, and reporting conventions. This metadata-driven approach allows multiple variations of the same TLG to be generated without changes to core program logic.

The Time-to-Event TLG is constructed declaratively using table layout specifications that define row and column splits, statistical analyses, and labeling. Survival summaries, Cox proportional hazards models, and time-point estimates are generated directly within the table construction process. In addition to the formatted table, R produces structured outputs that capture the numerical content of the TLG in a machine-readable form, enabling downstream validation.

An example of a declarative table specification in R is shown below.

```
lyt <- basic_table(show_colcounts = TRUE) %>%
  split_cols_by("TRTGRP", ref_group = "Control") %>%
  analyze_vars(vars = "is_event", .stats = "count_fraction") %>%
  surv_time(
    vars = "AVAL",
    var_labels = "Time to Event (Months)",
    is_event = "is_event",
    table_names = "time_to_event"
  ) %>%
  coxph_pairwise(
    vars = "AVAL",
    is_event = "is_event",
    var_labels = "Stratified Analysis",
    strata = c("STRAT1", "STRAT2", "STRAT3", "STRAT4"),
    table_names = "coxph_stratified",
    ties = "exact"
  )
```

Role of SAS in Independent QC

SAS is used to independently replicate the key statistical derivations produced in R. The QC programs operate directly on the same ADaM inputs and apply standard SAS procedures to compute patient counts, event summaries, Kaplan–Meier estimates, log-rank tests, and Cox model hazard ratios.

Rather than attempting to recreate the final visual appearance of the TLG, the SAS programs focus on re-deriving the underlying statistics. The results are assembled into a long-format dataset that contains identifiers for treatment group, statistic type, and table row. This structure is designed specifically to support automated numerical comparison with the R-derived outputs.

R–SAS Bridging Layer and Comparison Strategy

A key component of the workflow is the explicit bridging layer between R and SAS. Both systems converge to a common comparison format that represents statistical results independently of presentation details such as formatting, rounding, or layout. This approach minimizes false discrepancies caused by rendering differences and ensures that QC efforts focus on statistical agreement.

The bridging dataset generated in R is persisted as a platform-neutral Parquet file. In SAS, this same file is subsequently consumed using standard, non-proprietary interfaces, either through native Parquet readers where available or via a lightweight conversion step into a temporary SAS dataset (for example, using a Python-based conversion with commonly available open-source libraries for Parquet ingestion and SAS dataset export). In both cases, the dataset is

treated as a conventional long-format input for independent re-derivation and comparison, without requiring shared derivation logic, metadata, or computational code paths between R and SAS.
The specific technical mechanism used for Parquet ingestion in SAS is intentionally decoupled from the workflow design, allowing environments to use native readers or lightweight open-source conversion layers without impacting validation logic.
To make QC scalable and presentation-agnostic, we define an explicit comparison contract: both R and SAS output the TLG's numerical content in the same long-format structure. Each row represents a single statistic and is keyed by stable identifiers rather than by rendered table positions.

A minimal schema is shown below.

Variable	Description	Example
TRT	Treatment column label (as displayed in the TLG)	Treatment A
ROW_NUM	Row identifier aligned to the TLG specification	13
LABEL_NAME	Human-readable row label	Hazard Ratio
STAT	Statistic type	count / percent
VAL	Numeric value to compare	0.82

In R, this dataset is generated directly from the built table object (e.g., via as_result_df()), reshaped to a unified long format, and persisted as a Parquet file using the arrow package. In SAS, the same Parquet file is read to obtain the R-derived reference values, while SAS independently re-derives the corresponding statistics from ADaM inputs and assembles them into an equivalent long-format dataset. Automated comparison is then performed by matching on (TRT, ROW_NUM, LABEL_NAME, STAT) and applying predefined numeric tolerances to VAL.

A simplified illustrative example of this structure is shown below.

TRT	ROW_NUM	LABEL_NAME	STAT	VAL
Treatment A	1	Patients with event (%)	count	42
Treatment A	1	Patients with event (%)	percent	35.0
Control	1	Patients with event (%)	count	51
Control	1	Patients with event (%)	percent	43.2
Treatment A	2	Median Time to Event (Months)	count	14.8
Control	2	Median Time to Event (Months)	count	12.3

This simplified example illustrates how multiple TLG rows (ROW_NUM) are expanded into a unified long-format structure for automated comparison; values are illustrative only.

The following example illustrates how the R output is reshaped into a standard long format suitable for automated comparison.

```
tbl_df <- as_result_df(final_tlg)
trt_cols <- colnames(as_result_df(final_tlg, simplify = TRUE))[-1]
common_cols <- colnames(tbl_df)[!(colnames(tbl_df) %in% trt_cols)]

qc_df <- lapply(trt_cols, function(trt) {
  tmp <- tbl_df[, c(common_cols, trt)]
  names(tmp)[names(tmp) == trt] <- "VAL"
  tmp$STAT <- list(c("count", "percent"))
  tmp$TRT <- trt
  tmp
}) |>
  do.call(rbind, _) |>
  tidyr::unnest(cols = c(VAL, STAT))

arrow::write_parquet(qc_df, file.path("output", "tte_qc.parquet"))
```

Automated comparisons are performed using predefined tolerances to account for minor floating-point differences arising from distinct computational implementations. By isolating comparison logic from table generation, the workflow enables consistent and repeatable QC across multiple outputs and parameterizations.

Together, these design elements allow R to function as a production-grade TLG engine while preserving the independent verification role traditionally fulfilled by SAS, providing a balanced and scalable solution for clinical reporting workflows.

CASE STUDY: TIME-TO-EVENT TLG

Scope and Purpose of the TLG

The Time-to-Event TLG selected for this case study summarizes key efficacy endpoints derived from survival-type analyses. This class of output is commonly used in oncology trials and other therapeutic areas where the timing of an event is clinically meaningful. The table is designed to provide a comprehensive overview of treatment effects while combining multiple statistical components within a single, structured output.

The purpose of this TLG is twofold. First, it serves as a clinically interpretable summary of time-to-event outcomes across treatment groups. Second, from an operational perspective, it represents a complex reporting artifact that is well suited for evaluating the feasibility of an R-first production workflow with independent SAS-based QC.

Statistical Components of the Output

The TLG integrates several categories of statistics derived from the same underlying analysis dataset. These include patient counts and proportions for events and non-events, a breakdown of earliest contributing event types, and descriptive summaries of time-to-event distributions based on Kaplan–Meier estimation.

In addition, the table includes inferential components such as stratified and unstratified Cox proportional hazards model estimates, as well as time-point–specific survival metrics at predefined analysis time points. These elements are typically subject to close scrutiny during validation due to their dependence on censoring logic, time-scale definitions, and model specifications.

By combining descriptive summaries, model-based estimates, and time-point analyses within a single output, the Time-to-Event TLG provides a representative and demanding test case for the proposed workflow. Successful replication and validation of this table across independent R and SAS implementations demonstrates that the approach can accommodate both statistical complexity and operational rigor.

R IMPLEMENTATION

Metadata-Driven Data Preparation

The R implementation begins with standardized ADaM datasets as inputs and applies analysis-specific logic through external metadata. Filters, formatting rules, and reporting conventions are defined outside the core program logic and are applied consistently across outputs. This approach allows the same program to support multiple parameterizations of the Time-to-Event TLG by modifying metadata rather than code.

Analysis datasets are subset to the appropriate analysis population, and time-to-event variables are transformed to the reporting scale prior to table construction. Event indicators and descriptive labels are derived in a transparent and reproducible manner, ensuring that downstream analyses operate on clearly defined inputs.

Declarative Table Construction

The Time-to-Event TLG is constructed using a declarative table specification. Row and column structures, statistical analyses, and labeling are defined as part of the table layout rather than through sequential data manipulation. This includes the definition of treatment group columns, event and non-event summaries, breakdowns by event type, survival summaries, Cox model analyses, and time-point–specific statistics.

Survival analyses and regression models are executed directly within the table-building process, ensuring that the reported statistics remain tightly coupled to the table structure. This design reduces the risk of inconsistencies between derived statistics and their placement in the final output and supports reproducible regeneration of the TLG.

Time-point–specific survival estimates are defined directly within the table specification, as illustrated below.

```
surv_timepoint(
  vars = "AVAL",
  time_point = c(6, 12),
  is_event = "is_event",
  .stats = c("pt_at_risk", "event_free_rate", "rate_ci")
)
```

Automated Output Generation

In addition to producing a presentation-ready table, the R pipeline generates multiple machine-readable outputs that capture the numerical content of the TLG. These outputs include structured datasets designed for downstream validation and auditability.

By exporting the table results in a standardized long format, the implementation enables automated comparison with independently derived QC outputs. This design choice decouples statistical validation from table rendering and supports scalable QC across multiple outputs and reporting scenarios.

The R implementation relies exclusively on publicly available open-source packages, including tern, rtables, citril, arrow, dplyr, and tidyr, along with related ecosystem tools used for data manipulation and table construction. No proprietary or organization-specific libraries are required, ensuring that the workflow can be reproduced in external environments without access to internal infrastructure.

INDEPENDENT QC IN SAS

Re-Derivation of Key Statistics

The SAS QC implementation independently reproduces the key statistical quantities reported in the R-generated Time-to-Event TLG using the same ADaM inputs. Treatment assignments are mapped to QC-friendly labels, and the QC dataset is restricted to the appropriate analysis population prior to computation. Core table components are then re-derived using standard SAS procedures and explicit calculations, focusing on statistical concordance rather than table rendering.

Event and non-event counts and percentages are computed directly from the analysis dataset. Kaplan–Meier estimation is performed using PROC LIFETEST to obtain survival summaries, quartiles (including the median), and time-point estimates at predefined time horizons. Stratified and unstratified log-rank tests are generated through PROC LIFETEST stratification options, while hazard ratios and confidence intervals are obtained from Cox proportional hazards models via PROC PHREG, including stratified and unstratified specifications.

The following example illustrates how key Time-to-Event statistics are independently re-derived and compared in SAS.

```
ods output ProductLimitEstimates=pt_est;
proc lifetest data=adtte_qc timelist=(&T1 &T2) conftype=linear plots=none;
  strata TRTGRP;
  time aval_m*cnsr(1);
run;

ods output HazardRatios=hr_est;
proc phreg data=adtte_qc;
  class TRTGRP(ref='Control');
  model aval_m*cnsr(1)=TRTGRP / ties=exact;
  hazardratio TRTGRP / cl=wald;
  strata STRAT1 STRAT2 STRAT3 STRAT4;
run;

proc compare base=r_qc compare=sas_qc
  method=absolute criterion=0.001 listall;
  id TRT row_num stat label_name;
run;
```

This approach ensures that the QC program functions as an independent replication of statistical derivations rather than a post-processing check of formatted outputs.

Assembly of QC Datasets

To support automated comparison with the R-derived results, SAS assembles the independently computed statistics into a standard long-format dataset. Each record includes identifiers for treatment group, table row, statistic type (for example, count vs percent), and the corresponding numeric value. This structure is designed to align with the machine-readable output exported from the R pipeline.

By organizing the QC results into a stable comparison format, the workflow enables systematic validation through automated tools (e.g., PROC COMPARE) with predefined tolerances. This design supports repeatability across multiple parameterizations of the TLG and reduces the risk of false discrepancies driven by presentation differences.

R–SAS BRIDGING AND COMPARISON STRATEGY

The R-first, SAS-QC workflow relies on an explicit bridging layer that enables direct numerical comparison between independently derived results. Rather than comparing formatted tables or rendered outputs, both systems are designed to converge to a common, machine-readable representation of the TLG content. This design choice isolates statistical validation from presentation concerns and ensures that QC focuses on the underlying computations.

On the R side, the numerical results embedded in the table are exported to a structured long-format dataset that identifies treatment group, row position, statistic type, and value. This representation captures the full statistical content of the TLG while remaining agnostic to layout, formatting, or rounding applied for presentation purposes. The same conceptual structure is reconstructed in SAS after independent re-derivation of the statistics.

Once both systems produce comparable datasets, automated comparison is performed using stable identifiers as keys. Predefined numeric tolerances are applied to accommodate minor floating-point differences arising from distinct computational implementations, particularly for survival estimates, confidence intervals, and model-based statistics. By standardizing the comparison layer, the workflow minimizes false discrepancies and enables consistent QC across multiple outputs.

This bridging strategy transforms QC from a manual, table-centric exercise into a reproducible and scalable validation process. It allows independent implementations in R and SAS to be evaluated objectively, reinforcing confidence in the R-generated TLGs while preserving the independence required for robust quality control.

VALIDATION RESULTS AND TOLERANCES

Validation was performed by comparing the independently derived R and SAS outputs using the standard long-format datasets produced by each system. Comparisons were executed at the level of individual statistics, keyed by treatment group, table row, and statistic type, ensuring that each numerical value in the Time-to-Event TLG was evaluated independently.

For discrete quantities such as patient counts and event frequencies, exact agreement between R and SAS was observed. For continuous statistics derived from survival analyses and regression models, small numerical differences were expected due to differences in internal computational implementations and rounding behavior, as documented in cross-software comparisons of statistical methodology implementations across programming languages (Jen et al., 2023). To account for these effects, an absolute tolerance threshold was applied during automated comparison.

Within these predefined tolerances, the R and SAS results demonstrated consistent concordance across Kaplan–Meier estimates, Cox model hazard ratios, confidence intervals, and time-point survival metrics. Observed discrepancies outside tolerance thresholds were traceable to definitional or implementation differences rather than computational error, highlighting the importance of consistent time-scale definitions, population filters, and censoring logic across systems.

Overall, the validation results support the use of the R-first, SAS-QC workflow as a reliable approach for producing Time-to-Event TLGs. The combination of independent statistical re-derivation and structured automated comparison provides a level of assurance comparable to traditional dual-programming approaches while enabling greater flexibility and efficiency in production.

OPERATIONAL CONSIDERATIONS AND LESSONS LEARNED

Implementing an R-first, SAS-QC workflow for Time-to-Event TLGs highlighted several operational considerations that extend beyond statistical methodology. Chief among these is the importance of designing production code with validation in mind. Exporting machine-readable results and defining a stable comparison contract early in development greatly simplified downstream QC and reduced ambiguity during reconciliation.

Metadata-driven processing proved essential for scalability. By externalizing filters, time-point definitions, and formatting rules, the same production and QC logic could be applied consistently across multiple endpoints and parameterizations. This reduced code duplication and facilitated controlled updates when reporting requirements evolved.

Another key lesson was the need for strict alignment of analysis definitions across systems. Minor inconsistencies in population flags, time-scale transformations, or censoring conventions can produce discrepancies that are difficult to diagnose at the comparison stage. Explicit documentation and early validation of these assumptions were critical to maintaining concordance between R and SAS outputs.

Finally, the workflow demonstrated that QC should be treated as a first-class component of the reporting pipeline rather than a downstream check. By formalizing independent re-derivation and automated comparison, the process became more transparent, reproducible, and amenable to automation, supporting both efficiency and confidence in production TLGs.

CONCLUSION

This paper demonstrates that an “R-first, SAS-QC” approach to the production of Time-to-Event TLGs can be successfully implemented without compromising the level of quality control expected in clinical reporting. By positioning SAS as an independent QC engine rather than a parallel production tool, the proposed workflow preserves the benefits of open-source reporting while maintaining confidence through established validation practices.

The case study shows that complex survival-based outputs can be generated reproducibly in R using metadata-driven and declarative table construction and independently validated in SAS through full statistical re-derivation. Convergence to a common comparison layer enables objective, automated assessment of concordance and reduces reliance on presentation-level or manual checks.

Based on this experience, we recommend designing R production workflows with QC and comparability as explicit requirements from the outset. When combined with independent SAS-based validation, this approach provides a practical and scalable pathway for expanding the use of R in production TLGs while meeting operational and regulatory expectations.

ACKNOWLEDGMENTS

The author would like to thank Intego Clinical for providing the professional environment in which this work was developed, and the Intego Costa Rica team for their technical collaboration and peer discussions throughout the implementation process. Special thanks are extended to Yevhen Los for his ongoing guidance and constructive feedback throughout the author’s development as a statistical programmer.

CONTACT INFORMATION

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

Author Name: Adler Moisés Fallas Rodríguez
Company: Intego Clinical
Address: San Pedro Business Center, Carr. Interamericana Sur, San José, San Pedro, Costa Rica.
Email: moises.rodriguez@intego-group.com
Website: https://integogroup.com/

REFERENCES

Jen, M-H., Varney, B., Lee, K., Arancibia, B., Qi, M., Taylor, L., Rickert, J., Stackhouse, M., Rimler, M. (2023). *Key Considerations when Understanding Differences in Statistical Methodology Implementations across Programming Languages – An Introduction to the CAMIS Project*. PHUSE/PSI White Paper.