**Clinical Statistical Reporting in a Multilingual World**

**Introduction motivation**

**This is the problem white paper addresses (CLEAR and UPFRONT)**

**Background**

**Framework**

**Framework applied to use case**

**Call to action**

**Any language and comparison we hope this framework generalizes too.**

# **Introduction**

Over the last decade, there has been a dramatic rise in the availability of advanced analytic capabilities, through both hardware and software. There is a growing interest in leveraging a broader set of tools than has historically been used to analyze clinical trial data. The increased variety of tools has resulted in an overlap of capabilities, which has raised challenging questions of traditional approaches to clinical analyses – particularly in situations where the overlap yields different results.

One example of this challenge encompasses discrepancies which have been discovered in statistical analysis results between different programming languages, even when working within qualified statistical computing environments. Subtle differences exist between the fundamental approaches and assumptions implemented within each language, yielding differences in results which are correct and consistent with their respective documentation. The fact that these differences exist may cause unease for sponsor companies when submitting to a regulatory agency, as it is uncertain if the agency will view these differences as problematic. In its [Statistical Software Clarifying Statement](https://www.fda.gov/media/109552/download), the US Food and Drug Administration (FDA) states that the “FDA does not require use of any specific software for statistical analyses” and that “the computer software used for data management and statistical analysis should be reliable.” Observing differences across languages can reduce the analyst’s confidence in the reliability of results. By understanding the source of any discrepancies, one can reinstate that confidence.

This white paper aims to empower analysts to make informed choices on the implementation of statistical analyses when multiple languages yield different results. Our objective is not to prescribe what that choice should be, but rather provide guidance on the types of questions an analyst should ask in order to identify the fundamental sources of discrepant results. These discrepancies may exist for a variety of different reasons, of which this paper will explore and provide examples. In this context, the risk of interpreting numerical differences in analysis results due solely to differences in programming language can be mitigated, instilling confidence in both the sponsor company and the agency during the review period.

# **Background**

As clinical data analytics evolves within the pharmaceutical industry, a large and noteworthy contingent of people and organizations have explored the use of various computational technologies as an effort to reimagine how to tell the story about the data that is collected during the course of a clinical trial. These technologies, whether available commercially or as open source, offer new potential in the ability of a sponsor company to discover new medicines and demonstrate that they can be safely and effectively administered to patients for a given indication. We see applications of machine learning and artificial intelligence being built into exploratory analyses as well as automation of conventional reporting pipelines. We are witnessing a desired transformation of how we deliver clinical insights from flat data files with rows/columns and compiled PDF reports into dynamic visualization platforms which facilitate a reviewer to explore the trial database in a three-dimensional way. And, most notably, because the tools that other industries most commonly used for these ‘new’ ways of data engineering, data analytics, and data reporting are often built on programming languages not historically used within the pharmaceutical industry, we are experiencing a dramatic shift away from dependence on a small set of commercially available solutions and toward embracing many languages to build and use the best-fit tools to extract the most knowledge from clinical data.

This last piece has brought to light an element of our data analytics that was previously overlooked due to an overdependence on a single solution from one programming language. Within the clinical reporting pipeline (transforming patient level clinical trial data from collection to submission), the industry has predominantly relied on comparing results to an independently generated second set of results as the primary form of quality control (QC). In the early years, comparisons were made on paper and thoroughly verified by a human that the number in the table matched the number independently derived by a second programmer. As technology progressed, electronic comparisons of the output data presented in a table reduced the risk of human error that the validator missed a discrepancy. The theory is that if two people put the same inputs through two independently developed processes (the code) and achieve the same outcome, then the outcome must be right. It’s not a perfect system and it can produce false positives, but efficiencies were gained and quality improved.

However, up until recently, the QC process has nearly always been implemented with the same programming language being used both for the generation of results (‘on production’) and for independent QC. The shift in the industry to explore other languages has now raised questions such as “What if the numbers don’t match? Which is correct?”

For example, if one were to take a use case to compare rounding rules between SAS® and R, it is now becoming well understood that the default rounding rule (implemented in the respective language’s **round()** function) are different, but only when the number being rounded is equidistant between the two possible results. The **round()** function in SAS will round the number ‘away from zero’, meaning that 12.5 rounds to the integer 13. The **round()** function in Base R will round the number ‘to even’, meaning that 12.5 rounds to the integer 12. SAS also has the **rounde()** function which rounds to even and the **janitor** package in R contains a function that rounds away from zero. In this use case, SAS produces a correct result from its **round()** function, based on its documentation, as does R. Both are right based on what they say they do, but they produce different results.

Now, the analyst has a choice to make if both R and SAS are in their toolbox – how do I round this result? To answer this question, the analyst needs to understand the rationale behind round-to-even rule and the round-away-from-zero rule, and even other rounding rules that may exist. To our knowledge, this ‘how do I round’ question has *never* been asked with respect to clinical trial reporting until the difference between R and SAS default rounding was discovered. The ‘correct’ answer is up to the analyst to determine and justify. For example, with the appropriate number of significant digits, the difference between these results may be inconsequential to interpretation when presenting data on a table. However, rounding to even is intended to avoid biasing results away from zero, and if this is a risk within an analysis it should be considered.

Why should the analyst care? Why does it matter? One answer is because they want to tell the most accurate story of their data. However, and perhaps more importantly in the highly regulated pharmaceutical industry, because a third-party reviewer will be assessing the integrity of the data. If the reviewer attempts to reproduce the same results and chooses a different language, the analyst needs to be able to explain why results may differ, otherwise the integrity of the entire data package may be questioned. By fully understanding the implications of choosing a statistical modeling implementation in Language A vs Language B, the analyst can communicate the rationale of the choice, based on sound statistical reasoning, and instill confidence in the regulatory body reviewing the submitted data.

It should be noted that in what follows, it is assumed that statistical packages and routines perform in a manner consistent with their documentation. The question at hand is not whether the procedures are accurate or reliable, but rather in what ways do similar implementations across languages differ. Hence, we are not concerned with another major area of discussion within the industry – the validation of packages and software.

# Other Readings??

Perhaps cite the TransCelerate MoA project

Perhaps cite other working groups or published conference proceedings

**Analysis Framework**

With the number of statistical packages and macros currently in use, the analyst is faced with a considerable number of questions to answer: which package should I use? What degree of precision is required? Are the analyses reproducible? Etc. This may cause great concern, so we propose a framework to help guide the analyst’s thinking in determining the most appropriate statistical analyses for their application.  This framework is not to be taken as a determinative solution. Rather, it should help start the conversation between the statisticians and other relevant parties.

Step 1: Define research question

The research question may be specified in a variety of ways such as in the statistical analysis plan. If the analyst does not keep this in mind, the remaining steps will turn into a cycle that remains unguided in the overall purpose of the study in question.

Step 2: Define the statistical design to examine the research questions

The statistical designs are often informed by the protocols and procedures set in an experiment. Considerations such as randomization based on inclusion/exclusion criteria, stratified analyses, etc. help determine the model-making process.

Step 3: Looking at technical aspects

The analyst should perform a basic literature search to determine how to implement the statistical models as specified. There are a variety of packages that exist that may perform the same analyses, but utilize other kinds of optimization routines in the backend. Here is a starting list of questions to consider:

* Can the analysis be carried out in R, or SAS, or some other platform that is available to the organization?
* If so, what are the differences in documentation?
* What are the differences in modeling assumptions?
* Are there differences in default values used during the estimation procedures?
* Are there differences in available parameters?
* Are there differences in available output statistics?

These questions should be answered in context of the statistical design and be consistent with the organization’.

Step 4: Carry out the model and document the analyses

After considering all the technical considerations, the analyst should carry out the analysis as planned and document accordingly to reproduce the results. Details such as the R/SAS version, packages and macros used, etc., should be recorded.

Diagram, timeline

Description automatically generated   
Framework Application

We will now consider an application of the proposed framework in the context of survival analysis using the Worcester Heart Attack Study1.

Step 1: Define the Research Question

The objective is to evaluate efficacy of receiving artery fibrillation treatment in comparison without receiving treatment in term of the survival time for patient with heart attack.

Step 2: Define the statistical design to examine the research question

For the survival time, the Kaplan-Meier (KM) method (Kaplan and Meier, 1958) will be used to generate KM curves, the medians, quantiles and percentages of patients' event-free every 3-month interval for each arm. In the possible presence of substantial number of small cells, the hazard ratio (HR) for treatment effect and its 95% confidence interval (CI) will be estimated using un-stratified Cox proportional hazard (PH) semi-parametric model (Cox 1972) and a unstratified log-rank test will be performed.

Step 3: Looking at technical aspects

As a statistician, there is a common understanding that the Cox proportional hazards model is widely available in various platforms such as R/SAS/Python, etc. One may ask the question, does both SAS and R implement the Cox Proportional Hazards and from there, check the documentation for how certain statistical methods are handled such as the confidence interval computation, optimization routines, etc. We consider an initial set of questions the statistician may consider.

* **Are there differences in documentation?**

After looking at SAS 9.4, we observe that we can model the Cox PH model using SAS PHREG while in R, we can use *coxph* in the survival *survminer* package.

* **Are there different methods for handling ties in the data?**

Both SAS/R, according to their documentation, handles Efron’s method, Breslow’s method, and the exact method. By default in SAS, the method is Breslow. In R, the default is Efron. There is a reference that indicates that both utilize the same paper (ref).

* **Are there differences in the optimization routines used to compute the maximum likelihood estimates for the treatment (with or without receiving artery fibrillation treatment)?**

In SAS, the routine used is a modified Newton Raphson method (Heinze and Schemper, 2001) while in R, uses the standard Newton Raphson method with half-stepping.

* **Etc.**

Step 4: Carry out the model and document the analyses

Once an initial inventory of the technical considerations has been developed, the statistician should carry out the analyses as intended. Below, we indicate for this example that the statistician carried out the analyses in both R and SAS for exploratory purposes. Typically, the statistician may implement this in one statistical programming language.

Table

Description automatically generated 

We note that there are small differences which may lead us to consider asking several further questions to justify the statistician’s choice of platform. For example, how is the upper bound of the confidence intervals estimated? It is expected that the statistician will cycle between Step 3 and Step 4 to account for any discrepancies which may or may not be addressed by the statistical analysis plan such as the optimization routine or default methods which are not specified in the statistical analysis plan. After further consideration, according to ref, we learn that the values presented in the procedure is based on the following:

* **The default method for handling ties between the two platforms are different.** The survival package in R uses Efron’s method of handling ties while SAS uses Breslow’s method. In fact, both options are available in R and SAS, so by simply changing the default method, we would expect an identical HR and CI. From the arguments of coxph: There are three possible choices for handling tied event times. The Breslow approximation is the easiest to program and hence became the first option coded for almost all computer routines. It then ended up as the default option when other options were added to “maintain backwards compatibility”. The Efron option is more accurate if there are many ties, and it is the default option here. In practice, the number of ties is usually small, in which case all the methods are statistically indistinguishable. All 3 methods are asymptotically equivalent which is an ideal property to construct the relevant estimators.
* **The default method for confidence intervals of the KM estimates is different.** R uses “log”, and SAS uses “log-log”. R and SAS both offer the same types of confidence intervals but must be specified. “log-log” prevents the problem of having confidence intervals of >1 or <0, which might happen if using “log” transformation. However, both R and SAS will clip the interval at [0, 1] and report a bound >1 as 1 and <0 as 0.
* **The estimation procedure for the upper confidence interval is different between the two platforms.** The kth quantile for a survival curve is the location on the x-axis at which a horizontal line at height p = 1-k intersects the plot of survival probability. If the survival curve does not fall to 1-k, then that quantile is undefined.

Another question that may be posed after our initial considerations in Step 3 is how both methods do estimate the confidence intervals by some transformation. After looking through the guide, we can modify the expectations by changing the options and achieve the following results:



Summary

We constructed a framework which may prove useful for the statistician in considering the technical aspects of implementing the desired analyses.  The statistician may cycle between Steps 3 and 4 in order to provide a more exhaustive approach. We illustrate an example using both SAS and R to provide a frame of reference, though typically the statistician may implement the analysis in one language which should require careful attention to the available documentation.

# Call to action

This whitepaper outlines a framework for addressing discrepancies between statistical languages, there is no doubt that this is a cumbersome and complex task – particularly for an organization to approach alone. Rather, this is a perfect opportunity for organizations, working groups, and individuals to collaborate. As more findings are uncovered, that information should be accessible so as to improve the quality of data analysis produced as an industry. To facilitate this work, PHUSE has initiated a [GitHub repository](https://urldefense.com/v3/__https:/github.com/phuse-org/CSRMLW_bookdown__;!!AoaiBx6H!z588Y6-SOcl319FdxmQ-LazfP9OJwMRffuONlxfMMUe6GvP2dotGa5QmPk261u3oZaynZHzEyPrLZKtwvg7AvgBqoB0gnJhcxsCj_Q$) aimed at producing an eBook to organize and contain this work. Contribution is welcome from all, and contact can made through [workinggroups@phuse.global](mailto:workinggroups@phuse.global) or the GitHub repository directly.

APPENDIX

// INSERT CODE CHUNKS HERE

**Use Cases**

This whitepaper provides a framework for conducting statistical analyses in different programming languages, especially when multiple languages yield different results. The aim is to empower statisticians to make informed decisions when assessing differences in results due to differences in modelling assumptions, default parameters, choices of estimators or algorithms to compute a particular quantity, or even software related errors. Some of these differences may require adjudication by experts, while others may not be as critical. The primary objective is to provide guidance on the types of questions analysts should ask to identify fundamental sources of discrepant results. By doing so, the analyst is empowered to make informed decisions on how to perform the analyses of interest most appropriately.

Applications of the framework under development within the project initially focus on R vs SAS differences in four classes of statistical models: linear model, mixed model, survival analyses, and Cochran–Mantel–Haenszel analyses (CMH). The following list is a simple breakdown of what this paper will assist with.

|  |  |  |
| --- | --- | --- |
| Topic | Concept | Page Reference |
| Summary Statistics | Five Number Summaries | … |
|  | Frequency Tables |  |
|  | Grouped Summaries |  |
|  | Miscellaneous Summary Values |  |
| Linear Models | Regression |  |
|  | Comparing Means |  |
|  | ANOVA Models |  |
|  | ANCOVA Models |  |
|  | MANOVA Models |  |
|  | Using Contrasts |  |
|  | Tests of Normality |  |
|  | Tests of Equal Variance |  |
| Mixed Models | Mixed Model ANOVA |  |
|  | Repeated Measures |  |
|  | … |  |
| Survival Models | Kaplan Meier |  |
|  | Log-rank test |  |
|  | Cox Proportional Hazards Model |  |
|  | Parametric Model |  |
| CMH | Cochran-Mantel-Haenszel |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |