MODERNIZATION OF STATISTICAL ANALYTICS (MSA) FRAMEWORK

TransCelerate BIOPHARMA INC.
ACCELERATING THE DEVELOPMENT OF NEW MEDICINES

ABSTRACT

A lack of evolution within the pharmaceutical industry's analytic capabilities has given rise to inefficiency and a failure to leverage modern technologies within the clinical development space. Moreover, the limited regulatory perspective on this matter has become a barrier in implementing and leveraging newer analytical software capabilities. By developing a foundational framework that increases Health Authorities' confidence in use of modernized statistical analysis and efficient technology across the industry, an opportunity exists to provide clarity and instill confidence in the use of modern software technologies, including those that generate outputs that support regulatory submissions. This Framework is based upon a methodology that establishes the principles of accuracy, traceability, and reproducibility for a modern analytical software environment to demonstrate that results it generates is reliable. A set of best practices and guiding principles is also provided to support the Framework's implementation across industry sponsors and other stakeholders. Along with a developed Framework, engagement and consultation with Regulatory Agencies can be conducted to gain clarity on the Framework's robustness and ability to provide confidence in using modern software technologies within clinical development.

TABLE OF CONTENTS

Part I: MSA Framework

Introduction/Problem Statement

Biopharmaceutical companies operate in a highly regulated environment. As such, it is important that the software systems used to analyze data, as part of the clinical development evidence generation process, can be relied upon to produce accurate and consistent results. Per ICH 9 guidance:

> *The computer software used for data management and statistical analysis should be reliable, and documentation of appropriate software testing procedures should be available.*
>
> - Guidance for Industry E9 Statistical Principles for Clinical Trials

Regulatory agencies do not require that any specific software be used. However, given the limited guidance on what constitutes reliable software, the actual breadth of software used in the Biopharmaceutical industry is limited. This Framework provides a methodology that stakeholders across the industry can use to demonstrate to Health Authorities the reliability of software – whether traditional or non-traditional – and thereby build confidence in analytical programs that may even be at the cutting-edge.

With the emergence of electronic and digital data sources (e.g., wearable tech data, electronic health records) and in the sheer quantity of data collected, there is a strong desire and need to implement analytic capabilities (automation, advanced statistical methods, machine learning algorithms, and other Artificial Intelligence techniques) that requires multiple solution providers (whether traditional, non-traditional, or both) to take advantage of that data for generating new insights, decision making, and evidence generation for clinical trials. Moreover, the programmers, statisticians, and data scientists leveraging these technologies are just as well versed. As Biopharmaceutical companies are faced with the challenge of modernizing their analytics infrastructure, capabilities, and talent pool, they are looking to develop a breadth of software capabilities where they can customize best fit for purpose tools to deliver the innovative solutions needed to support regulatory submissions aimed at bringing new medical breakthroughs to patients more quickly.

While Biopharmaceutical companies and their Clinical Research Organizations (CRO) partners have well established Standard Operating Procedures for assessing the suitability of traditional software for inclusion as a component of a modern statistical analytics environment, many do not have practical procedures in place to enable a similar assessment for non-traditional software. The objective of this Framework is to provide practical guidance on how to approach regulatory agencies in an effort to demonstrate to them that ***any*** analytical solution produces credible results and are suitable for use in the regulatory process.

Scope

The scope of this paper focuses on setting out a Modern Statistical Analytics (MSA) Framework, including best practices on implementation. The MSA Framework comprises a set of principles that can be adhered to build and maintain a modern computing environment that Health Authorities will find reliable and output of which can be used with confidence to support regulatory filings (i.e., marketing applications for clinical trials novel treatments or therapies). The Framework includes discussing requirements for programming infrastructure and processes, including risk-assessment and mitigation practices to execute to demonstrate reliability of an MSA environment.

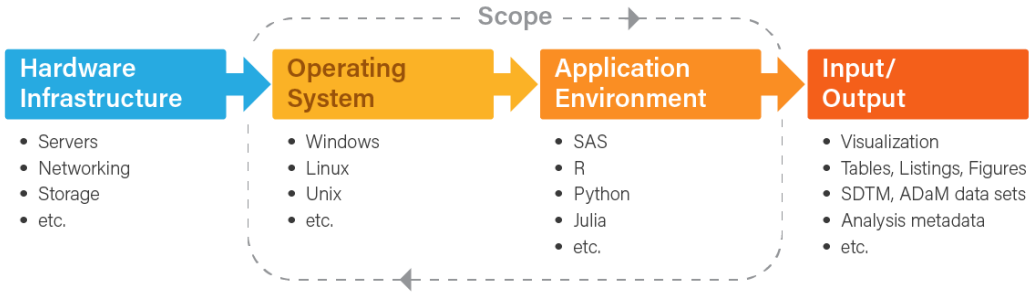

Figure 1: MSA Technology Stack: This reference to specific brands of software products is done merely for illustrative purposes and, by merely referring to these products as examples in both **Figure 1** and other passages in this Paper, TransCelerate is neither endorsing nor promoting the use of any of these products.

While this paper provides principles and guidance on establishing an MSA environment, specific details and end-to-end implementation steps are not provided as they should be the province of individual organizations working with their respective IT groups to build an MSA environment that best fits their organization's needs. This paper also does not provide guidance on any specific outputs of an MSA environment, how to produce any outputs, or product selection in developing an MSA environment (except examples to illustrate or clarify framework concepts) as these decisions should be made unilaterally by each company or organization implementing a MSA environment. Specifically:

- Out of Scope
  - Use of this Framework to produce specific deliverables, such as tables or any other component of a CSR regulatory submission is out of scope
  - Change Management and how to transition to an MSA from a legacy environment
  - Hardware (servers, memory, storage, networking) and hosting (cloud, on premise, hybrid) infrastructure
  - Cyber Security-related infrastructure
  - Selection of software products or solution
  - Specific method, packages, and outputs of applied complex modeling and simulation analysis
  - Infrastructure and processes to manage Big Data from a programming workflow perspective

Framework Approach & Design

This Framework proposes approaches for developing an MSA environment, regardless of whether it supports relatively traditional or less traditional software, that will give health authority agencies, and thus industry, confidence in the reliability of the results generated from such an environment, even when the agencies heavily scrutinize the results. This is accomplished by demonstrating that for each output, the MSA environment produces accurate and reproducible results with documentation that traces those results to their lineage of dependencies. This end-to-end control based upon accuracy, reproducibility, and traceability over the software environment will provide the health authorities confidence that the result is reliable even under the most critical reviews for data analysis – outputs that support efficacy or safety of a new drug candidate in regulatory submissions.

Note: While the foundational principles (accuracy, reproducibility, and traceability) of the Framework are specific and required for the MSA environment, the specific implementation of them is entirely the purview of the respective organizations building it. The accountability and responsibility for developing the specific processes, products, and infrastructure to build an MSA environment and using it in support of a regulatory submission that is scrutinized by health authorities can only be taken by the implementing clinical development organization. As such, the contents below are general proposals, contextual information, and best practices to help any given organization with its efforts to implement a reliable MSA environment and ultimately demonstrate to the regulators that submissions based on outputs generated from such an environment are reliable.

Accuracy

When testing does and does not matter (as much)

Regarding implementing this Framework, accuracy can be the most challenging, especially for less traditional, open source software. Within this Framework's context, accuracy is the measure of correctness of software libraries that are used to generate results from an MSA environment – it is not the programs that developers write using those software libraries. Those programs, such as a primary analysis program that demonstrates overall survival for a therapeutic agent, will typically incorporate Quality Control (QC) processes and testing that are in place irrespective of any software that is chosen. Software that has become established in the industry often has the benefit of a single organization that is developing, selling, and maintaining a software tool. Accuracy can be assured, in among other ways, by a commercial entity documenting its Software Development Lifecycle (SDLC) processes, testing methodologies and results, or a surrogate of such evidence through contractual warranties indicating that those processes were followed. IT organizations typically rely upon that documentation as part of their implementation of traditional MSA environments.

For less conventional analytical software tools, such as open source software like R or Python, organizations often must navigate through a varied community of developers and in some cases, commercial solution providers, and determine to what degree reliable software development practices were followed. The primary benefit of open source software is in the vast community of authors that typically come together to provide a host of functionality with incredible speed. It is also the most challenging aspect of enabling its use within a regulated setting given the potentially vast differences in quality of their development.

One clear way to demonstrate accuracy would be to fully test the functionality of the software against requirements. However, that approach proves to be infeasible given:

- The extensive amount of code and functionality to be tested and the amount of resources and time it would require executing
- Managing dependencies and re-testing code whenever a dependency is updated (see Figure 2)
- The demand from users who may require new packages for using new analytical methodologies that may have time-sensitive impacts on deliverables

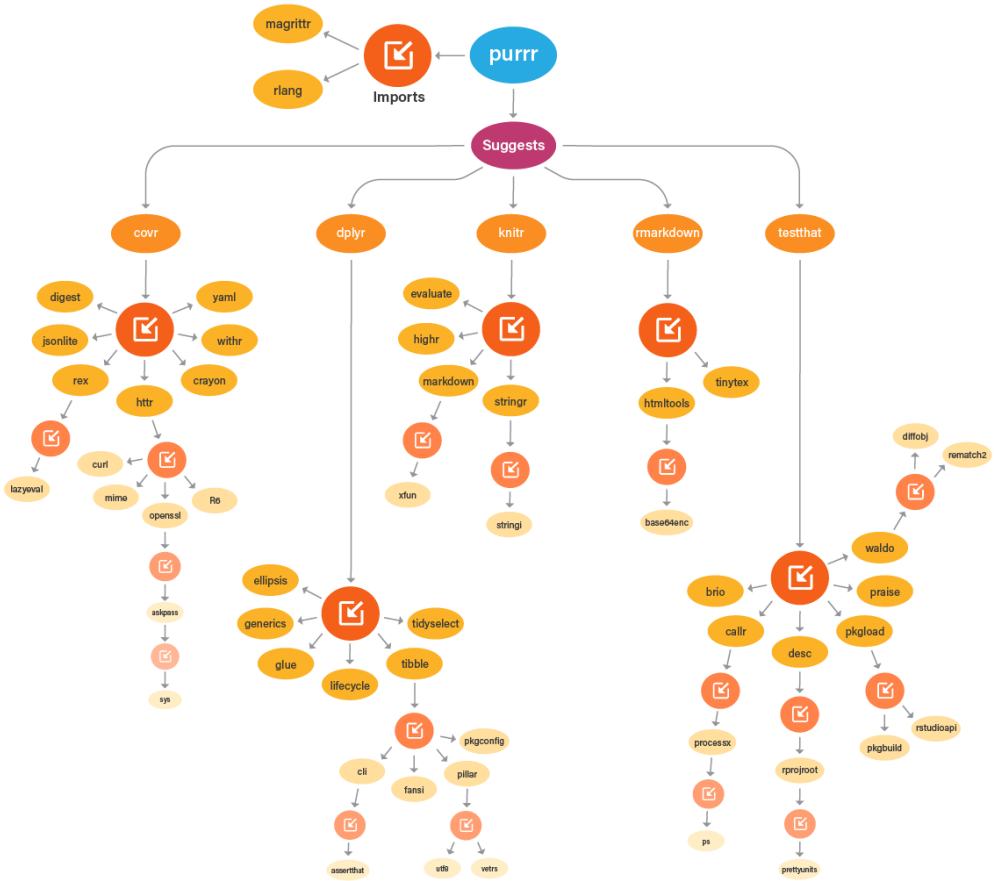

Figure 2: Dependency tree for purr

The FDA mentions this challenge when giving guidance for software validation of medical devices:

> *Generally, it is not feasible to test a software product with all possible inputs, nor is it possible to test all possible data processing paths that can occur during program execution.* (**General Principles of Software Validation** section 5.2.5)
>
> *The level of validation effort should be commensurate with the risk posed by the automated operation…The extent of validation evidence needed for such software depends on the device manufacturer's documented intended use of that software.* (**General Principles of Software Validation** section 6.1)

Given the FDA's acknowledgement regarding testing of software and the acceptable use of assessing risk with intended use, this Framework proposes establishing a library's accuracy by combining the inherent qualities of reliability for a software library with the intended use of a software library and the impact of that intended use on a regulatory submission. This combination of factors determines a risk level for the software library's use and thereby determines whether its accuracy should be accepted by health authorities with sufficient confidence for that intended use.

How Accuracy Works Within This Framework

Within an MSA environment, accuracy is not a binary designation, i.e., yes or no, but an assessment (see Figure 3) based upon 2 independent factors: the software's ability to demonstrate confidence in its results and the intended use of its output.

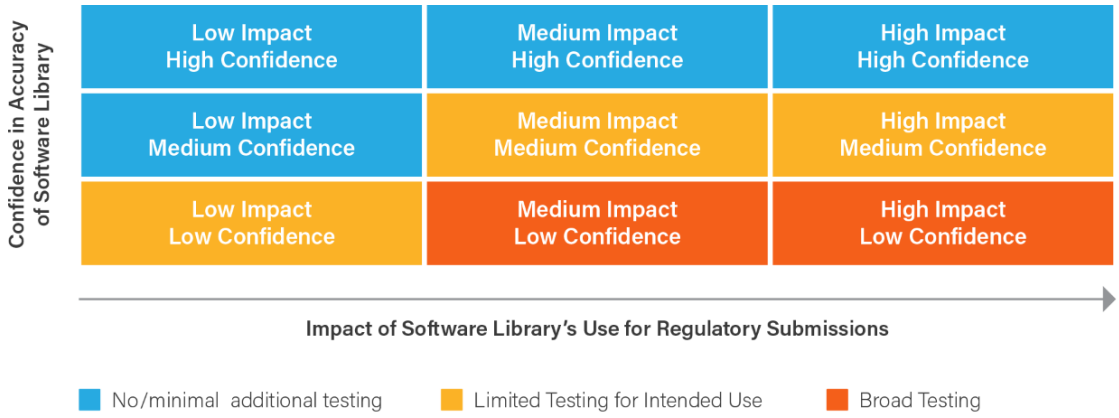

Figure 3: Assessment of Accuracy

In the blue tier, it could be appropriate to use the software library for regulatory submission use without any or with minimal additional testing beyond what is provided by the software authors or is generally available. In the orange tier, again no additional testing or perhaps minimal testing would be needed if the software library is used for internal decision making but would require additional testing if used as part of a regulatory submission. Red tier is only appropriate without any additional testing for exploratory use. Any additional use for decision making or regulatory submission would require testing of varying levels depending upon its intended use or on thorough testing of any specific analysis.

Confidence in the Software Library

When scrutinizing any given software library for quality, we can classify a combination of attributes to denote a High, Medium, or Low level of confidence that the library can reliably deliver the functionality it proposes to do (see Figure 4).

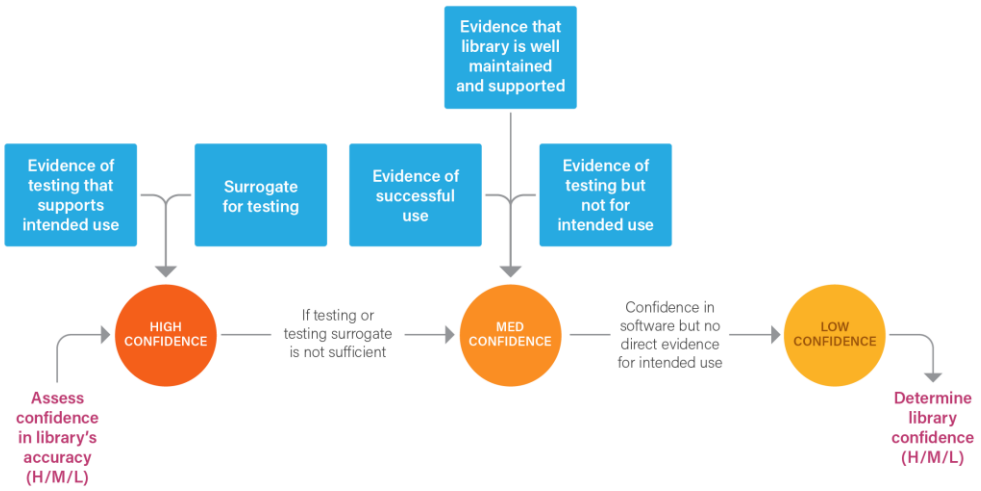

Figure 4: Assessing software library confidence

High Confidence means that there is little ambiguity that a software library produces functionally accurate results from evidence that relevant tests were successfully executed. This can be established through:

- Documented test cases and results for its library functionality
- Executable tests cases that can be used to verify published test results
- Testing of dependencies

OR a surrogate for testing can be derived from:

- An auditable SDLC or equivalent exists by the library authors with artifacts such as requirements and design specifications
- A warranty (from a commercial entity) for its intended use
- High percentage of code coverage {such as > 75%) provided by the library authors. Code coverage measures the degree to which programming code is executed by tests

High Confidence designates a software library as being suitable for use with the Highest Impact to the business outcome without any additional testing needed to mitigate risks. If the provided evidence of testing or its surrogates are not sufficient to provide High Confidence, the software library may be designated as Medium or Low Confidence.

Medium Confidence is established with the following characteristics:

- Test cases are provided and executed without error but does not adequately test intended use
- Source code for library is published and maintained
- Issue management for library is maintained and published by authors
- Usage of library (i.e., number of downloads) is significantly higher relative to similar capabilities
- Usage of library in a successfully submitted primary or key secondary analysis
- Library is mature (i.e., has been in existence for 3+ years, given version has been in use for 1+ years and is not the first major version)

Medium Confidence designates a software that has inherent trust but does not have direct evidence of functional testing for its intended use. This type of library's confidence may be increased to High by adding additional tests. Use of medium confidence software may be justified in some specific instances by carefully documenting that results are appropriate for use.

Naturally, if neither testing exists to support High Confidence or other characteristics to support Medium Confidence, a Low Confidence designation would be made, indicating that the given library should only be used for exploratory or experimental purposes. Again, additional testing may be performed to increase its confidence level to Medium or High.

Intended Use

Intended use, its impact, and why confidence alone is insufficient

In parallel to assessing the standalone qualities of a software library, it is critical to understand the purpose that library will be used. We must ask, "What does it do?" and "How will it be used?" to determine what impact (High/Medium/Low) the output will have to the broader business outcomes. The specific categorizations of software may vary per organization implementing an MSA environment, but in general may be categorized in the following manner:

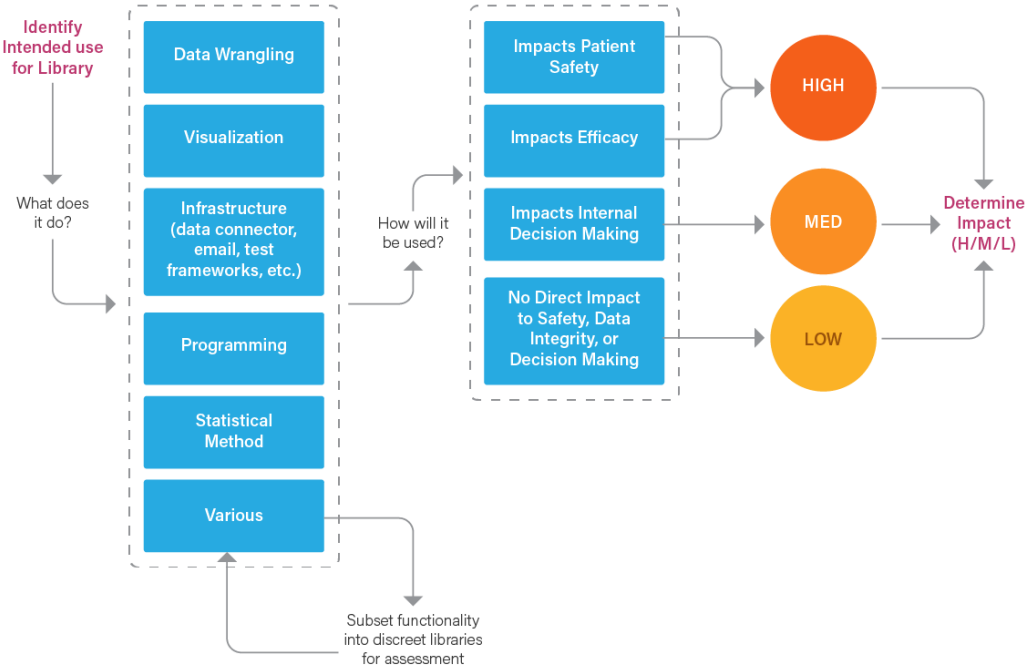

Figure 5: Assessing impact for software library's intended use

High:

- Directly impacts determination of patient safety
- Directly impacts evidence supporting efficacy for a regulatory submission

Medium:

- Directly impacts data integrity supporting internal decision making

Low:

- Supports exploratory analyses
- Does not impact efficacy, safety, and internal decision making

In general, libraries may be designated for a given use in conjunction with its confidence. For example, infrastructure libraries (such as base and stats distributed with the R software and developed by the R foundation) may have a high impact when used for a clinical study. That said, these libraries tend to be on the high confidence side because of their maturity (I.e., no large changes) and prevalence (used by millions of developers). While statistical methods will typically be high impact, it is likely many of the supporting libraries are not as broadly used and, consequently, will have a lower confidence without further vetting of the package's accuracy. These categorizations are purposefully broad given that it can be interpreted in

different ways per each implementing organization. Therefore, it is important that a process (i.e., a Standard Operating Procedure or SOP) be defined specific to an organization perspective on impact and risk for determining a High, Medium, or Low designation.

One additional consideration must be made for libraries that provides more than one intended use. For example, the R package Hmisc encapsulates a variety of different functions from statistical methods to string manipulation to graphics. In such cases, it would be prudent to wrap a subset of functionality into its own custom library so that a proper assessment of use and impact could be made. Once risk is established for the custom library, a programmer could leverage that custom library for high-impact, high-risk outputs should adjust its risk profile support it, which would not be supported by leveraging the Hmisc library itself. In summary, accuracy takes into consideration the vast libraries of software needed by users that may deliver functionality of varying quality and be used to support business objectives varying impact and makes a determination on a software libraries appropriate use.

Reproducibility

In the scope of this paper, if a statistical analysis output (see Figure 6) can be recreated from the original dataset along with the associated environment, including all artifacts and dependencies, it is considered to be reproducible. If an output is not reproducible, it is questionable whether results of the output can be trusted. Hence it is essential that the environment should be able to reproduce an output.

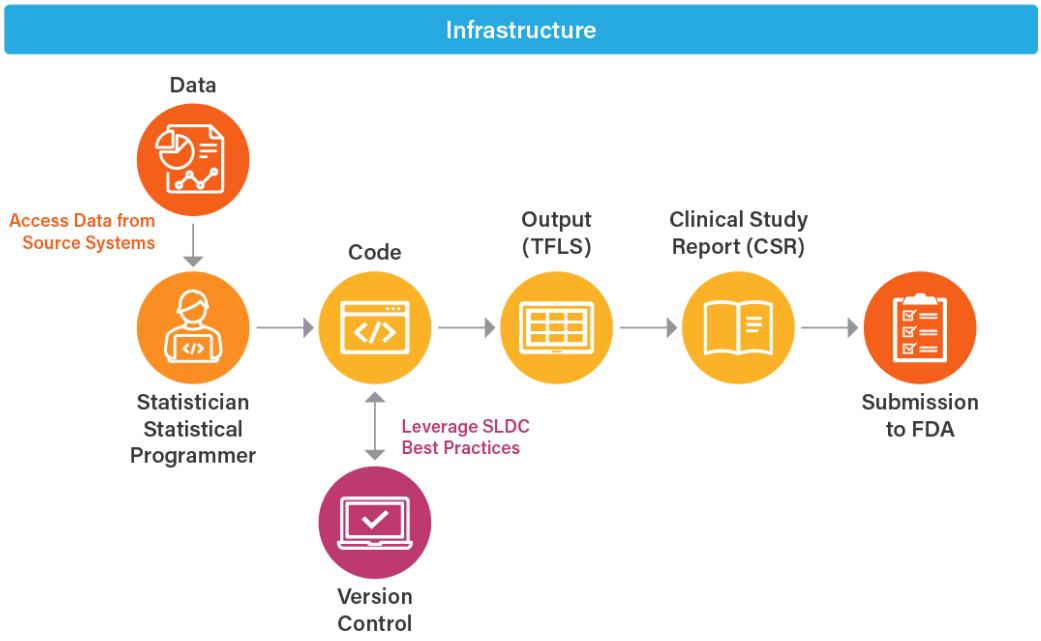

Figure 6: Clinical Programming Process Flow

Some of the attributes presenting the challenges to reproducing an analysis in a Modern Statistical Analytics Environment include:

- **Verification:** Identifying and verifying that the results can be repeatedly generated is necessary to confirm that the accuracy is not compromised during reproducibility
- **Infrastructure:** The key components of a computing infrastructure comprise of operating system, compiler versions and their associated system dependencies. It is essential to keep track and maintain lineage of these components to ensure reproducibility of the compute environment
- **Data:** Data is the important input into a statistical analysis. Ensuring data integrity and data lineage is essential in maintaining source control of data to have confidence in the analysis

- **Library Dependencies**: Within the compute environment, for every analysis a set of packages or libraries are used for the analysis. Many such libraries have underlying dependencies. The libraries and their associated dependencies would have to be re-loaded to recreate the environment to ensure repeatability
- **Source code:** The source code is set of instructions written in a programming language that can be transformed by an assembler or compiler into binary machine code that can be executed by the computer. Source code management enables versions/revisions to the program. Each version is given a timestamp and includes the person responsible for the change supporting auditability
- **Human errors:** Traditional environments that have manual processes to ensure reproducibility inherently pose risk due to human errors. Adopting automation tools where possible will reduce errors and increase the confidence in reproducibility

Traceability

Traceability refers to the ability to trace inputs to outputs, with the main goal being to provide evidence to connect e-data, code, and environment to the final output that is produced. For our use cases, we have a few elements that are especially critical:

Datasets used to produce output: Datasets must be in the format for ingestion by the scripts to generate the final outputs. Datasets may be updated overtime and may be re-formatted for use in an analysis. The datasets must be versioned and indexed in a way for retrieval by scripts and any downstream analyses executed to generate outputs to be replicated.

Programs and/or dependent programs that are called to produce an output: Orchestrating the top-most layer of execution are the actual programs called to generate an output from the source data. Minimally, this must be included to replicate an output. Furthermore, these scripts must be with lower-level assumptions about the system.

System libraries: System libraries include installed binaries from the OS and compiled programs that are extensions of the programming engine itself (e.g., R packages). These are critical for replicating an environment because there are version mismatches that may occur which may cause an output to not be replicated. Additionally, compilation issues may occur which make working with pre-compiled binaries more desirable than source programs.

Programming engine: This refers to the programming interface, whether it is SAS, R, Python, etc. The version of this is key to replicating not only some of the libraries but also some of the outputs. As such, tracking this, versioning it, and sharing it as part of the submission package is critical for reproducing an output.

Compute environment: The compute environment is something not strictly tracked before and represents an opportunity for innovation. Specifically, there have been advances made in versioning and sharing compute kernels that making environments more portable.

Reproducibility and traceability are key elements to enable trust in the statistical environment. Current environments have challenges in interoperability, involve complex inefficient processes including manual effort when running apart from host environment. Inefficient process such as those involving manual effort can expose significant risk. We will propose best practices for a modern statistical environment (which includes but is not limited to open source software) to increase efficiency and confidence over current state.

ART (Accuracy, Reproducibility, & Traceability)

How Accuracy, Reproducibility, and Traceability Work Together to Provide a Reliable MSA Environment

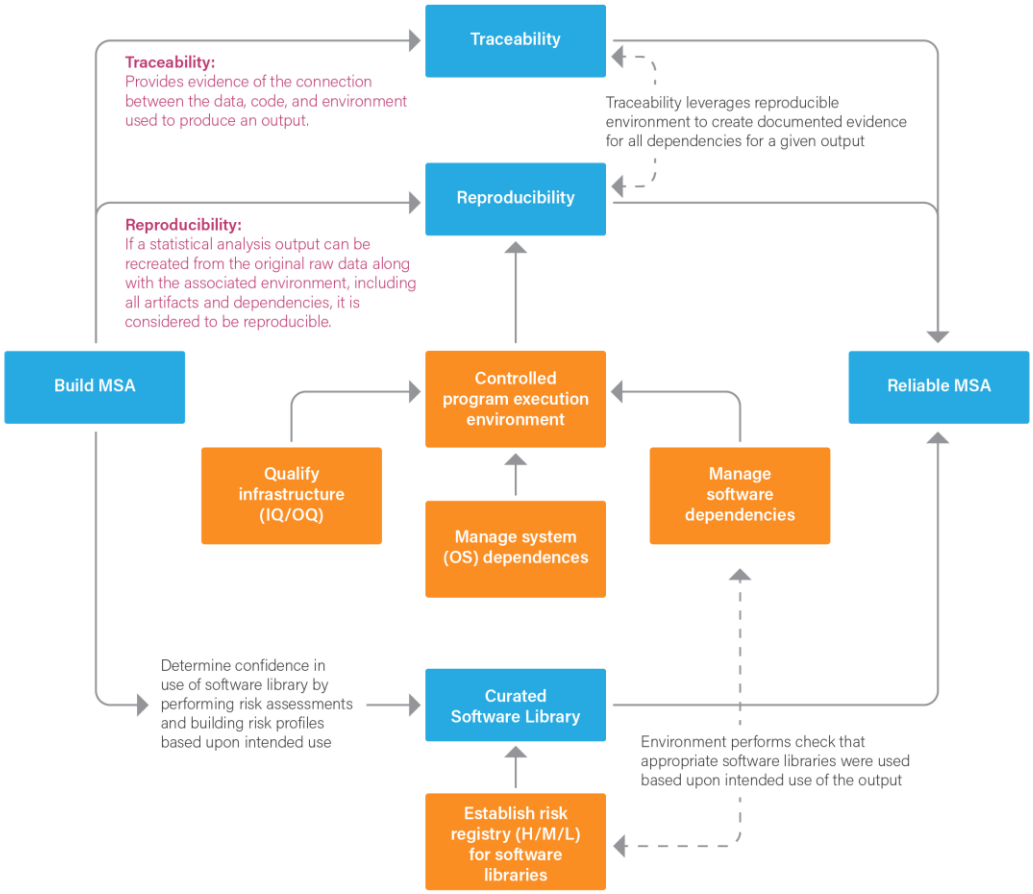

Figure 7: MSA Framework Integration of Principles

Reproducibility, traceability, and accuracy are core tenets of the Framework that delivers a reliable MSA environment. They cannot do this as standalone concepts. They must be integrated together cohesively through infrastructure tools and standardized processes (see Figure 7 above).

This integration begins by identifying a program output's intended purpose (submission, decision making, or exploratory/other). Processes must be established where programs executed within the MSA environment are appropriately tagged with intended use by the programmer. Tools should be developed to scan all programs or program logs for the software libraries used and determine, based upon the program's intended use, whether the risk level for the libraries are aligned with the allowable risk level for the intended program use. Quality control processes should be established to scan all outputs from an MSA environment so that programs use appropriate software libraries for its intended use.

In addition to logs showing software library components for a given program, those logs should also contain complete traceability from output back to:

- Source datasets with date timestamp and file size
- Software libraries with version
- Any other external inputs to the program such as other non-library programs that were referenced
- Any infrastructure components impacting ability to reproduce the output (docker container information if applicable, Operating system version, compiler information if applicable)

Given that environments will change over time with software libraries and/or operating system version upgrades, dependent components should be located or localized such that the specific inputs to the program can be re-used. Current infrastructure capabilities can enable this reproducibility such as docker containers and code repositories.

Governance Through Process and Tools

Why all of this does not work without it

Within Clinical Development, standard operating procedures (SOPs) are the de facto rules for governing how new therapeutics are developed and assessed for efficacy and safety. In using an MSA, SOPs must be established to ensure that the environment is demonstrating accuracy, reproducibility, and traceability. For accuracy, governance should specify a Change Control Board (CCB) composed of Subject Matter Experts (SMEs) from stakeholders including Statisticians, Statistical Programmers, Quality Assurance, and Information Technology to assess the risks of new or changes to software libraries. Managing changes and consistency is key to enable consistent integration of studies to establish totality of evidence.

Procedures would define:

- Building a centrally managed, curated library
- Adding new software to curated library
- Up-versioning an existing library
- Updating the software ecosystem (such as versions of SAS, R, Python, etc.)
- Updating the global environment (OS, compiler, code repository)

For reproducibility and traceability, SOPs should be developed that require IT to maintain a validated environment by building tools and controls to demonstrate that:

- Programs cannot leverage non-assessed packages, such as a user downloading and using a library outside of the organization's designated library repository
- Programs cannot bypass rules for risk, i.e., using a library that is designated as "Low Confidence" to calculate an endpoint for a regulatory submission
- Analyses must still be reproducible, traceable, and accurate through changes to software ecosystem and global environment (i.e., an analysis must be reproducible with consistent results even though a library, dependency for that library, or the operating system is up versioned)

Users should be required to:

- Follow SOPs for developing programs that specify intended use and check for errors or warnings that indicate proper use of software libraries was not followed

Tools and processes must be developed and required to be used through SOPs to ensure that an MSA demonstrates reliability throughout its use for generating outputs supporting regulatory submissions.

PART II: IMPLEMENTATION GUIDANCE FOR MSA

The concepts specified within the Modernization of Statistical Analysis Framework can be basic in its tenets but may prove challenging in practice to implement. Current environments in Biopharmaceutical companies often rely upon legacy infrastructure and processes to provide a basis for how analytical programming environments can and cannot evolve. However, when attempting to implement a new type of environment without the guidance and constraints of legacy, a set of best practices and explorations of use cases can be invaluable.

Best Practices for Framework Implementation

Considerations for implementing a modern statistical analysis environment that supports credible regulatory submissions over multiple programs and many studies across many years include:

- Maintaining control through lifecycle of a study/program. Tools and processes can be leveraged including the use of code version control systems and automated log checking to ensure proper code is maintained and used, and any errors in programming is identified and remediated for production use.
- Establishing a culture that collects and documents any efforts that were performed to make a software/package/library valid as well as any subsequent use cases, and to make them a public (internal) wiki
- Developing senior leadership support to focus on a mindset for innovation and fostering the culture to embrace change
- Building infrastructure capabilities to support the application of the Framework for newer use cases, such as Big Data utilization
- Establishing a community-based software risk library where users, member companies, and other organizations can contribute metrics, testing to support a library's particular risk profile
- Creating the handshake between IT and Business when it comes to implementing and changing the statistical analysis environment. IT traditionally builds and maintains computing environments based upon SOPs and other defined processes that ensures reliability. Business functions will also build tools and utilities to automate processes or make tasks easier. This often has risk and change management aspects to consider since business and IT may not be required to follow the same processes for the statistical analysis environment. Coordination across the organizations within R&D and IT should include:
    - A validated software environment (R/Python/SAS/etc.) with core functionality managed and supported by IT
    - Access to production instances should be carefully managed to ensure traceability and controlled usage, and to enable automation. For example, users may be granted indirect access and permission to execute programming indirectly through APIs to maintain adherence to established processes in partnership with IT
    - A use-case validation policy (beyond IQ/OQ) at hand that also considers scenarios where reproducibility cannot be exact, e.g., analyses based on simulations
    - Add-on functionality evaluated according to this Framework and implemented/maintained in cooperation with the IT peer group
    - Documentation of added functionality and intended use-cases is provided using a convenient and controlled document management system
    - Production applications (scripts, logs, local data, output) are archived automatically.

Roles within an MSA environment

Effective collaboration with stakeholders is critical in moving forward on creating these modern statistical environments. Namely, when new technologies get introduced to environments, it is typically not in a vacuum. Rather, it is built onto existing environments.

There are various roles involved in moving these environments forwards, both within the statistics organization and within IT. Below is a brief description and how the roles work together to advance the statistical computing environment.

1. Statisticians: Statisticians will work together with programmers to vet and define which new programs will be added to the environment.
2. Programmers: Programmers will work together with statisticians to vet new software but also carry out specific validation efforts as it relates to specific use of new software.
3. Clinical Quality: Clinical quality will provide regulatory oversight in guaranteeing the tools remain compliant for use
4. IT: IT will take Commercial-off-the-shelf (COTS) solutions and install them into the server environments. Additionally, they will work closely with engineering to develop streamlined or automated processes for these deployments
5. Engineering: Engineering will work with IT to custom configure the COTS solutions. Additionally, they will work with statisticians and programmers to guide the use of new tools for building things like automation pipelines
6. Maintenance: Once the new tools have been deployed, a maintenance competency will be needed within IT to manage maintenance and upgrades to the environment

Use Cases

The most important aspect of this Framework is that it can be used to address unmet needs for accelerating the use of analytical tools for regulatory submissions. Three high-level scenarios have been defined below indicating how the Framework could be applied in introducing new analytical tools in a compliant manner: environment creation, reproducibility and traceability, and submission. These use cases have been identified as being especially relevant because they follow a logical progression of maturity for a programming organization to follow for adopting this Framework. The environment must be created first to meet Good Clinical Practices (GCP) standards as it relates to both SDLC and change control. Additionally, the workflow for generating outputs from standardized data must be outlined to guarantee it supports reproducibility and traceability. Lastly, there is an example of how this Framework fits into the submission process and how it contributes to that experience.

Environment Creation

Basic user story: As a statistician/programmer, I would like to have an analytical environment where I can run statistical analyses with new tools in a risk compliant manner.

Currently, I have a server which my organization can log into for running statistical analyses. It has secure connections to our data, an existing suite of software, and a dedicated process for working in this environment. As part of an update, I would like to add software capabilities for use in a regulatory submission.

Specifically, I would like to add a few R packages for use in a shared library dedicated for regulatory works. Up front, I have decided which packages I would like to include and begin the process of pushing these packages through the Framework – capturing details about their intended use and any specific tests that may map to my intended use.

Testing scenario	Action
Complete testing suite for intended use	Include package and dependencies in library for regulatory use
Complete testing suite for intended use but not dependencies	Evaluate testing of dependencies and determine whether additional testing is needed
Incomplete testing suite for intended use	Determine whether additional testing is needed
No testing for intended use	Test and determine whether software is adequate for regulatory use

After taking each package that I want to use through the Framework, I have a library of packages to use for regulatory submission. Additionally, I will have some packages that did not meet the regulatory use threshold – these will be included in libraries for different risk levels depending on the outcome. For packages that had incomplete testing for intended use, they may fall into a medium risk profile where I may choose to implement additional testing that could raise the lower risks and allow regulatory use. However, for packages that were determined inadequate for regulatory use, I could still use them for exploratory work based upon confidence I derive through other factors such as popularity within the community, maturity of the package, accessibility, and availability of the authors, etc.) even though their continued use did not warrant additional testing. Operationally, I have used the Framework to expand the analytical capabilities of my computing environment. Additionally, I have road-tested the Framework internally with my company, setting the stage for faster iterations for future updates.

Decision Tree/Process Flow:

- **Step 1:** Clinical trial statistician finds a piece of software they would like to use to analyze data and defines how they would like to use it.
- **Step 2:** Impact analysis is done on the software to see what other changes to software must be made, including OS, system dependencies, software versions and packages.
- **Step 3:** Review the impact and outline any additional test-cases that may need to be re-run.
- **Step 4:** Assign a risk metric category for the software and develop a validation plan, including test cases based on intended use.
- **Step 5**: Committee review of change to analysis environment, including verification of test plan.
- **Step 6:** IT updates infrastructure and goes through IQ/OQ.
- **Step 7:** SOP is created for appropriate use of the software versions and packages. Supplemental documents are provided for assisting with inspectors

Reproducibility & Traceability

Basic user story: As a statistician/programmer, I would like to be able to reproduce an analysis I did for a regulatory submission.

As previously defined, there are several components required to replicate an analysis and environment. Below are specific details on ways to support reproducibility and traceability with these given components, including the coordination between IT and biostatistics.

Component	IT & Engineering	Biostatistics
Datasets	Provide governance to the data sets and programs, as well as managing backups.	Generate ADaM datasets for analysis. Develop and version programs for generating outputs. Have defined process for replicating outputs from given scripts.
Programs to Generate Outputs		
System Libraries	Manage and version updates to the system libraries, programming engine, and overall compute environment.	Have defined processes for replicating run-time environments by setting given system libraries and programming engines in a session.
Programming Engine		
Compute Environment		

Containerized Proof of Concept (PoC) Submission

Basic user story: As a statistician/programmer, I would like to use new analytical tools to create a submission package.

Specifically, I am looking to leverage both R packages and Docker containers to encapsulate the required programs, data, instructions, logs, a development portal, an interactive web application server and documentation for a submission package.

Docker was implemented and scripts were provided to instantiate the Analytic environment.

Steps	Detail
Docker Installation	Via instructions and scripts
Testing the Docker Installation	Test instructions included
Running the Docker Installation	Issues reported around writing to local drives – seeking to resolve with further PoCs
Building the Container Images	

The initial PoC faced some technical challenges including recipient security around writing to local file systems. It is advisable to bring in IT expertise to assist with any issues.

Opportunities for Innovation

Regarding ways to identify opportunities to introduce some of these new changes, there is a careful balance of risk and impact or even a phased approach. Namely, experimentation will be key with new technologies being broadly promoted after careful internal vetting with minimally risky projects. To do so, we propose focusing on individual competencies as listed below:

- Leveraging rigorous ways to replicate computing environment consistently which are portable and more efficient, such as containerized technology, can help support reproducibility
- Capturing ideas for leveraging newer modern capabilities (ex: notebooks for submission, footnotes, version of tables, opportunities systematic, best software dev practices etc.)
- Developing test-driven programming strategy:
    - A pharmaceutical company-built R packages for internal use for trial submission. As part of these they have test suites that work in parallel to validating the R package. This fits with reproducibility because many of the test-cases depend on reproducing bugs; additionally, they prevent new tools or environments being introduced which may prevent reproducibility
    - Alternatively, they have external R packages that they are evaluating/validating for use in a regulatory submission. For these, they evaluate the tests and write up additional test scenarios to fit specific use cases
    - Lastly, with updated deployments they have CI/CD pipelines that depend on test suites. These prevent bugs from being introduced into an environment that their IT team may be updating

CONCLUSION

This Framework is part of a bigger culture shift happening within analytics across the Biopharmaceutical industry, and within analysis and reporting competencies for these companies. Specifically, there are innovations happening at an accelerated pace with software used for statistical analysis. While risk-averse companies may choose to avoid using these software capabilities, there is an opportunity cost which ultimately impacts the ability to bring drugs to market faster. Additionally, these innovations can only be leveraged across our industry if the Health Authorities ultimately find that these capabilities produce reliable results. This Framework and these Best Practices give industry an approach for demonstrating to the Health Authorities the reliability of analytical software, including those that have recent emerged on the industry's horizon. By working together to build the Health Authorities' confidence in these software, we will facilitate the modernization of our statistical tools. We only stand to gain from accelerating the industry's transformation. Lastly, in general, organizations must adopt a more malleable approach to their operations – including the use of frameworks like this to help with the industry's transition to more modern computing solutions. Each organization will have to decide on its own their appetite for change and balance the associated risks. From the field of statistics at large to the individual study teams – embracing this culture shift will be critical to the success of modernizing analytics.

GLOSSARY

Term	Definition
CI/CD	Continuous integration/continuous deployment. An agile approach to software delivery for frequently making small updates to software which go through automated integration testing and automated deployments.
Cloud	The on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user.
Container	A software execution environment that makes shared used of the operating system kernel but otherwise runs in isolation from other applications.
CRO	A contract research organization (CRO) is a company that provides support to the pharmaceutical, biotechnology, and medical device industries in the form of research services outsourced on a contract basis.
Docker	An open source software platform for building applications based upon containers.
Library	In software development, a library is also a collection of computer programming code that exhibits behaviors and has a well-defined interface by which the behavior is invoked. Behavior is often implemented such that it can be re-used by multiple independent programs that are not related to each other.
On prem	Abbreviated from on-premises, it implies the use of local resources to place software which is installed and runs on computers on the premises of organization.