



SPA MDR CDR SCE

Reproducible Validation Framework for R & Python in GxP

Kala Shivali & Don Muldoon



Agenda

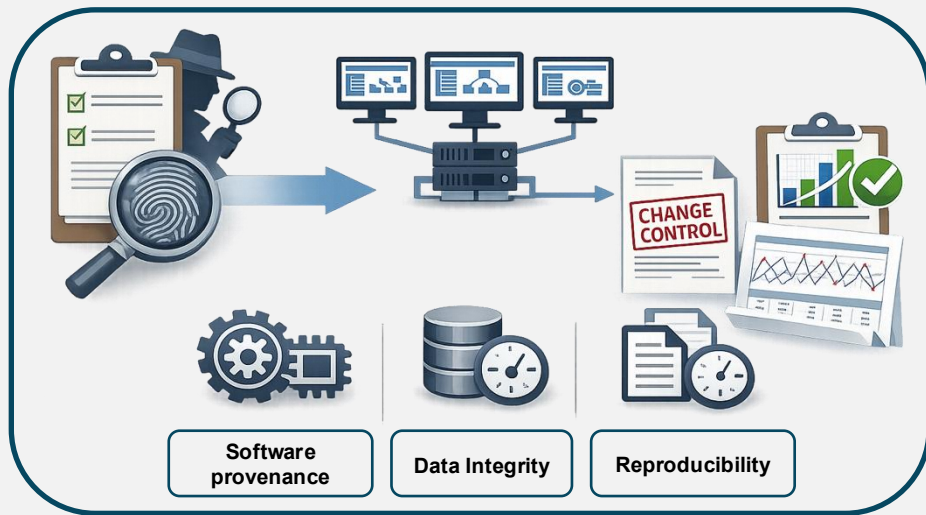
- Regulatory Expectations and Context
- Challenge of Open-Source Analytics for GxP Deliverables
- How to Address Challenges: Framework & Design Principles
- Architecture & Software Build of Materials
- Validation Strategy, Evidence & Audit Readiness
- Future Outlook and Conclusion



Regulatory Expectations and Context

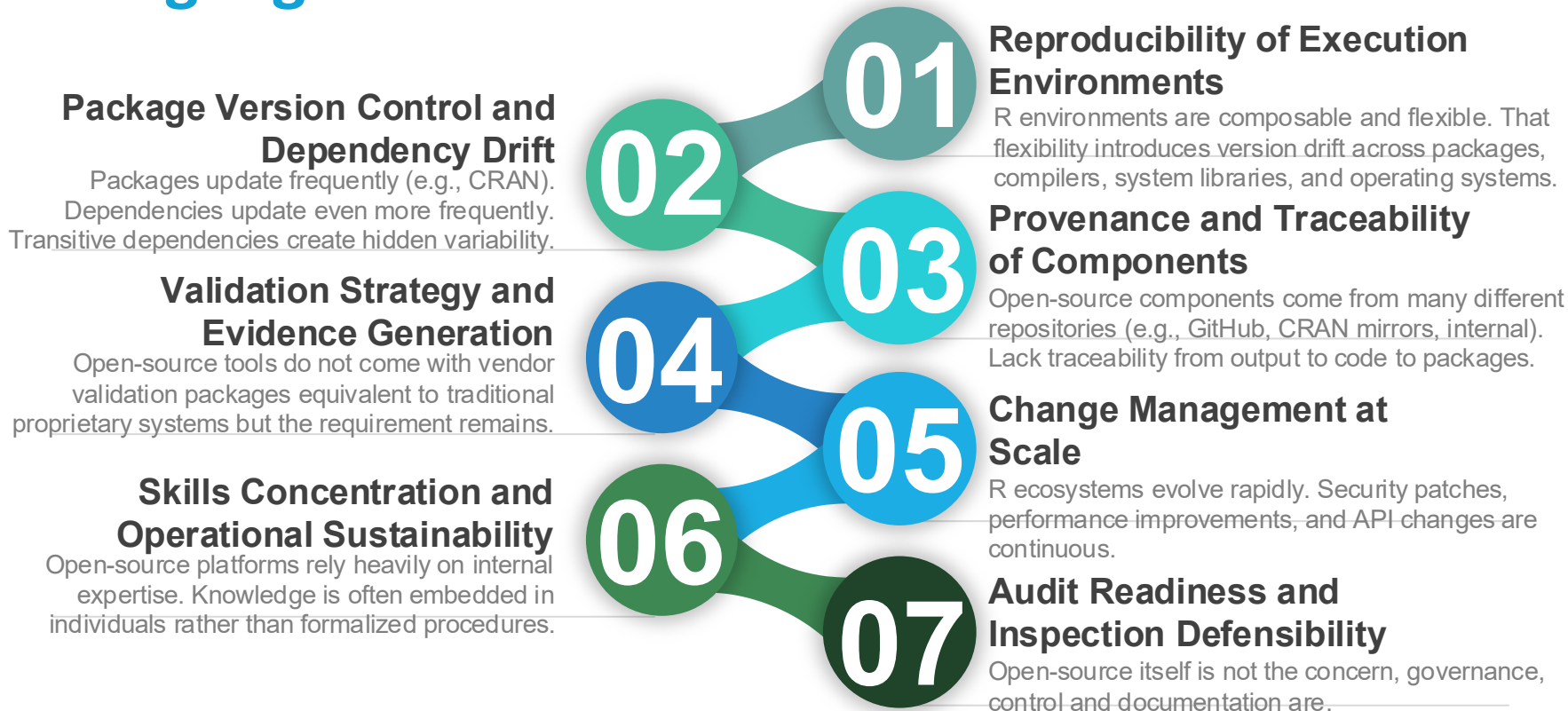
Key elements of regulatory guidance include:

- Defined intended use
- Reproducibility of results
- Traceability and traceability of changes
- Documented governance of changes

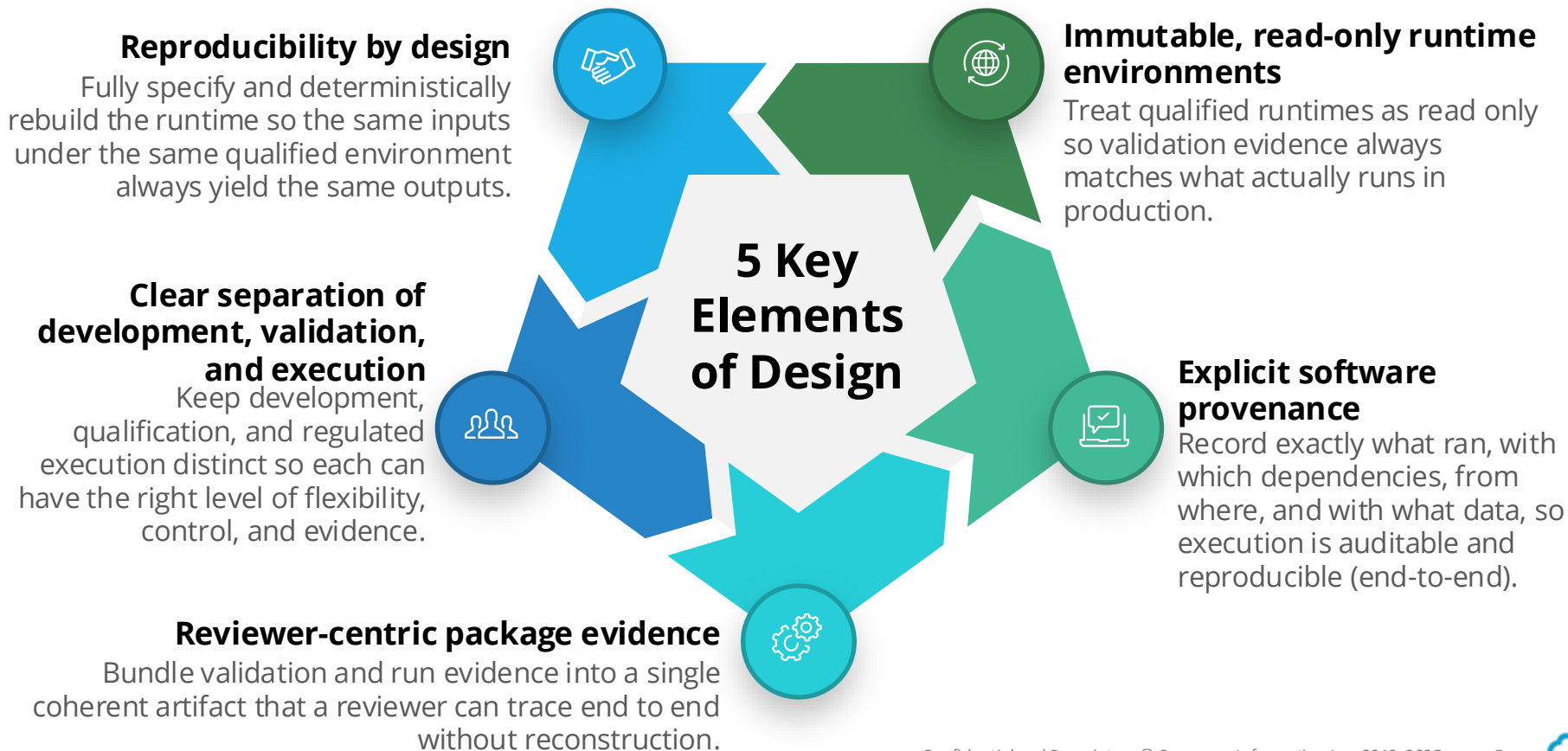


Inspectors routinely focus on the consistency of execution environments, the provenance of software components, and the ability to reproduce outputs

Top Key Challenges of Using Open-Source Languages for GxP Deliverables



Foundational Design Principles



Framework & Design Principles

Design Principle	SAS Baseline Practice	Open-source Equivalent Control	Evidence Artifact a Reviewer Expects
Reproducibility by Design	Reproducibility supported by vendor-controlled releases, standardized binaries, and centrally managed server images	Fully specified runtime with deterministic rebuild rules, locked dependencies, and a stable runtime identifier	Runtime specification, dependency record, build manifest, runtime identifier referenced in run records
Immutability	Validated SAS servers treated as controlled assets, upgrades gated, unauthorized changes prevented	Qualified runtimes promoted as immutable versioned images, production execution restricted to approved versions	Approved runtime inventory, change control records, promotion approvals, immutable runtime identifiers
Explicit Provenance	Vendor provenance for core SAS runtime and version identity	Provenance assembled from locked dependency records plus runtime metadata so origin and change can be demonstrated	Provenance manifest, dependency record, runtime metadata, run record binding code, data snapshot, and runtime

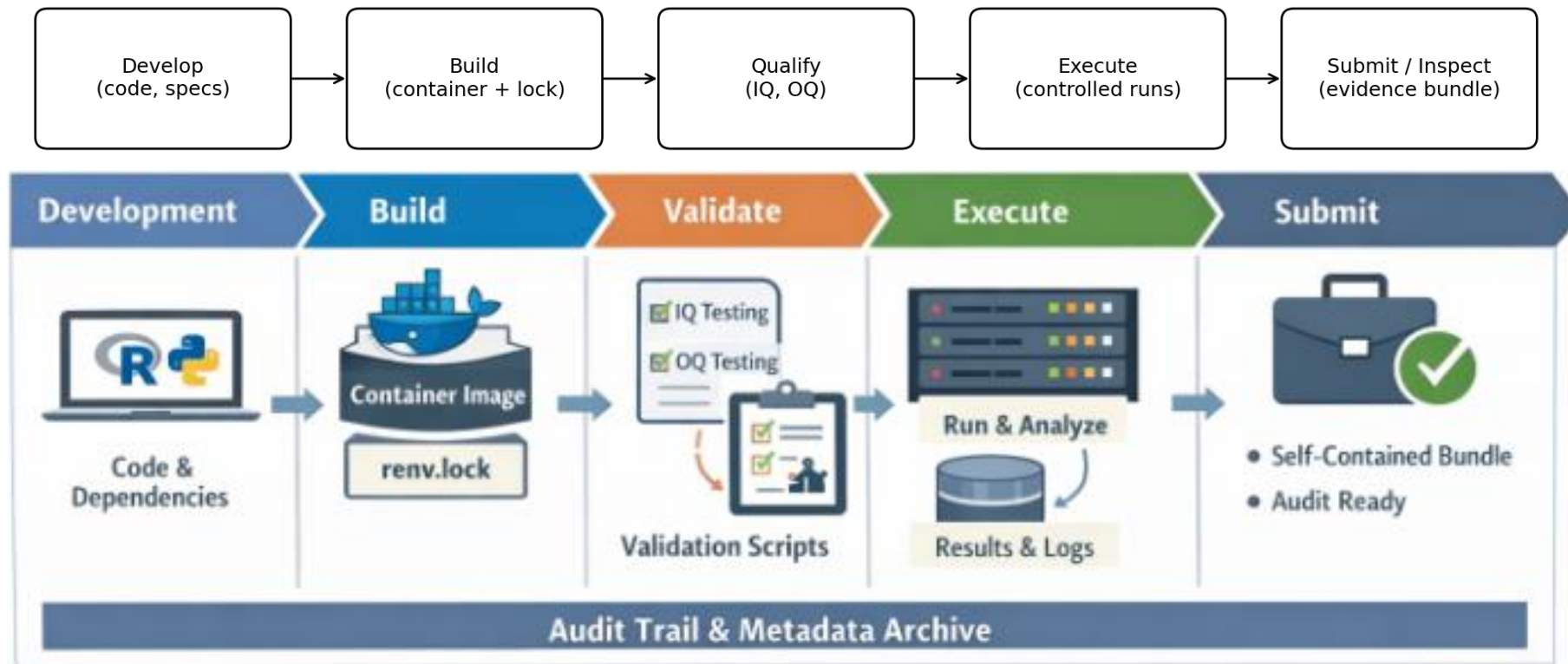


Framework & Design Principles, Continued

Design Principle	SAS Baseline Practice	Open-source Equivalent Control	Evidence Artifact a Reviewer Expects
Separation of Concerns	Separation between development workstations and validated grid or server execution	Separation between dev environments and regulated execution, with qualification performed in controlled pipelines	Environment boundary definition, pipeline records, qualification reports, access control evidence
Reviewer Centric Packaging	Validation packages include installation evidence, configuration baselines, and test reports	Evidence bundle that binds runtime identifier, qualification evidence, and execution records into one inspectable package	Evidence bundle index, traceability excerpt, IQ and OQ reports, execution logs, reviewer guide



Architecture



Software Bill of Materials (SBOM):

Enable deterministic rebuilds and support audit inquiries into component provenance

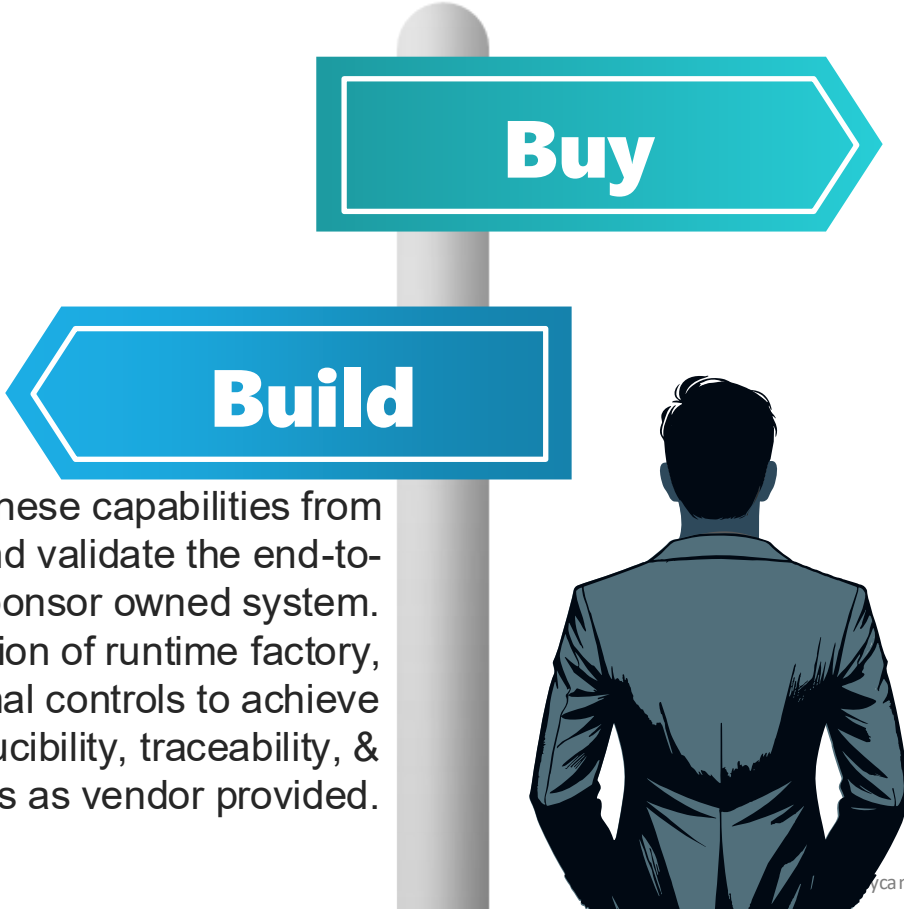
Layer	Artifact	Purpose
OS	Container digest	Immutable runtime definition
Language	R/Python versions	Interpreter control
Packages	renv.lock	Dependency pinning
Metadata	Hash identifiers	Audit traceability

- Container images encapsulate the operating system, system libraries, and language runtime
- Lock files such as `renv.lock` record package names, versions, sources, and cryptographic hash identifiers includes the R and Python dependencies
- Together they provide transparent insight into each software component used during execution



Implementation Strategy

Both buy & build paths must meet GxP expectations, but sharply differ in effort and risks



Buy

Adopt a vendor platform that provides controlled runtime creation, governed promotion, and audit ready evidence as integrated capabilities within the existing CDR-SCE and not a separate environment.

Build

Build &/or assemble these capabilities from individual components and validate the end-to-end process as a sponsor owned system. Requires deeper validation of runtime factory, its pipelines, & operational controls to achieve the same reproducibility, traceability, & inspection readiness as vendor provided.



Validation Strategy

Vendor Provided Solution

- Sponsor qualifies the platform and its configuration, then relies on the platform's controlled runtime creation, promotion, and audit features as the mechanism that supports multiple runtime environments defensibly

Build/Assemble Solution

- Sponsor must validate the end-to-end process, including the build pipeline, dependency control, promotion gates, evidence capture, and the operational procedures required to keep runtimes stable over time



Example IQ Test Set (illustrative)

IQ TEST OBJECTIVE	METHOD	ACCEPTANCE CRITERIA	EVIDENCE
VERIFY BASE IMAGE AND OS IDENTIFIER RECORDED	Capture image digest and OS release metadata	Digest and OS metadata stored, matches build record	Build manifest, digest record
VERIFY R AND PYTHON RUNTIME VERSIONS	Query runtime versions inside container	Versions match specification	IQ log output
VERIFY LOCK FILE INGESTION	Compare lock file inventory to installed packages	No unexpected packages, all required present	Comparison report
VERIFY CONTAINER IMMUTABILITY CONTROLS	Attempt modification in controlled test	Modifications prevented or detected	Control test log
VERIFY EVIDENCE STORE WRITE AND RETENTION	Write IQ artifacts to evidence store	Artifacts stored with timestamps and access control	Evidence store record



Minimum Requirements for Intended Use

1. Only qualified runtime images can be used for regulated execution,
2. Runtime identifier is bound to each execution record
3. Dependency inventories are captured and retained
4. Access is role-based with audit trail
5. Validated change control governs upgrades
6. Reproducibility is demonstrated through repeat execution tests



Inspection-Ready by Design: Evidence & Audit Readiness

Auditors and
QA reviewers
typically
focus on five
key questions

- 01** Intended use and system boundary
Provide intended use statement & the defined system boundary for the runtime factory, promotion controls, evidence capture, and regulated execution.
- 02** Validation for intended use
Provide requirements & test traceability matrix, followed by the approved IQ and OQ protocols and reports demonstrating coverage for each requirement.
- 03** Change control and upgrades
Provide change control records, including impact assessment, approvals, & qualified runtime inventory showing what is authorized for regulated execution.
- 04** Evidence based record integrity
Provide role-based access control evidence & audit trails, plus evidence store controls demonstrating retention, integrity, and handling of approved records.
- 05** Reproducibility
Provide execution records that binds outputs to the inputs, the code, and the qualified runtime identifier, then demonstrate replay using same qualified runtime version.



Minimum Contents of an Evidence Bundle:

A single evidence bundle supports these inspection questions and remains consistent across runtime versions and studies

- Validation plan and risk assessment
- Approved intended use statement and system boundary
- Requirements and test traceability matrix
- IQ/OQ and report package including results, deviations, and approvals
- Qualified runtime inventory including runtime identifiers & immutable digests
- Dependency inventory artifacts linked to each qualified runtime identifier
- Change control records for platform changes and runtime content changes
- Execution records for representative runs including run manifest and log extracts binding inputs, code, runtime identifier, and outputs
- Evidence store controls supporting retention, integrity, access control, and read-only status for approved records



Future Outlook

- Regulated R and Python is moving from exception to operating model.
- Inspectors will look for controlled execution, complete evidence, and on demand replay.
- Provenance plus dependency inventories become the proof.
- Deterministic replay is required, not optional.
- Govern the runtime as a configuration item with stable identity and boundaries.
- Differentiation will be consistency at scale across runs and runtime versions.
- Standard evidence bundles cut inspection friction and late remediation.
- SBOM and continuous evidence automation strengthen the case for qualified runtimes, controlled change, and drift proof replay across any analytics stack.



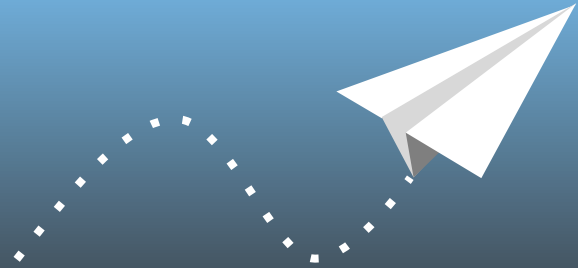
Conclusion

- GxP open source is defensible only when the execution environment is controlled as rigorously as the code.
- Immutable runtimes, explicit dependency inventories, and formal qualification are the foundation.
- Requirements traceability plus an inspection ready evidence bundle are required.
- Deterministic replay closes the inspection gap of proving exactly what environment produced historical results.
- A runtime factory or vendor equivalent enables versioned change, fast replay, and audit responses without rebuilding context.
- **Result:** scalable, inspection ready open-source analytics across studies and over time.
- **Bottom line:** this operating model makes R and Python viable for compliant clinical trial analysis as integrity and transparency expectations tighten.



Thank You

<https://www.sycamoreinformatics.com>



Kala Shivali

kala.shivali@SycamoreInformatics.com
<https://www.linkedin.com/in/williamqubeck/>

Eric Zloty

eric.zloty@SycamoreInformatics.com
<https://www.linkedin.com/in/suhasgiriraj/>