PharmaSUG 2026 - Paper TT-391

Taming Polyglot Analytics: Simplifying Cross-Language Workflows in a Unified IDE

David Ward and Troy Wolfe, Triam

ABSTRACT

Modern analytical workflows increasingly span multiple programming languages, each chosen for its strengths in data access, transformation, modeling, or visualization. While this polyglot approach is powerful, it introduces complexity through fragmented tooling, inconsistent workflows, and increased maintenance overhead, ultimately reducing productivity and increasing operational risk.

This paper demonstrates how cross-language analytical workflows can be simplified using the **Siemens Analytic Workbench**, an integrated development environment that enables analysts to work with multiple programming languages from a single, unified interface. The presentation demonstrates a project in which SAS, Python, R, and SQL are written and executed within a single development environment, highlighting both individual and organizational benefits.

This paper is intended for statistical programmers and analysts working in increasingly heterogeneous analytical environments. Participants will gain a clearer understanding of how unified tooling can reduce complexity in polyglot workflows while preserving flexibility and improving productivity.

INTRODUCTION: THE RISE AND RAGE OF POLYGLOT ANALYTICS

Over the past decade, analytical workflows have expanded beyond any single programming language, with teams selecting tools based on their respective strengths. SAS remains prevalent in regulated data preparation and reporting, Python provides access to a broad ecosystem of machine learning libraries, R supports advanced statistical modeling, and SQL remains central to relational data access.

As a result, many analytical projects are inherently polyglot. A single workflow may span SQL for data extraction, SAS for preparation and validation, Python or R for modeling, and additional tools for reporting or visualization.

This flexibility, however, introduces operational challenges. Analysts must often work across multiple development environments, each with distinct tooling, dependency management, execution models, and debugging interfaces. As workflows grow in complexity, this fragmentation creates friction in development and increases the difficulty of maintaining consistent, reliable processes.

Several common pain points emerge in these environments:

- **Context switching**
  Analysts frequently move between tools to write, test, and execute code, increasing cognitive overhead and interrupting workflow continuity.
- **Reproducibility and maintainability**
  Distributed environments make it difficult to maintain a clear, reproducible execution path and to understand how components interact over time.
- **Collaboration challenges**
  Teams must coordinate across tools and practices, increasing onboarding complexity and making shared workflows harder to reason about.

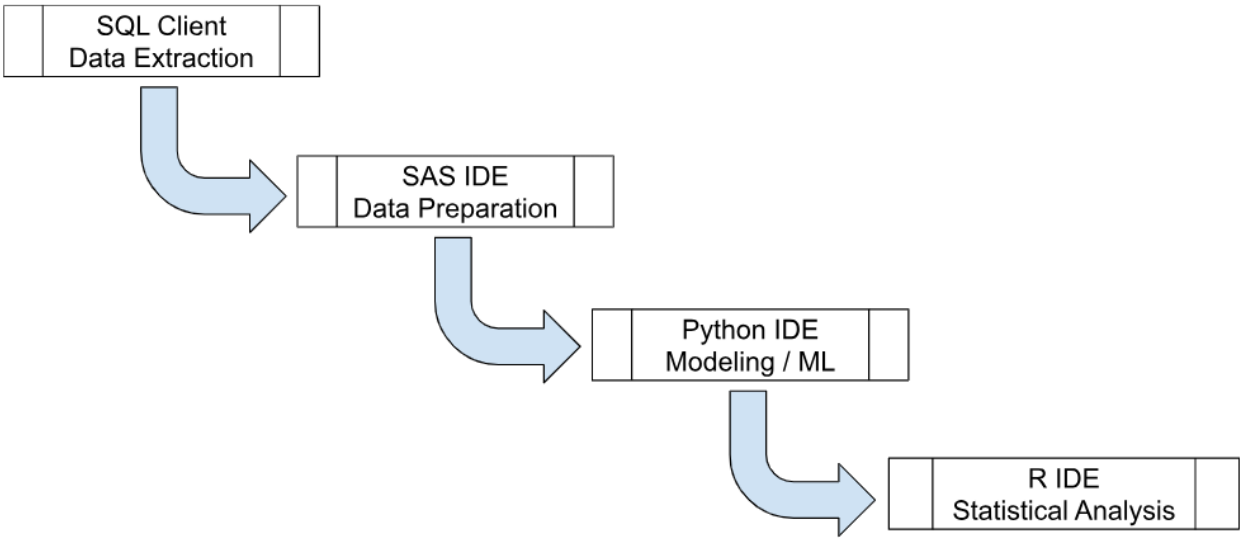
These challenges arise not from the use of multiple languages itself, but from managing them across disconnected environments. One approach to addressing this problem is the use of a unified development environment that supports multiple languages within a single workspace.

A UNIFIED IDE APPROACH

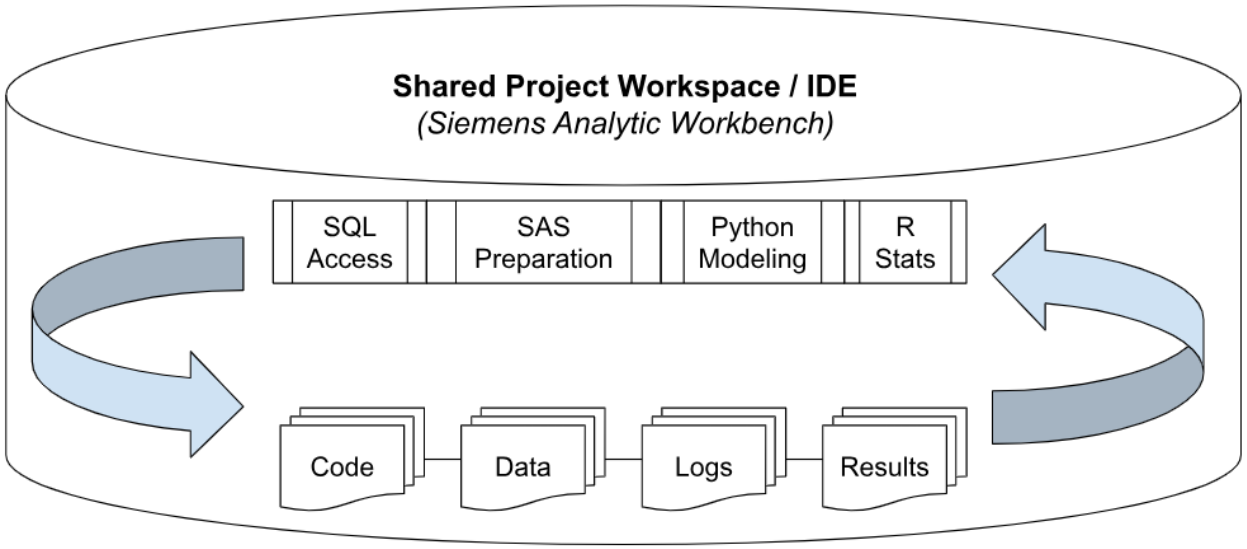
IDEs centralize editing, execution, and project management, but analytical workflows often remain fragmented across language-specific tools.

Figure 1. One environment to rule them all - a comparison of traditional multi-tool analytical workflows and a unified development environment for polyglot analytics.

Traditional Polyglot Workflow



Unified Analytical Workspace



A unified IDE approach attempts to reduce this fragmentation by supporting multiple programming languages within the same development environment. Rather than switching between separate tools, analysts can write, execute, and manage code written in different languages from a single interface.

One example of this approach is the **Siemens Analytic Workbench**, which allows analysts to maintain projects that include scripts written in SAS, Python, R, and SQL, providing a common interface for editing, execution, logging, and project management. Scripts are organized within a shared project structure and executed from a common interface alongside related artifacts.

This model offers several advantages. Analysts can maintain a clearer view of how different components of a workflow interact, and teams can organize multi-language projects in a way that is easier to navigate and maintain. In addition, common development features such as syntax highlighting, version control integration, and debugging tools can be applied consistently across languages.

The remainder of this paper will demonstrate how a multi-language analytical workflow can be implemented using this unified environment. The demonstration project combines SAS, Python, R, and SQL scripts within a single workspace, illustrating how cross-language workflows can be developed and executed without requiring analysts to switch between multiple development tools.

DEMONSTRATION: A MULTI-LANGUAGE ANALYTICAL WORKFLOW

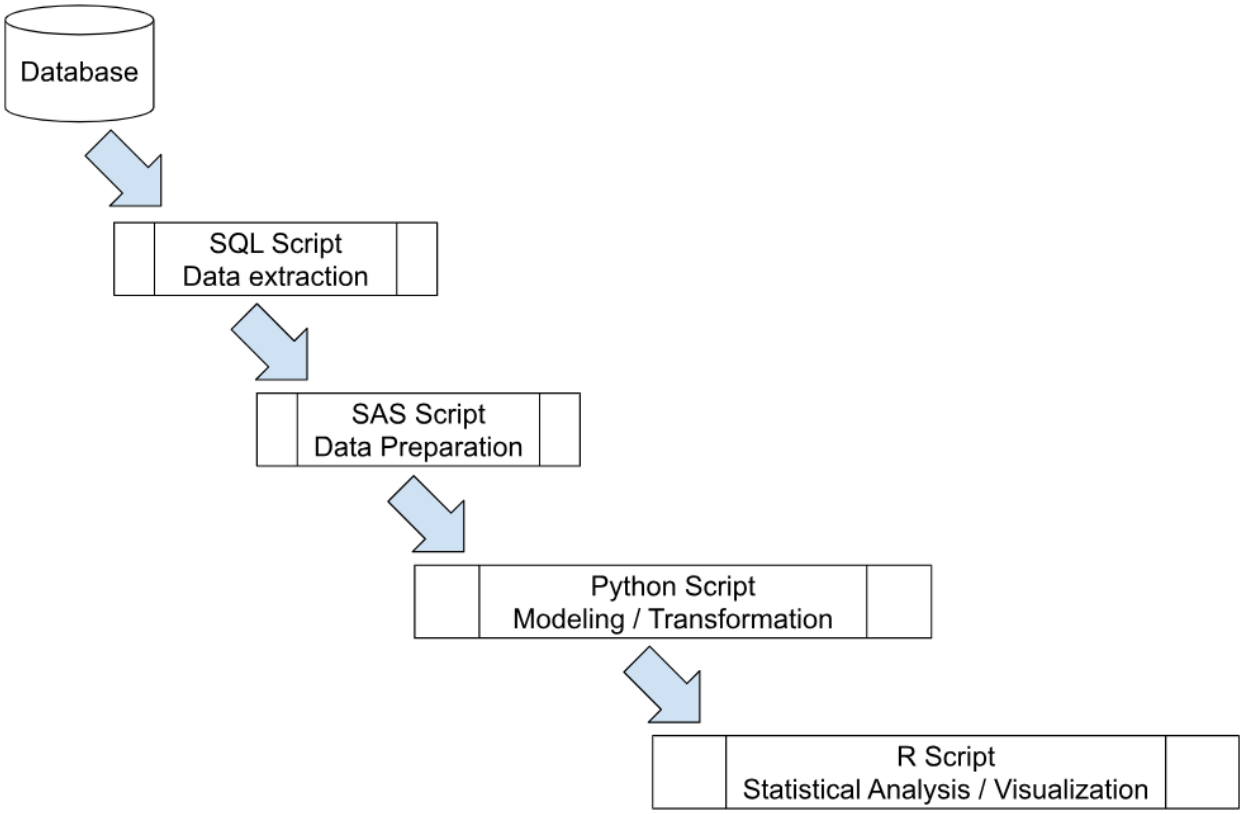
This section demonstrates a four-language workflow within a unified development environment consisting of four steps:

1. Extract data from a relational database using SQL.
2. Prepare and validate the data using SAS.
3. Perform modeling or transformation using Python.
4. Generate statistical summaries or visualizations using R.

The workflow is managed within a single project workspace.

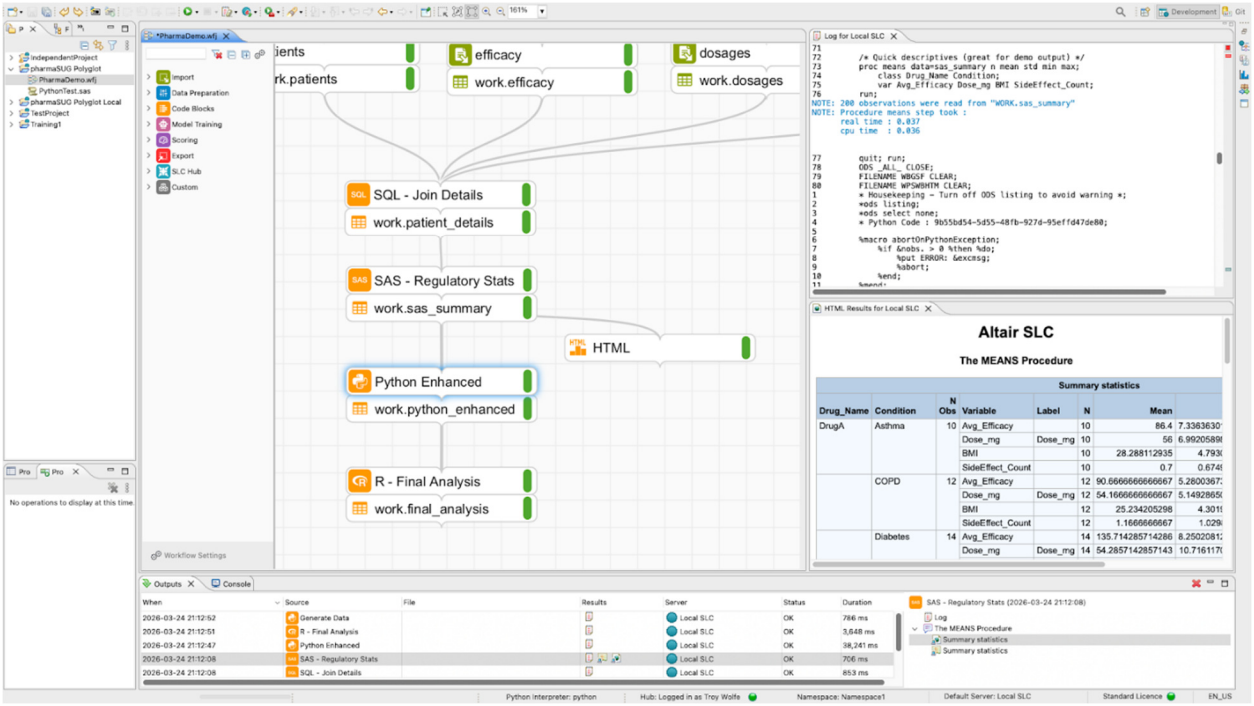
EXAMPLE WORKFLOW

Figure 2. Conceptual structure of the demonstration project



In traditional environments, these steps are executed in separate tools such as DBeaver, SAS Studio, Jupyter, or R Studio, complicating coordination and reproducibility. Coordinating these steps across multiple tools can make the workflow difficult to maintain and reproduce. In the Siemens Analytic Workbench, all scripts are stored and executed within the same project workspace.

Screenshot 1: A Project in Analytic Workbench (showing the workflow as main window, tabs in background, log/output)



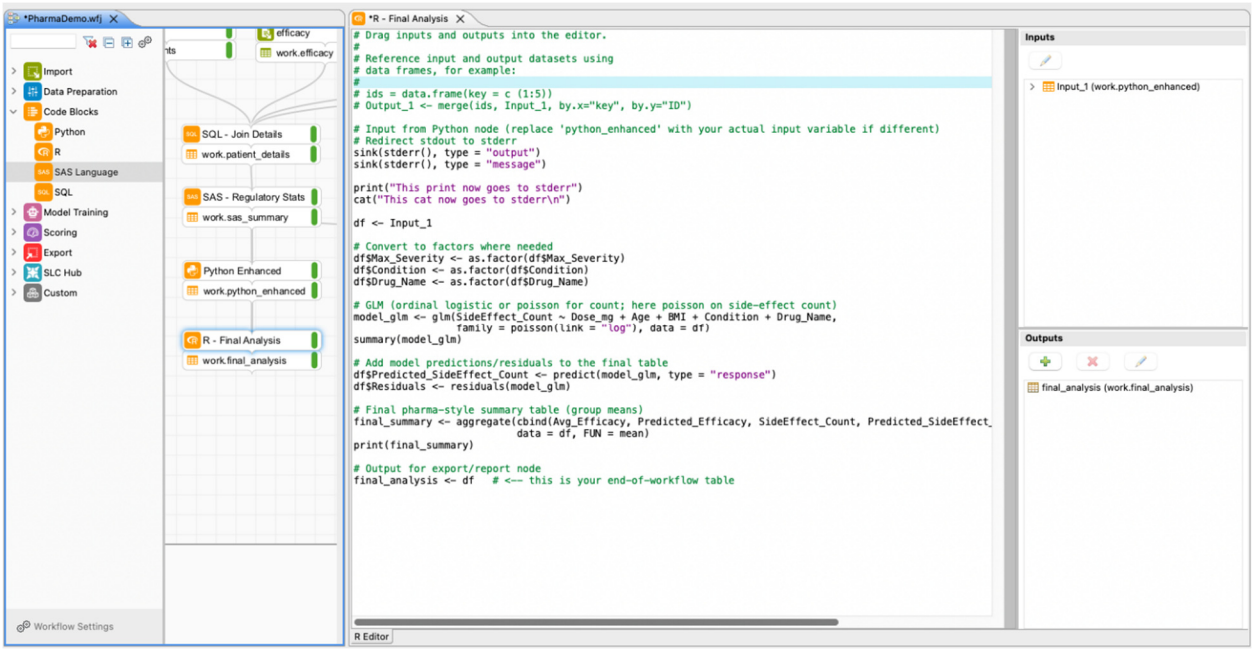
PROJECT STRUCTURE

The demonstration project contains four primary scripts:

```
project_workspace/
  extract_data.sql
  prepare_data.sas
  model_data.py
  analyze_results.R
```

Each script is written in its native language and accessed through a shared editor interface with language-specific support.

Screenshot 2: View of a particular scripting language in the editor



This structure allows analysts to view the entire workflow in one place, making it easier to understand how each stage of the analysis depends on the others.

COORDINATED EXECUTION

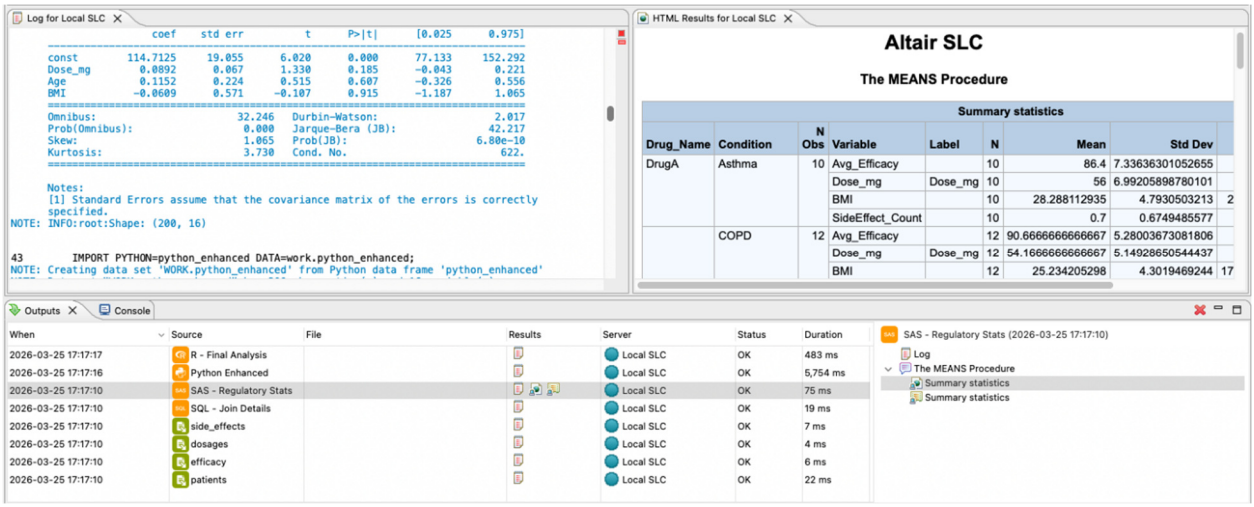
The unified environment simplifies execution. Analysts can run each script individually or execute them sequentially as part of a larger pipeline.

For example, the workflow might proceed as follows:

1. The SQL script extracts source data and stores the result in a staging table or intermediate file.
2. The SAS script reads the extracted data, performs validation and transformation steps, and writes a cleaned dataset.
3. The Python script performs modeling or feature engineering using libraries such as pandas or scikit-learn.
4. The R script generates statistical summaries or visualizations of the results.

This structure makes data flow across stages easy to trace.

Screenshot 3: Demonstration of log/output panels and/or workflow being or already executed



MANAGING DEPENDENCIES AND ARTIFACTS

The workspace also standardizes artifact management across languages. Intermediate datasets, configuration files, and output results can all be stored alongside the scripts that generate them.

For example, a project directory might include folders such as:

```
data/
scripts/
results/
logs/
```

This structure helps ensure that analytical workflows remain organized and reproducible. New team members can quickly understand the project layout and identify the role of each component in the pipeline. In addition, logs and execution output from different languages can be viewed within the same environment, making it easier to diagnose errors and verify that each stage of the workflow completed successfully.

OBSERVATIONS FROM THE DEMONSTRATION

While the underlying scripts remain written in their native languages, the unified environment changes how analysts interact with the workflow. Rather than moving between multiple tools, developers can maintain focus within a single interface while still leveraging the strengths of multiple programming languages.

The next section examines the practical benefits that this type of unified environment can provide for statistical programmers and analytics teams working in polyglot environments.

BENEFITS FOR STATISTICAL PROGRAMMERS AND ANALYSTS

A unified development environment reduces the operational overhead associated with multi-language workflows by consolidating development activities within a single interface.

IMPROVED PRODUCTIVITY AND USABILITY

A primary benefit is reduced context switching between tools. In traditional environments, analysts move between editors, execution environments, and debugging interfaces, each with its own conventions and interaction patterns. Consolidating these activities within a unified interface allows analysts to focus on analytical tasks rather than tool management, shortening feedback loops and improving development efficiency.

IMPROVED TRANSPARENCY AND REPRODUCIBILITY

A unified workspace improves visibility into the structure of cross-language workflows. When scripts and artifacts are co-located within a single project, it becomes easier to trace how data moves between stages of extraction, transformation, modeling, and reporting. This clarity simplifies maintenance and makes workflows more reproducible, particularly in collaborative environments.

REDUCED COGNITIVE AND OPERATIONAL OVERHEAD

Working across multiple programming languages inherently requires maintaining awareness of different syntax, libraries, and execution models. A unified environment reduces additional overhead by standardizing how analysts interact with those languages. While the languages themselves remain distinct, core development activities—editing, execution, logging, and project organization—follow consistent patterns, making multi-language workflows easier to navigate over time.

PRACTICAL CONSIDERATIONS AND LIMITATIONS

While unified development environments can simplify many aspects of polyglot analytics, they are not a universal solution for every analytical workflow. Organizations considering this approach should evaluate both the benefits and the practical constraints that may arise when integrating multiple languages into a single development environment.

WHEN A UNIFIED IDE IS A GOOD FIT

Unified analytical workspaces tend to be most effective in environments where analysts regularly work across multiple languages within the same project. In these situations, consolidating development activities within a single interface can significantly simplify workflow management.

Typical scenarios where a unified IDE can be particularly helpful include:

- **Cross-language analytical pipelines** in which different stages of an analysis are implemented in different languages.
- **Collaborative team environments** where analysts with different language expertise contribute to the same project.
- **Projects with shared artifacts**, such as intermediate datasets, configuration files, or output reports that must be accessed by scripts written in multiple languages.
- **Analytical workflows that require frequent iteration**, where quickly moving between different parts of a multi-language pipeline can improve development efficiency.

In these contexts, a unified IDE can help analysts maintain a clearer view of how different components of the workflow interact while reducing the overhead associated with managing multiple development tools.

INTEGRATION WITH EXISTING ENVIRONMENTS

Adopting a unified development environment does not necessarily require replacing existing tools or workflows. In many organizations, analysts will continue to use specialized tools for certain tasks while using a unified workspace to coordinate multi-language analytical projects.

For example, database administrators may still rely on dedicated database management tools for schema design or performance tuning, while data scientists may occasionally use specialized environments for advanced model development. A unified IDE can serve as the central location where the operational workflow is assembled and executed.

In this sense, the unified workspace acts less as a replacement for other tools and more as an orchestration layer that helps bring different components of a project together.

LIMITATIONS AND TRADE-OFFS

There are also situations where a unified development environment may introduce challenges. One potential concern is that multi-language environments can become complex if the underlying runtime environments for each language must be carefully managed. Dependencies, package versions, and execution configurations may still need to be maintained separately for each language even when development occurs within a shared interface.

Another consideration is organizational adoption. Analysts who are already comfortable with their preferred tools may be hesitant to transition to a new development environment, particularly if existing workflows are well established.

Finally, some highly specialized development tasks may still benefit from dedicated environments tailored to a particular language or framework. In these cases, the unified IDE may serve primarily as a coordination layer rather than as the primary development tool.

Despite these limitations, unified development environments can provide a practical way to reduce complexity in polyglot analytical workflows, particularly in projects where multiple languages must be used together on a regular basis.

KEY TAKEAWAYS

Modern analytical workflows rely on multiple languages but introduce complexity when distributed across tools. Unified environments address this complexity by consolidating development into a single workspace. By allowing analysts to write, execute, and organize scripts written in different languages within the same interface, unified IDEs can simplify workflow management and reduce the friction associated with multi-tool development.

The demonstration presented in this paper shows how a workflow involving **SQL, SAS, Python, and R** can be coordinated within a single project environment while allowing each language to perform the tasks for which it is best suited.

Several practical observations emerge from this approach:

- **Polyglot analytics is often unavoidable**
- **Tool fragmentation increases complexity**
- **Unified environments simplify cross-language workflows**
- **Unified tools complement existing ecosystems**

As the number of languages used in analytical environments continues to grow, tools that simplify cross-language development will play an increasingly important role in maintaining productive and maintainable workflows.

CONTACT INFORMATION

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

Triam Ltd – https://triamltd.com/
David Ward – david.ward@triamltd.com
Troy Wolfe – troy.wolfe@triamltd.com