Cytel

ML14: From Automation to Audit-Readiness AI's Growing Role in Statistical Programming

Manoj Kumar Maripally
Principal Statistical Programmer
Cytel Inc

Disclaimer

Please note, the views expressed in this presentation are my own and do not necessarily represent the official position of Cytel.
The content presented is based on publicly available information.

Agenda

- Statistical Programming for Modern Drug Development
- Automation in Clinical Trials Statistical Programming
- The Pharmaverse Open-source Ecosystem
- Metadata Driven TFL Output Generation
- Validation Framework
- Automation and AI/ML Applications
- Large Language Models (LLMs) for Statistical Programming
- AI/ML Hybrid Workflow
- AI/ML Implementation in Statistical Programming
- Strengths vs Limitations of AI/ML Implementation
- Summary

Statistical Programming for Modern Drug Development

Clinical development relies heavily on manual statistical programming

- Clinical trials are growing rapidly in Data volume, Protocol complexity and Regulatory requirements
- Statistical programmers spend more time on data prep and cleaning
- Manual approaches are slow and quality risk, cannot meet compressed trial timelines (10–15 days post database lock)

Challenges

- Data processing bottlenecks
  - Manual validation and cleaning dominate workflows
- High resource demands
  - Multiple resources needed for validation and QA
- Quality & consistency issues
  - Error rates in complex analyses
- Documentation burden
  - Significant time spent on compliance and reporting

Automation in Clinical Trials Statistical Programming

Statistical programming for clinical trials remains labor-intensive and error-prone, particularly for TLF (Tables, Listings, Figures) and study specific ADaMs generation and validation

Recent developments include:

- Metadata-driven architectures that separate specifications from implementation
- Open-source R package ecosystems ('pharmaverse') providing modular, validated components aligned with CDISC standards (SDTM IG 3.4, ADaM IG 1.3, Define-XML 2.1, Analysis Results Standard 1.0)
- Machine learning for automated data mapping
- Emerging applications of large language models (LLMs) for code generation, Natural Language Processing (NLP) for text as input and Deep learning for pattern detection and prediction
- Validation automation through risk-based strategies and CI/CD
- Platform considerations for R, SAS, and Python in regulatory contexts
- AI/ML applications for code generation

Statistical programming automation efforts—particularly TLF generation, validation frameworks, and AI-assisted code development— are still emerging

The Pharmaverse Open-source Ecosystem

End-to-end, production-ready toolchain supporting CDISC-compliant clinical reporting

- **Dataset Generation (R)**
  - *admiral (★★★):* CDISC-aligned ADaM derivations (200+ functions)
  - *REDCap2SDTM (★★):* Automated SDTM mapping with major time savings
- **TLF Generation (R / Shiny)**
  - *rtables, Tplyr (★★★):* Metadata-driven, high-quality table production
  - *TLFQC (★★):* Codeless TLF creation with integrated QC
- **Submission & Validation (R / Java)**
  - *xportr, Pinnacle 21 (★★★):* Submission-ready XPTs and CDISC validation
  - *defineR (★★):* Automated Define-XML 2.1
- **Workflow Orchestration (R / Python)**
  - *targets, Snakemake (★★–★★★):* Reproducible, dependency-tracked pipelines

⋆⋆⋆ = regulatory submissions; ⋆⋆ = production use

Modern clinical reporting can be fully automated using **open-source, production-mature tools**, enabling faster delivery, improved quality, and scalable, metadata-driven workflows.

Cytel

Metadata Driven TFL Output Generation

Clinical study reporting requires hundreds of Tables, Listings, and Figures (TFLs) that frequently change, making traditional manual programming time-consuming, error-prone, and difficult to maintain at scale.

A **SAS-based, metadata-driven framework ("atlas")** that leverages **CDISC Analysis Results Standard (ARS)** metadata embedded in TFL shells to automatically generate fully executable TFL programs

- ARS metadata is defined prospectively in TFL shells
- Metadata drives macro selection and parameterization
- atlas generates ready-to-run SAS programs for each TFL

Benefits

- Significant reduction in manual programming
- Consistent, traceable, and submission-ready outputs
- Rapid adaptation to TFL changes during study execution
- Improved oversight and productivity for programming teams

This framework shifts TFL programming from **code-centric to metadata-centric**.

Validation Framework

Traditional validation approaches relied on independent double programming, provides confidence on quality of results but drawbacks:

- Resource intensity (effectively doubling programming effort)
- Redundancy (much effort replicates work rather than addressing high-risk areas)
- Late error detection (discrepancies discovered late require costly rework)

Risk-Based Validation Frameworks

Align ICH Q9 (Quality Risk Management) principles

Output criticality determines QC intensity:

- High-risk analyses (primary efficacy, key safety) receive full double programming
- Medium-risk outputs use peer review with automated testing
- Low-risk outputs leverage automated testing alone

Risk assessment dimensions:

- Output criticality (primary endpoints receive intensive validation)
- Statistician performs independent validation through raw-to-TFL programming
- Change frequency (stable codes require less validation)
- Regulatory scrutiny (analyses supporting labelling claims receive enhanced validation)

Automation and AI/ML Applications

Automation must maintain ALCOA+ data integrity principles: Attributable (clear authorship), Legible (readable outputs), Contemporaneous (timestamped), Original (source preservation), Accurate (verified correctness), plus Complete, Consistent, Enduring, and Available

Automated systems can enhance ALCOA+ compliance through immutable audit trails via version control, automated provenance tracking, reproducible execution environments, and systematic documentation generation

Current AI/ML applications

- Mapping automation (ML models predicting CDISC variable mappings)
- Code generation (LLMs generating statistical programming code)
- Protocol parsing (NLP extracting structured information)
- Anomaly detection (ML identifying data quality issues)

Machine Learning for Data Mapping

- Effective models use variable-level features (name, label, data type), contextual features (domain membership, database structure), and semantic features (text embeddings using BERT)
- Neural networks achieved the highest accuracy for complex mapping relationships
- Random forests provided interpretable feature importance

Large Language Models (LLMs) for Statistical Programming

Current Capabilities

LLMs (e.g., GPT-4, Claude, ClinicalBERT, GatorTron) support code drafting, specification interpretation, and code review

- General-purpose models excel at complex reasoning
- Domain-specific models outperform on clinical NLP tasks

Performance in Statistical Programming

Routine programming tasks, Higher performance for standardized, repetitive operations and Real-world deployments show variable success

Key Limitation

- Reproducibility challenges due to non-deterministic outputs
- Limited embedded CDISC & therapeutic-area knowledge without fine-tuning
- No formal regulatory guidance for AI-generated code

Best-Practice Positioning

- Treat LLMs as “first-draft assistants”, not autonomous programmers
- Align use with risk-based validation and software assurance principles

AI/ML Hybrid Workflow

Protocol / SAP (Text)
↓
NLP
↓
Structured metadata
↓
LLM
↓
Statistical code (SAS/R/Python)
↓
ADaM datasets
↓
Deep Learning / ML
↓
Classical statistical summaries

LLMs work at the interface between human intent and statistical code

LLMs reduce manual programming effort but still require human validation

NLP focuses on unstructured text → structured statistical inputs

NLP acts upstream, feeding clean inputs into statistical programming

Deep learning is used less for classical inference, more for pattern detection and prediction

Deep learning complements statistics; it doesn't replace it

AI/ML Implementation in Statistical Programming

Example 1: TFL Automation	• NLP reads mock shell • LLM generates SAS code • Programmer reviews & validates • Output traced to ADaM
Example 2: QC Automation	• Deep learning flags unusual distributions • LLM explains why they're unusual • Programmer decides if it's real or an error

Example 3: Model Exploration	• Deep learning finds nonlinear effects
	• Statistician formalizes with:
	  • Cox model
	  • Mixed model
	  • Bayesian model

Strengths Vs Limitations of AI/ML Implementation

Technique	Strength	Limitation
LLMs	Code & explanation speed	Hallucination risk
NLP	Extracts meaning from text	Domain tuning required
Deep learning	Complex pattern detection	Low interpretability
Classical stats	Interpretability, compliance	Less flexible

Regulatory & Good Programming Practice (GPP)

In regulated environments:

- AI output is not final
- Traceability must be maintained
- Human review is mandatory
- Models must be explainable or justified

LLMs are treated as **programmer productivity tools**, not analysis engines

LLMs = statistical programming copilots

NLP = bridge from text to data

Deep learning = pattern discovery engine

Together, they speed up programming, improve consistency and expand analytical insight, while statisticians retain control

Summary (1)

Automation has matured technically driven by open-source ecosystems (pharmaverse), industry consortia (PHUSE, CDISC, R Validation Hub), growing regulatory acceptance of R

Adoption is uneven, large pharma leads and smaller sponsors face resource & skill constraints
Major disconnect: Validation dominates practice

Key enablers

- Strong metadata governance & standardization
- Reusable component libraries
- Modern engineering practices (Git, CI/CD, automated testing)
- Phased implementation with pilot studies
- Proven regulatory pilots (FDA R submissions)

Key Barriers

- Legacy SAS investments & migration inertia
- Skill gaps (SAS → R/Python + metadata engineering)
- Regulatory uncertainty → risk aversion

Statistical programmers shift from manual coding to metadata, automation and quality engineering

Inspection Readiness: Validation rationale & risk assessments, reproducibility demonstrations, traceability: SAP → code → data → outputs and automation must strengthen ALCOA+ compliance

Summary (2)

Automation and AI implementation struggle with domain-specific reasoning, contextual interpretation of results, and human oversight remains essential

AI will not replace statistical programmers, it will automate routine coding, elevate human roles to higher-order analysis and should be used as a tool not a substitute

The profession's longevity is secure—but transformed
Professionals who adapt will see better career prospects and higher-value roles

Upskilling Priorities

Technical skills

- Programming: R, Python (and emerging languages)
- Data visualization
- Data science & machine learning fundamentals

Soft skills

- Collaboration & communication
- Critical thinking & problem solving
- Continuous learning & adaptability

References

Automation in Clinical Trial Statistical Programming: A Structured Review of TLF Generation, Validation Frameworks, and AI/ML Integration (2020–2025). Jaime Yan, Jason Zhang, Tingting Tian

Upputuri, V. (2025). Leveraging AI/ML in Statistical Programming: Enhancing Efficiency, Compliance, and Insights in Clinical Trials. International Journal of Research in Computer Applications and Information Technology

PHUSE Paper PP01 The Evolving Role of Statistical Programmers in the Pharmaceutical Industry: Impacts of Artificial Intelligence

https://pharmaverse.org/

Cytel

Your comments and questions are valued and encouraged. You may contact author at:

Manoj Kumar Maripally
Cytel Inc.
Email: Manojkumar.Maripally@cytel.com

Thank You