

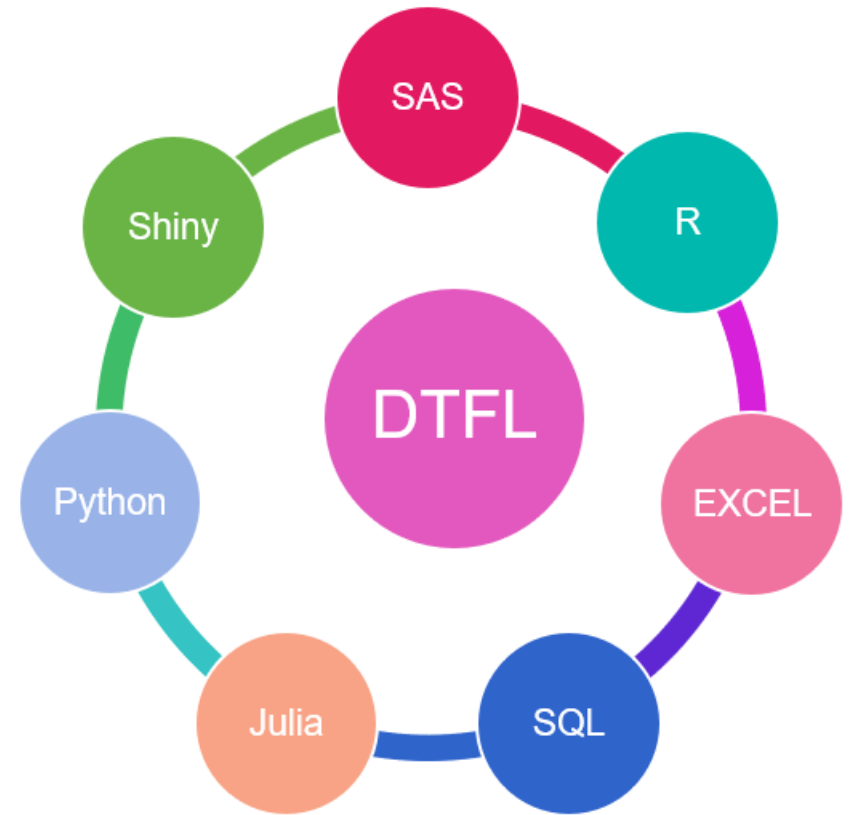


Multilingual Programming Language: A Paradigm Shift in Clinical Programming

Pavan Kumar KP

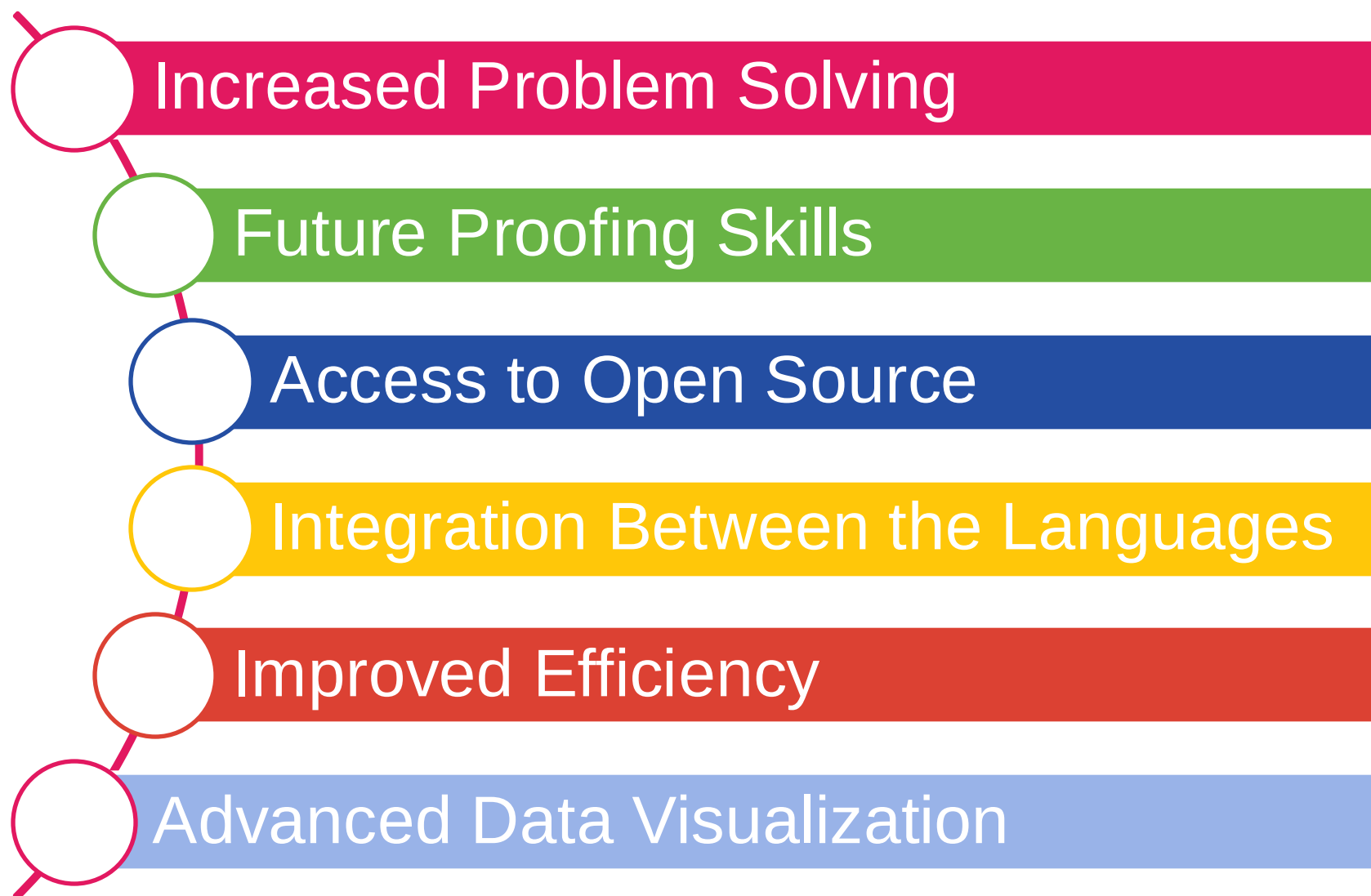
What is Multilingual Programming ?

- Multilingual programming refers to the practice of using multiple programming languages within a single project, software system, or organization. Instead of relying on one language for all tasks, developers choose the most appropriate language for specific functions based on the strengths and features of those languages.
- In Clinical Programming, multilingual programming refers to the practice of seamlessly integrating multiple programming languages—such as SAS, R, Python, SQL, or any other language—within a single project or workflow. The goal is to harness the strengths of each language to create a Datasets (SDTM or ADAM for Analysis) and TFLs for the Reporting.



Why it is important ?

- In today's interconnected and globalized world, multilingual programming is becoming increasingly crucial. It is critical and impossible to ignore, particularly in clinical programming. Many pharmaceutical giants are transitioning towards using R along with the latest open-source technologies.
- With the benefits of open source, collaboration across companies is occurring in projects like ADMIRAL and ADMIRAL ONCO. If these trends continue, there could be an end to the dominance of SAS within clinical programming.



Does FDA have any issue ?

The **FDA** (U.S. Food and Drug Administration) does **not** have any inherent concerns if the data is fully programmed in **R**. In fact, there have been significant developments in using R for regulatory submissions. :

- **R-Based FDA Submissions:**

- The **R Consortium** successfully submitted an **R-based test submission pilot 3 package** through the **FDA eCTD gateway** on **August 28, 2023**. This submission package followed **eCTD specifications** and included **R scripts** to generate **ADaM (Analysis Data Model) datasets** and **TLFs (Tables, Listings, Figures)**.
- To our knowledge, this was the **first publicly available R-based FDA submission package** that included R scripts for generating ADaM datasets and TLFs.
- The FDA CDER staff evaluated this submission, marking an important milestone for future open-source language-based submissions.
- GSK (SOLARA) and Novo Nordisk also submitted the full CRT package using R.

Useful Links :

- R packages and Shiny for FDA clinical trial → <https://posit.co/blog/novo-nordisks-journey-to-an-r-based-fda-submission/>
- Other R Package to FDA : <https://r-consortium.org/posts/news-from-r-submissions-working-group-pilot-3/index.html>

Continue..

- **Previous Successful R-Based Submissions:**

- Prior to this, the **R Submissions Working Group** completed a successful **pilot 1 eCTD submission** using R. The initial submission was made on **November 22, 2021**, and the final response letter from the FDA was received on **March 14, 2022**. The pilot 1 submission included a **proprietary R package**, R scripts for analysis, an **R-based analysis data reviewer guide (ADRG)**, and other required eCTD components.
- Throughout the process, R scripts were successfully submitted through the eCTD gateway, and the FDA staff verified the results by installing proprietary and open-source R packages and independently evaluating the programming in R.

- **Myth Debunked:**

1. There has been a persistent myth that the FDA restricts the use of R. However, the **FDA's Statistical Software Clarifying Statement** declares that **any suitable software** can be used for regulatory compliance.
2. While some data-exchange regulations may specify the use of the **XPT format** (also known as **SAS XPORT**), this format is an **open standard** and **not restricted to SAS**.

In summary, the FDA recognizes the value of R and accepts submissions based on open-source programming languages. Sponsors can confidently use R for regulatory submissions, and the successful examples mentioned above demonstrate its viability.

How to Covert the SAS Code into R

- **Manual Conversion:**
 - Understand the logic and functionality of your SAS code.
 - Rewrite the equivalent logic in R syntax.
 - Pay attention to differences in data handling, functions, and libraries between SAS and R.
 - Test your R code thoroughly to ensure it produces the same results as your original SAS code.
- **Using AI Conversion Tools (May Not be Fully validated Tool):**
 - **Usage of GEN AI Tools like : CHAT GPT, COPILOT...**
 - **CodeConvert AI** provides a free online converter that can automatically convert SAS code to R.
 - Visit the → <https://www.codeconvert.ai/sas-to-r-converter>
 - Type or paste your SAS code into the input box. → Click the “Convert” button.
 - The resulting R code will be displayed in the output box.

Remember that automated tools may not always produce identical code due to variations in language features and conventions. Manual review and adjustments are often necessary.

Company's Approach in Adapting Open Source?

- Use R for both Production and Validation.
- Continue to use the Existing SAS Macros for the Production, and perform the Validation using R.
- Usage of Selective R Packages (TIDYVERSE, HEAVEN, DPLYR...) / Frozen R packages. This will avoid the usage of the Unvalidated R Packages.
- Replacing the XPT datasets into JSON files
- Developing In-House Open-Source tools to replace the traditional way of Programming... GSK has developed some advance tools for Data Visualization, SAS Outputs to PDF Bundles....
- **Multilingual Platforms?**
 - DOMINO (Provides access to both SAS and R)
 - SAS VIYA
 - And Many More

How can Programmers Adapt ?

- **Multilingual is not new, as technologies changes, it is our duty to adapt to the new tools and get use to new ways of working. “OPEN SOURCE IS THE FUTURE” and “ADAPTABILITY is the KEY”.**
- Programmers may need to upskill or reskill to stay competitive. The role of programmers is shifting towards more advanced analytics, such as machine learning, AI, and big data analytics.
- **Lifelong Learning and Adaptability :**
 - **Rise of R and Python:** R and Python have become the most popular languages in data science and analytics, offering extensive libraries and frameworks for statistical analysis, machine learning, and data visualization
 - **Integration with SAS (SAS VIYA → PROC IML and PROC FCMP):** SAS has recognized the importance of open-source tools and has developed integrations with R and Python.
 - **SAS Viya:** SAS has developed SAS Viya, a cloud-native platform that integrates with open-source technologies and provides a more modern, scalable environment for data analytics. SAS programmers who adopt SAS Viya can benefit from this hybrid approach, leveraging both SAS's proprietary strengths and open-source tools.

- **Shift Towards Higher Abstraction Levels**
 - **Low-Code/No-Code Platforms:** The rise of low-code and no-code platforms is making it easier for non-programmers to build applications, automating many routine coding tasks. **Example** : Cursor, Codeium, Zoho Creator, Bubble, Airplane, Kissflow, Appian....
- **Evolving Programming Languages and Paradigms**
 - **AI-Assisted Development:** Tools like Github, Gitlab, Chatgpt, GitHub Copilot, which use AI to assist in coding.
 - **Automating Routine Tasks:** AI Tools replace many routine programming tasks.
 - **Natural Language Generation (NLG)** : can be used to automatically create summary tables or reports, significantly speeding up routine statistical output generation.
 - **Automation of CDISC Conversion (SDTM, ADaM)** : ML models can suggest or directly apply mappings from raw data to SDTM or ADaM datasets.
 - **Automated Table, Listing, and Figures (TLF) Generation** : AI can automatically cross-check variable values, identify discrepancies between programming outputs and clinical databases, and highlight anomalies, reducing the time spent on manual QC.
 - **Automated Data Cleaning and Transformation** : AI algorithms can automatically identify outliers or errors in datasets and suggest or perform corrections, minimizing the need for manual checks.
 - **Metadata-Driven Programming** : AI can auto-generate data manipulation and analysis scripts based on a study's metadata, reducing the need to manually rewrite code for each study.
 - **Automating FDA/EMA Submission Preparation or Providing Response to FDA/EMA Questions.**

Benefits of Multilingual Programming

- Adopting open-source tools can attract many Gen Z programmers and potentially transform the traditional approaches to programming.
- **Access to Cutting-Edge Tools to analyze trends and Outcomes** : R and Python offer extensive libraries for bioinformatics, machine learning, and data visualization (e.g., R's ggplot2 or Python's scikit-learn)
- **Data Visualization** : Replace the traditional graphs with Shiny Apps. Dashboards..
 - Adverse Event Reporting with Interactive Dashboards (Python & R)
 - Survival Curves and Kaplan-Meier Plots (R)
 - Heatmaps for Biomarker Data (Python)
 - Patient Recruitment and Retention Funnel (Python)
 - Waterfall Plots for Tumor Response (Python & R)

Challenges of Multilingual Programming

Challenges	R	SAS
Industry-Specific Usage : Programmers may face industry-specific demands that Favor one tool over the other, requiring them to be proficient in both even if they have a preference or more experience with one.	R : is widely used in academia, research, and industries like finance and marketing.	SAS : is heavily used in Clinical Domain due to its reliability and compliance with industry standards.
Licensing and Cost : Budget constraints can dictate which tool is used.	R : is open-source and free to use, making it accessible for a wide range of users.	SAS : is a commercial software with significant licensing costs, which can limit its use to organizations with sufficient budgets.
Syntax and Language Differences Switching between these languages requires constantly adapting to different syntax rules and conventions, which can slow down development.	R : is a scripting language designed primarily for statistical computing and graphics. It has a more flexible syntax, and users often leverage its wide range of packages for diverse applications.	SAS : It often involves using specific procedures (PROC) and Data Step.
Performance Considerations : Programmers need to be mindful of these performance differences, especially when working with large-scale data and choose the appropriate tool for the task.	R : can be slower than SAS when dealing with very large datasets because it typically holds data in-memory.	SAS : is optimized for handling large datasets and is often faster in such scenarios, particularly in industries where performance with big data is critical.

Continued..

Challenges	R	SAS
Error Handling and Debugging : The approaches to debugging and error handling are different, requiring programmers to learn and adapt to both systems' methods.	R : provides interactive debugging tools, and error messages are often descriptive. However, debugging can become complex due to the language's flexibility.	SAS : has its own error-handling mechanisms, but the error messages can sometimes be cryptic, and the batch-processing nature of SAS can make debugging more challenging.
Graphical Capabilities : Creating consistent visual outputs across both platforms can be difficult, especially when high customization is required.	R : R excels in data visualization with packages like ggplot2, which allows for extensive customization and creation of complex visualizations.	SAS : SAS also offers graphical capabilities, particularly with PROC SGPLOT and PROC GPLOT, but they are generally considered less flexible than R's options.
Learning Curve and Community Support : Balancing the learning curve of both tools and finding the right resources can be time-consuming.	R : R has a large, active community, and extensive resources are available online. However, its vast ecosystem can be overwhelming for beginners.	SAS : SAS has a strong presence in certain industries and a more structured support system, but it may not have as extensive a community as R for more diverse applications.

► Future : Programmers Managing AI Coding Agents.

- I think in the Future, Programmers won't write code at all. There are many auto coding platforms which write a code based on our commands. (Just speak and these Platforms write code for you).
 - Programming Tools : Will be integrated with GEN AI Tools.
- It may be Full of AI BOTS, ask BOTS everything and make a minor tweaks to the code. The ERA of memorizing the syntax will be ended.
- **Should you Panic :**
 - The Programmers who are proficient with coding without code generators can never be replaced.
 - People who built programming languages and foundational tools and alike to build apps.
 - There may be many Programmers will be hired to find the flaws In the Applications.
 - Domain Knowledge Play a Vital Roles compared to Programming Strength.
- **Future would be more and more applications and standards.**

Contact information

Name:	Pavan Kumar KP
Department:	Biostats
email:	pavan.10.kumar@gsk.com

GSK



BACK UP SLIDES

Why it is important ?

- **Increased Problem-Solving Flexibility:** Knowing multiple programming languages allows developers to choose the best tool for a specific task. Different languages have unique strengths and are suited to different types of problems
- **Enhanced Career Opportunities:** Being proficient in multiple languages makes you more attractive to employers. Many companies look for developers who can work across different technologies and platforms
- **Future-Proofing Skills:** Technology evolves rapidly, and languages that are popular today might not be in the future. Learning multiple languages helps ensure your skills remain relevant
- **Intellectual Stimulation:** Learning new languages can keep your work interesting and prevent monotony. It also encourages you to think in different ways and approach problems from new angles
- **Legacy Integration :** Many organizations have legacy systems written in older languages. Being multilingual helps in integrating these systems with newer technologies without the need for a complete rewrite.
- In today's interconnected and globalized world, multilingual programming is becoming increasingly crucial. It is critical and impossible to ignore, particularly in clinical programming. Many pharmaceutical giants are transitioning towards using R along with the latest open-source technologies.
- With the benefits of open source, collaboration across companies is occurring in projects like ADMIRAL and ADMIRAL ONCO. If these trends continue, there could be an end to the dominance of SAS within clinical programming.

Benefits of Multilingual Programming

- Adopting open-source tools can attract many Gen Z programmers and potentially transform the traditional approaches to programming.
- **Access to Cutting-Edge Tools to analyze trends and Outcomes** : R and Python offer extensive libraries for bioinformatics, machine learning, and data visualization (e.g., R's ggplot2 or Python's scikit-learn)
- **Data Visualization** : Replace the traditional graphs with Shiny Apps. Dashboards..
 - **Adverse Event Reporting with Interactive Dashboards (Python & R)** : Python's **Plotly** and **Dash** or R's **Shiny** can be used to create interactive dashboards that allow users to filter AEs by severity, patient demographics, or treatment groups..
 - **Survival Curves and Kaplan-Meier Plots (R)** : R's **ggplot2** and **survminer** packages are excellent for creating sophisticated **Kaplan-Meier survival curves** used in time-to-event analysis.
 - **Heatmaps for Biomarker Data (Python)** : When dealing with large datasets involving genetic or biomarker data, Python's **seaborn** or R's **pheatmap** can be used to create **heatmaps**.
 - **Patient Recruitment and Retention Funnel (Python)** : Using Python's **Plotly** or R's **ggplot2**, you can create a funnel chart to visualize patient recruitment and retention over the course of a clinical trial.
 - **Waterfall Plots for Tumor Response (Python & R)** : Waterfall plots are particularly useful in oncology trials for showing individual patient responses to treatments, especially when looking at tumor size reductions. Python's **Matplotlib** or R's **ggplot2** can be used to create these plots, with clear visual differentiation between patients showing complete response, partial response, or progression.

Supervised Learning Models for Data Mapping

Supervised learning models can be trained to learn mappings between **raw clinical data** and **CDISC-compliant datasets** based on historical data. These models learn the relationships between variables in raw data and their corresponding variables in the SDTM or ADaM format. **Example Models:**

- **Decision Trees** : can be trained to learn rules for mapping specific raw variables to their respective CDISC variables. The tree structure can capture complex relationships, such as different mappings for different study types.
 - Mapping different patient ID formats or date variables to standard formats in SDTM/ADaM.
- **Random Forests** : A random forest, an ensemble of decision trees, improves mapping accuracy by combining the predictions from multiple decision trees trained on different parts of the dataset. It can handle variability and small differences in data, which is common in clinical datasets.
 - Mapping various demographic, lab, or adverse event data into structured SDTM datasets while accounting for slight variations across studies.
- **Gradient Boosting Machines (GBM)** : GBM builds a series of models sequentially, with each new model correcting the errors made by the previous ones. This results in more precise mapping rules.
 - Learning how to handle complex variable transformations, such as converting raw lab data into standardized units for ADaM datasets.

▶ Sequence-to-Sequence Models for Complex Mappings

Sequence-to-sequence (Seq2Seq) models, commonly used in language translation, can also be applied to translating raw clinical data into SDTM or ADaM datasets. These models are particularly useful when complex mappings and multiple steps are needed to transform raw data into the required format. **Example Model:**

- **Recurrent Neural Networks (RNNs) / Long Short-Term Memory (LSTM) Networks** : LSTMs can handle sequence data, where the order of operations is important. These models can learn how to apply multiple transformations to raw data, such as handling nested fields or complex patient timelines.
 - Converting longitudinal patient data (e.g., multiple visit dates, treatment cycles) into a structured format like SDTM **VS (Vital Signs)** or **EX (Exposure)** domains.