Paper OS08

Validation Unveiled: Striking the Balance for Data Integrity and Compliance in R, Python, and SAS

Sy Truong, Meta-Xceed, Inc., Fremont, USA

ABSTRACT

The Statistical Computing Environment (SCE) has expanded beyond SAS®, encompassing languages such as Python and R. While the validation of SAS is widely recognized in Pharma, validating these additional languages is less familiar. This paper outlines new approaches for testing and validating Python and R, drawing upon concepts from established SAS validation best practices.

- Risk-Based: Python and R exploratory analysis differs from SAS, leading to less formal validation.
- User Acceptance: Similar to SAS OQ testing, executing user-representative Python and R scripts on user data is effective.
- Performance Qualification: Unique performance considerations are tested when multiple users run Python, R, and SAS code on large datasets in parallel.
- AI-Generated Test Scripts: Tools like ChatGPT aid in generating useful Python and R tests while safeguarding proprietary data.

Python and R, being open-source tools, have distinct use cases compared to SAS. To ensure data integrity, combining SAS validation techniques with fresh approaches is crucial to establish a compliant Statistical Computing Environment.

RISK BASED VALIDATION

An important early step in the process of validation involves conducting a Risk Assessment to gauge potential risks inherent in utilizing systems such as SAS, Python and R for analyzing clinical trials data. A similar risk assessment process can be applied to Python and R, as compared to with SAS. This assessment focuses on how the system impacts data integrity, analysis accuracy, and strategies for mitigating these risks through rigorous testing and validation. While the methods share similarities, integrating AI can automate and streamline these steps. The following outlines the conventional steps in validating SAS systems alongside how AI automation can aid in the validation process for Python and R. The AI examples in this case uses ChatGPT version 3.5 as indicated by icon: followed by gray text.

Traditional Risk Assessment for SAS

- **STEP 1:** Arrange meetings with SAS system users to inquire about the risks they've encountered concerning data integrity and accuracy while conducting data analysis. Engage in discussions to gather insights into the specific challenges they've faced and how these relate to maintaining data integrity and ensuring accuracy in their analyses.
- **STEP 2:** Utilize Excel or MS Word to record and document the risk factors identified from the interview responses. Create structured documents or spreadsheets to comprehensively capture and organize the insights gleaned from the interviews regarding potential risks associated with data integrity and accuracy.
- **STEP 3:** Assign a risk score to each identified task based on predefined business rules. You can use numerical scoring within Excel or opt for a grading system during analysis and evaluation to gauge the level of risk associated with each task. This scoring system helps quantify and prioritize risks according to established criteria.
- **STEP 4:** Consolidate the risk assessment scores to evaluate the overall risk level for the system, by aggregating individual components and/or questions. This summary offers insights into the risk levels associated with each component or aspect. Then, generate an overall risk assessment score for the system as a whole, which serves as a guiding metric to determine the required level of validation necessary for the system.

AI Enhanced Risk Assessment for Python and R

- **STEP 1**: Besides interviewing users, similar interview-style questions can be directed towards Large Language AI models. Begin by defining the user's role, such as: "As a Python programmer involved in clinical trials..." or "As an R biostatistician analyzing pharmaceutical data...". Follow this by providing specific examples such as "when performing survival analysis using Kaplan-Meier...". The more detailed the example within the question, the more precise and relevant the risk assessment responses are generated. Conclude the prompt by asking the AI model to identify risks: "list the top 10 risks I might face when using the R system." This method extracts tailored risk assessments from AI models based on specified scenarios and roles. The following is an example response:

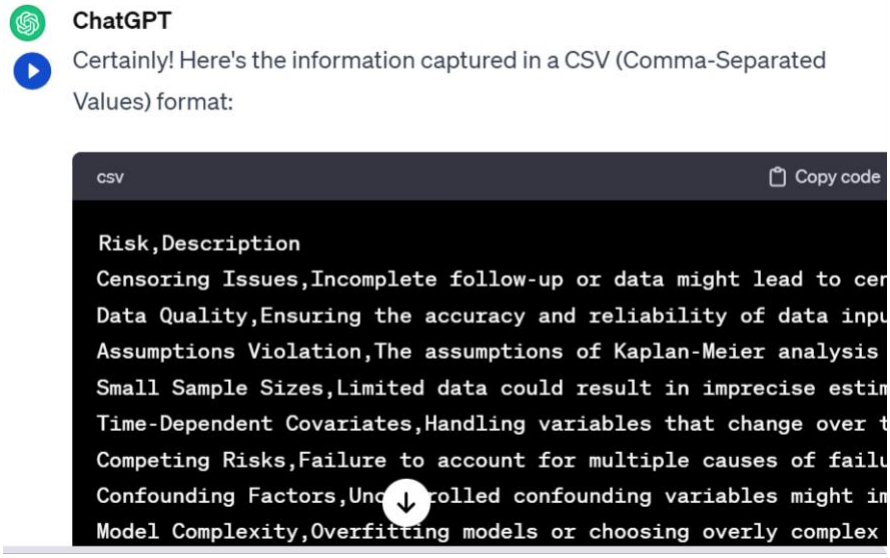
ChatGPT

When working with survival analysis using R's Kaplan-Meier method in the pharmaceutical industry, here are ten common risks you might encounter:

- **Censoring Issues:** Incomplete follow-up or data might lead to censoring, impacting the accuracy of survival estimates.
- **Data Quality:** Ensuring the accuracy and reliability of data inputs is crucial; any errors could significantly affect results.
- **Assumptions Violation:** The assumptions of Kaplan-Meier analysis (such as independent censoring) may be violated, affecting the reliability of results.
- **Small Sample Sizes:** Limited data could result in imprecise estimates, affecting the robustness of your conclusions.
- **Time-Dependent Covariates:** Handling variables that change over time requires careful consideration and appropriate methodologies.
- **Competing Risks:** Failure to account for multiple causes of failure could lead to biased estimates of survival probabilities.
- **Confounding Factors:** Uncontrolled confounding variables might impact the observed association between survival and covariates.
- **Model Complexity:** Overfitting models or choosing overly complex models might result in poor generalization to new data.
- **Model Validation:** Failing to validate the model appropriately could lead to misleading conclusions about the predictive performance.
- **Interpretation Challenges**: Communicating complex survival analysis results to non-statistical stakeholders might pose difficulties.

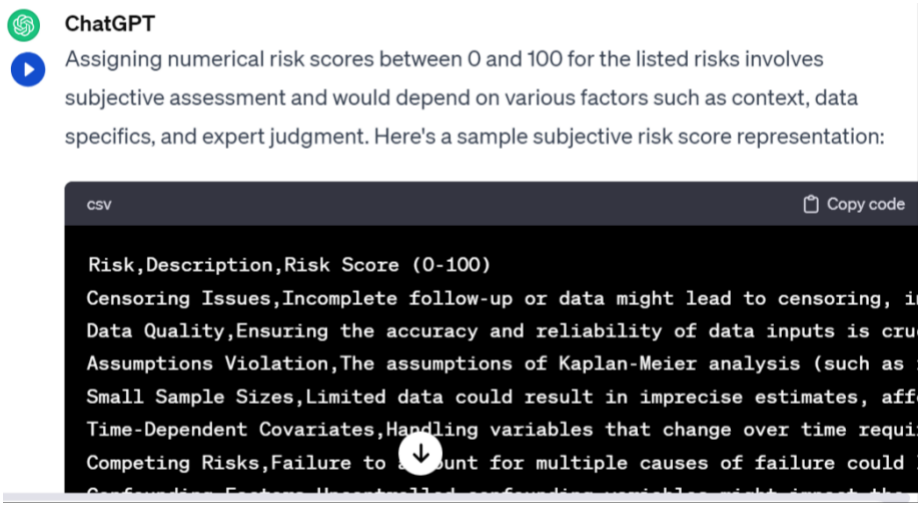
To mitigate these risks, it's vital to thoroughly understand the underlying assumptions of the Kaplan-Meier method, validate models rigorously, perform sensitivity analyses, and collaborate closely with domain experts for context-specific interpretations.

- **STEP 2**: The interview questions and corresponding answers can be collected and summarized in CSV (Comma-Separated Values) file format to simplify analysis. For instance, a follow-up prompt could be: "Please provide the above response in CSV format." Once generated, you can copy and paste the obtained results directly into an Excel spreadsheet for further examination and analysis. This method streamlines data handling and facilitates analysis by transferring interview data into a compatible Excel format.



- **STEP 3:** To ensure accurate risk assessment scores, established business rules or human judgment are necessary. Nonetheless, AI can offer valuable additional scores that can either be compared with or integrated into the final assessment. For instance, a prompt like: "What is the risk score, ranging from 0 to 100, for conducting survival analysis on clinical data using R (or specific risks mentioned)?" can be utilized. This approach combines human expertise with AI-generated insights to enhance the comprehensiveness of

risk evaluation.



- **STEP 4:** AI can assist in summarizing your analysis with the following prompts: “Write a summary of risk assessment based upon the following risk analysis: [paste csv]”.

ChatGPT

The risk assessment based on the provided analysis of various factors in survival analysis using Kaplan-Meier methodology is as follows:

- **Data Quality (80)**: Ensuring the accuracy and reliability of data inputs is identified as the highest-risk factor. Errors in data could significantly impact the outcomes of survival analysis.
- **Confounding Factors (80)**: Uncontrolled variables affecting the association between survival and covariates pose a considerable risk. Proper control and understanding of these factors are crucial.
- **Assumptions Violation (75)**: Violation of key assumptions in Kaplan-Meier analysis, such as independent censoring, could affect the reliability of the results and pose a significant risk.
- **Competing Risks (75)**: Failing to account for multiple causes of failure may lead to biased survival probability estimates, posing a substantial risk to the accuracy of the analysis.
- **Model Validation (75)**: Inadequate validation of the model might lead to misleading conclusions about predictive performance, posing a significant risk to the analysis's reliability.
- **Interpretation Challenges (70)**: Communicating complex survival analysis results to non-statistical stakeholders is identified as a risk due to potential difficulties in conveying the findings accurately.
- **Censoring Issues (70)**: Incomplete follow-up or data leading to censoring poses a moderate risk, impacting the accuracy of survival estimates.
- **Time-Dependent Covariates (70)**: Handling variables changing over time requires careful consideration, posing a moderate risk to the analysis.
- **Model Complexity (65)**: Overfitting or selecting overly complex models poses a moderate risk by potentially hindering generalization to new data.
- **Small Sample Sizes (60)**: Limited data, while a risk, is ranked lower as it may lead to imprecise estimates but is relatively less impactful compared to other factors.

This summary ranks risks based on their potential impact on the accuracy, reliability, and interpretability of survival analysis using Kaplan-Meier methodology in the pharmaceutical context. Prioritizing attention to high-risk factors, such as data quality and confounding variables, is crucial for robust and reliable analysis outcomes.

USER ACCEPTANCE TESTING

For every requirement gathered from the prior steps, a user acceptance test script can be used to validate the system's capability to meet these requirements. Seasoned system users can often provide many, if not most, of these test scripts. However, there might be cases where no test scripts exist, or you plan to supplement existing scripts with additional tests. AI can be instrumental in generating these supplementary test scripts when needed.

Traditional User Acceptance Testing

- **STEP 1:** Capture User Requirements – Conduct interviews with users to ascertain the necessary outputs and functionalities essential for generating the desired deliverables. Subsequently, document these requirements in a detailed, itemized MS Word file. This file serves as a comprehensive record outlining the outputs needed and the specific system functionalities required to produce the intended deliverable.

- **STEP 2:** Acquire or Create Test Scripts – Identify existing scripts from users that align with the documented requirements. Some scripts may need adjustments to operate within a validation environment or folder. For requirements lacking scripts, collaborate with users to develop new scripts tailored to meet those specific requirements. This ensures a comprehensive set of test scripts covering all documented requirements.

- **STEP 3:** Execute Scripts – Manually execute individual scripts within the validation environment. Alternatively, develop an additional script to automate the sequential execution of test scripts, ensuring they run in the correct order. This automation option can include features to capture relevant outputs as needed, streamlining the validation process.

- **STEP 4:** Summarize Test Results – If there are a limited number of scripts, manually examine the logs to pinpoint any issues. Review the output to verify if the tests align with the requirements. Initially, document the findings in an Excel sheet, then consolidate and summarize these results in a final MS Word document.

  However, if the logs or outputs are too extensive for manual inspection, create scripts to automatically analyze and summarize the findings from these logs and output files. This automated process ensures efficient handling and comprehensive summarization of results.

AI Enhanced User Acceptance Testing

- **STEP 1**: Obtain User Requirement Prompts – Preface the prompt by describing the user role such as "As a Python Programmer analyzing clinical data" or "As a Biostatistician skilled in R". Follow the prompt describing the need for capturing user requirements such as "identify the top 10 user requirements for a statistical computing environment that uses Python".

  ChatGPT

  Certainly! Here are the top 10 user requirements for a statistical computing environment utilizing Python for a Python Programmer analyzing clinical data:

  - **Data Import and Preprocessing**: Efficient tools to import diverse clinical datasets (CSV, Excel, etc.) and preprocess them for analysis, including data cleaning and transformation.
  - **Statistical Analysis Libraries**: Access to comprehensive statistical libraries (e.g., Pandas, NumPy, SciPy) to perform various analyses like regression, hypothesis testing, and survival analysis.
  - **Visualization Capabilities**: Robust visualization modules (Matplotlib, Seaborn) to generate clear and insightful graphs and plots for data exploration and presentation.
  - **Machine Learning Integration**: Seamless integration with machine learning libraries (Scikit-learn, TensorFlow) for predictive modeling and advanced analysis, such as predictive modeling for patient outcomes.
  - **Reproducibility and Version Control**: Tools to ensure reproducibility of analyses and the ability to maintain version control for scripts and datasets to track changes.
  - **Documentation and Reporting**: Features facilitating easy documentation of analysis steps, allowing for reproducibility and creating comprehensive reports for regulatory compliance.
  - **Compatibility and Interoperability**: Compatibility with various clinical data formats and the ability to interface with other tools or systems commonly used in clinical research.
  - **Security and Data Privacy**: Robust security measures ensuring compliance with data privacy regulations (HIPAA, GDPR) when handling sensitive patient information.
  - **Performance and Scalability**: Capability to handle large-scale clinical datasets efficiently and effectively, ensuring performance and scalability.
  - **Community Support and Resources**: Access to a strong community and ample resources (forums, documentation, tutorials) for troubleshooting, learning, and staying updated with advancements in Python for clinical data analysis.

  These requirements address the essentials needed for a Python-based statistical computing environment tailored for analyzing clinical data.

To capture more specific user requirements, consider tailoring follow-up prompts aligned with your specialized role. For example, "As a biostatistician conducting analysis on oncology studies, what are the essential statistical analysis libraries crucial for Python?" This approach ensures a more targeted and detailed list, enhancing the list of identified requirements.

- **STEP 2**: For example, if you've received example test scripts from users for SciPy but lack scripts for Pandas or NumPy, you can generate sample test scripts thru AI automation, for instance: "As a Python programmer analyzing oncology clinical trial studies, could you provide an example Python script using the Pandas library?" This approach enables you to leverage AI assistance in generating specific test scripts tailored to the Pandas library, addressing the gaps in your existing set of test scripts.

ChatGPT

Absolutely! Pandas is a powerful library for data manipulation and analysis in Python, often used in data science and analytics tasks. Here's a simple example demonstrating how you might use Pandas to perform basic analysis on oncology clinical trial data:

Let's assume you have a CSV file named clinical_trials.csv containing oncology clinical trial data with columns like TrialID, PatientID, Treatment, Outcome, etc. Here's an example script:

python Copy code

```python
import pandas as pd

# Load the clinical trial data from a CSV file
clinical_trials_data = pd.read_csv('clinical_trials.csv')

# Display basic information about the dataset
print("Summary of the clinical trials data:")
print(clinical_trials_data.info())

# Show the first few rows of the dataset
print("\nFirst few rows of the dataset:")
print(clinical_trials_data.head())

# Basic statistics about numerical columns
print("\nSummary statistics:")
print(clinical_trials_data.describe())
```

This script does the following:

Imports Pandas: import pandas as pd.
Loads Data: Reads the clinical trial data from a CSV file into a Pandas DataFrame using pd.read_csv().
Basic Data Summary: Prints the data summary and the first few rows of the dataset using info() and head() functions.
Summary Statistics: Provides basic statistics (mean, standard deviation, etc.) of numerical columns using describe().
Treatment Counts: Counts the occurrences of different treatments using value_counts().
Filtering Data: Filters the data to show only patients with positive outcomes.
Replace 'clinical_trials.csv' with the correct file path and column names according to your dataset.

This is just a starting point; Pandas offers numerous functions for data manipulation, cleaning, analysis, and visualization. You can perform various operations, such as data aggregation, grouping, merging, and more, depending on the specific analysis or insights you aim to derive from the clinical trial data.

You can copy and edit the generated example scripts to significantly speed up the process of generating new scripts compared to starting from scratch. This approach allows for leveraging existing code structures and functionalities, enabling quicker adaptation and customization to suit test scripting specific to your requirements.

- **STEP 3:** Automating the execution of multiple test scripts in their correct order and capturing related outputs can streamline the process. You can achieve this with a prompt like: "Please write a Python script that runs the following programs—v_test1.py, v_test2.py, v_test3.py—and captures their respective outputs." This request prompts the creation of a script to automate the sequential execution of specified programs while collecting their outputs, simplifying the testing process.

ChatGPT

Certainly! Below is a Python script that runs three programs (`v_test1.py`, `v_test2.py`, and `v_test3.py`) sequentially and captures their output:

python Copy code

```python
import subprocess

# Define the programs to be run
programs = ['v_test1.py', 'v_test2.py', 'v_test3.py']

# Loop through each program and capture its output
for program in programs:
    print(f"Running {program}...")
    try:
        # Run the program using subprocess
        output = subprocess.check_output(['python', program]
        print(f"Output of {program}:")
        print(output)
    except subprocess.CalledProcessError as e:
        # If there's an error running the program, print the
        print(f"Error running {program}: {e.output}")
    print("=" * 50)
```

Replace `'v_test1.py'`, `'v_test2.py'`, and `'v_test3.py'` with the actual filenames of your programs. This script will execute each program sequentially and capture their respective outputs, displaying them in the console. Adjustments can be made to handle output formatting or error handling based on specific requirements.

- **STEP 4:** While human analysis is crucial for drawing conclusions in a test result summary, AI can be instrumental in facilitating this process. Utilizing AI to summarize results allows for the inclusion of AI-generated summaries at the initial sections of the test summary report. For instance, a prompt like: "Could you generate a summary of the test results based on the CSV list provided: [paste CSV text]?" This approach harnesses AI capabilities to generate a concise overview of the results, aiding in initiating the test summary report.

PERFORMANCE QUALIFICATION TESTING

Performance qualification testing shares similarities with user acceptance testing as both assess the system's functionality based on user requirements. However, the primary distinction lies in their operation. Performance testing assesses how the system handles under stress. This stress can stem from processing large and data or many users at once, pushing the system to its limits. This section will outline the conventional steps involved in performance qualification testing and explore an AI-enhanced process that mirrors this approach.

Traditional Performance Qualification Testing

- **STEP 1:** As per the business guidelines, record the maximum number of users expected to operate on the system concurrently. Additionally, outline the performance criteria defining an acceptable decrease in system speed when all users are simultaneously using the system, compared to the performance when only one user is active. For instance, the system's performance will meet the acceptable threshold if it operates at half the speed with all users working on the server at the same time compared to when only a single user is using it.

- **STEP 2:** Identify a series of test scripts designed to execute system-intensive tasks, like conducting multivariate regression analysis on large datasets. These scripts can be sourced from previous user acceptance testing sessions.

- **STEP 3:** Execute the test scripts while incorporating a method to measure the duration of the process. For instance, create a script that records the system time at the beginning of the script's execution and again once the test script finishes. The time difference between these two instances can be recorded as the duration or elapsed time of the process.

- **STEP 4:** Copy the previously mentioned scripts into separate folders, each representing distinct user sessions. This setup enables multiple scripts to run simultaneously, simulating multiple users engaging with the system to evaluate its performance under stress. For instance, if the system is designated for a smaller server, create five separate folders to represent five users. For larger servers, scale this approach by replicating folders to simulate 10, 25, 50, or even 100 users. Choose the appropriate number based on your team size and server capacity.

- **STEP 5:** Create a script to automate the simultaneous execution of the scripts located in their individual folders. Once these scripts finish running, implement a subsequent process to capture the duration of each session's execution.

- **STEP 6:** For smaller servers catering to five users, manually record the elapsed time for each process and compute the average time compared to a single process, as recorded in step 2. Determine the difference between the duration it takes for a single user versus when five users. This comparison must align with the criteria outlined in step 1 for the performance test to be deemed successful.

  For larger servers accommodating more users, create a script designed to parse each elapsed time automatically. Use this script to calculate the average elapsed time, reflecting the collective execution time of multiple users. This approach allows for a more automated assessment of performance on servers large number of users.

AI Enhanced Performance Qualification Testing

- **STEP 1**: In addition to following business guidelines on testing performance of a system, you can gain insights on this process by employing the following prompt: “How would you recommend testing the performance of a server used to run Python scripts upon clinical trials data for five users.”

  ChatGPT

  To test the performance of a server used for running Python scripts on clinical trials data for multiple users, especially five users in this case, here's a suggested approach:

  - **Define Test Scenarios**: Identify key scenarios that represent typical use cases for these Python scripts on clinical trials data. These scenarios should cover various types of operations or analyses that the users commonly perform.
  - **Create Test Data**: Generate or acquire a representative dataset similar in size and complexity to the actual clinical trials data. This data will be used during testing to ensure the simulation closely mimics real-world usage.
  - **Script Execution Management**: Copy the scripts into individual folders, each representing a separate user session. Ensure these scripts can be run independently and simultaneously.

- **Automate Execution**: Develop a script or use testing tools to automate the simultaneous execution of the Python scripts for the five user sessions. This will simulate concurrent users interacting with the system.
- **Monitor Resource Usage**: During the script execution, closely monitor the server's resource usage (CPU, memory, disk I/O) to identify any potential bottlenecks or limitations. Tools like top (for Linux) or Task Manager (for Windows) can be used for this purpose.
- **Measure Execution Time**: Capture and record the start and end times for each script execution for the five users. Calculate the elapsed time for each session.
- **Analyze Performance**: Compare the performance metrics (elapsed time, CPU usage, memory consumption) against predefined success criteria. Evaluate if the server meets the defined criteria in terms of acceptable speed and resource utilization when five users are simultaneously running the Python scripts.
- **Iterate and Optimize**: If the server doesn't meet the defined criteria, analyze the bottlenecks identified during testing. Optimize the scripts, consider parallel processing, or adjust server resources (if feasible) to enhance performance.
- **Documentation and Reporting**: Document the testing process, results, any optimizations made, and recommendations for improvement. Provide a comprehensive report detailing the server's performance under the simulated load.

By following these steps, you can assess how well the server performs when handling multiple users running Python scripts on clinical trials data simultaneously. Adjustments and optimizations based on the test results can help improve the server's performance and scalability for real-world usage.

- **STEP 2**: In addition to existing test scripts, you can have additional scripts created by the following prompt "Generate a Python program to test the performance on large data utilizing the CPU processing and memory of the server."

ChatGPT

Certainly! Here's an example Python script that generates a large dataset, performs some CPU-intensive operations, and measures the elapsed time:

python Copy code

```python
import numpy as np
import time

# Define the size of the dataset (adjust according to your server's cap
data_size = 1000000  # For example, 1 million data points

# Generate a large dataset (random numbers for illustration purposes)
large_data = np.random.rand(data_size)

# Function to perform CPU-intensive operations on the data
def perform_operations(data):
    # Example: Squaring each element in the dataset
    squared_data = np.square(data)
    # Add more CPU-intensive operations here as needed

    # Simulate memory usage by storing the result in a variable
    result = squared_data  # In a real scenario, this might involve mor

    return result

# Measure the time taken for the CPU-intensive operations
start_time = time.time()
result_data = perform_operations(e_data)
end_time = time.time()
```

This script utilizes the NumPy library to generate a large dataset of random numbers and performs CPU-intensive operations (squaring each element in this example). You can modify the operations within the perform_operations() function to simulate different types of CPU-intensive tasks.

Please adjust the data_size variable according to the capacity of your server and monitor the script's execution, as large data sizes or highly intensive operations might consume substantial memory and CPU

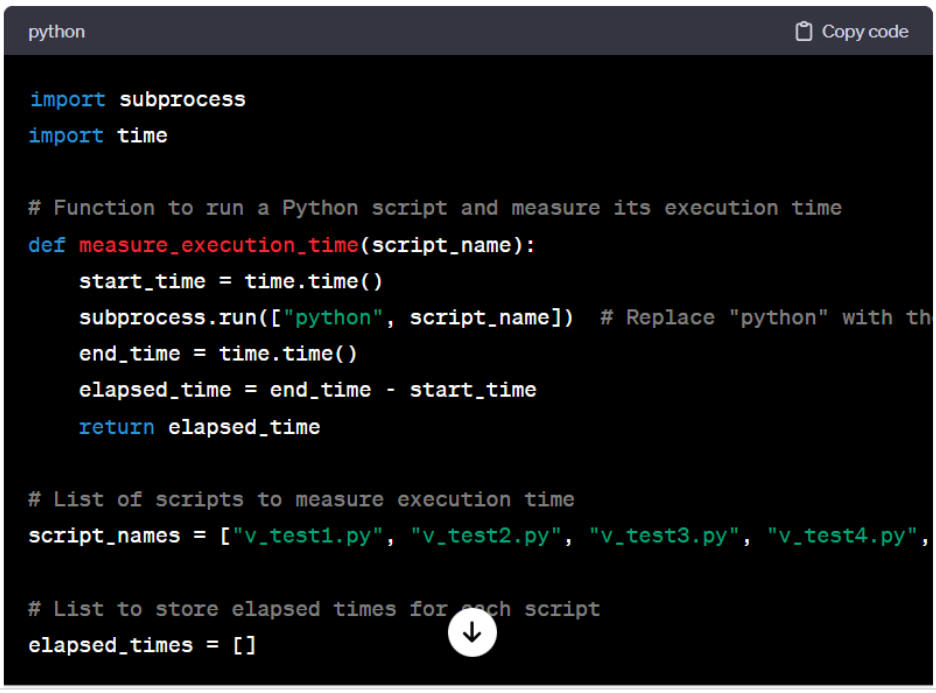
resources. This script aims to stress the CPU and memory, providing an example of performance testing with Python.

While the script might need adjustments to align precisely with your requirements, it serves as a rapid prototype compared to creating programs from scratch.

- **STEP 3:** You can also automate the process of running the script by the following prompt: “Write a Python script to capture the elapsed time of how long it takes to run scripts: v_test1.py, v_test2.py, "v_test3.py, v_test4.py and v_test5.py. Then calculate the average time to took to run these five programs.”

ChatGPT

Certainly! Here's an example of a Python script that measures the elapsed time for running five Python scripts (**v_test1.py** to **v_test5.py**) and calculates the average time taken:



Replace "v_test1.py", "v_test2.py", "v_test3.py", "v_test4.py", and "v_test5.py" with the actual filenames of your Python scripts.

This script measures the elapsed time for running each script, stores the elapsed times in a list, and then calculates the average time taken for all five scripts. Adjust it as needed to match your specific environment or requirements.

The traditional approach to script development involves step-by-step iteration, where each logical step is built incrementally. AI, however, can combine multiple logical steps in its initial draft, as seen in this example where the program automates script execution and average calculation within a single prompt.

Even with this efficiency, debugging and updates still require an iterative manual process. By initially defining all the functions of the script, you can get closer to the end goal faster.

CONCLUSION

In the traditional validation of systems like SAS, creating scripts to perform the test and to process the results are essential. For R and Python, similar methods apply, but there's a significant difference due to their open-source nature. The abundance of scripts available on platforms like GitHub, processed by large language models, makes AI a potent tool in generating a diverse set of functional scripts for various validation needs.

AI can efficiently craft sample scripts for specific user requirements and automate validation tasks, boosting the efficiencies of the validation process. However, human scrutiny remains an important step. Generated scripts often

need tweaking to align with specific business needs. Yet, this automation enriches the process, yielding more robust test cases for a thorough validation of R and Python systems, ensuring data accuracy and integrity.

The examples in this paper use an older ChatGPT 3.5 version. Newer iterations are expected to bring improved accuracy, further boosting the effectiveness of this technique. The ongoing development in prompt engineering will advance, enhancing AI's role in system validation and promising increased efficiencies in the near future.

CONTACT INFORMATION

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

Author Name: Sy Truong
Company: Meta-Xceed, Inc.
Address: 39899 Balentine Dr #200, Newark, CA 94560
Work Phone: (510) 979-9333
Email: sy.truong@meta-x.com
Website: http://meta-x.com