PharmaSUG 2021 - Paper AD-64

Harness analytical strengths: SAS, R, Python Web Services

Sandeep Juneja, SAS institute
Ben Bocchicchio, SAS Institute

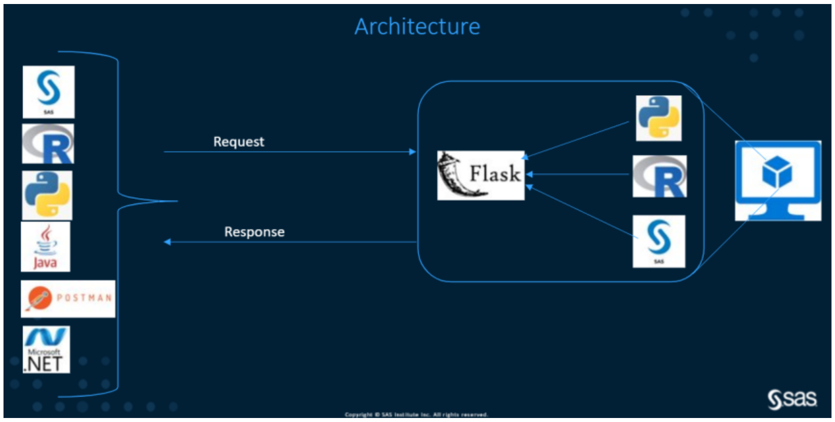
ABSTRACT

We all have our favorite system to code in and every programming language has its own inherent strengths and weakness. Programmers develop a firm understanding of the programming language they use to become proficient in problem solving. There are a few programmers that can code proficiently in more than one language, they are the lucky few. When it comes to problem solving, the question is, are you using the best coding language to solve the problem? What if there was a way to submit your “problem solving need” to another program language that can easily solve the problem? Upon submission, it returns your desired solution. This would allow you to harness capabilities of various programming languages for problem solving.

INTRODUCTION

This paper will present use cases on how to solve defined business problems using Web Services and a virtual machine. The VM will have various applications installed. The basic concept is to build a RAAS model (results as a service) to solve business needs. The primary reason for use is, the user’s individual coding system may not have the capability to solve a business problem or it may be very difficult to code in one’s native programming language.

Figure 1. Architecture and data flow using a web service to submit requests / get responses from a VM



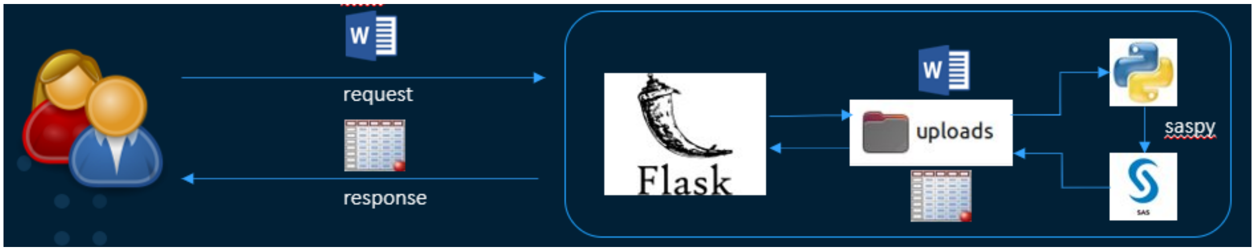
USE CASE 1

SAS has been used to generate Table outputs into Word files. The need is to save time /effort going through each printed result table and manually cross checking the data against a dataset that contains the known counts /precents for given treatment groups in the study. If a programmatic comparison was available, it would drive efficiencies and produce higher quality results.

Is it possible to convert MS Word output into a SAS dataset for validation purposes?

It is possible. It’s a two-step process: There is a package in Python named DOCx that will convert a Word file into a DataFrame, and there is a SAS python package named SASPY that initializes SAS session and helps convert a DataFrame into a SAS Dataset.

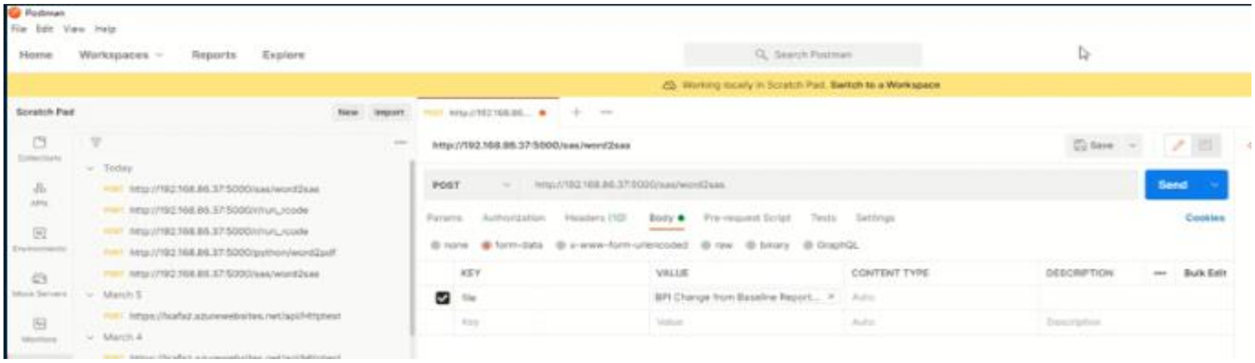
Figure 2. Illustrates the data flow of the events.



POSTING THE REQUEST

How is the request file initially sent? There are several ways to send a request file to the web service. It can be done POSTMAN, a cURL command, or through SAS code itself.

Error! Reference source not found. Figure 3. Displays an example of Postman call to the virtual machine.


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New Import
Today
March 5
March 4
Save
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Bulk Edit

http://192.168.86.37:5000/sas/word2sas

POST http://192.168.86.37:5000/sas/word2sas

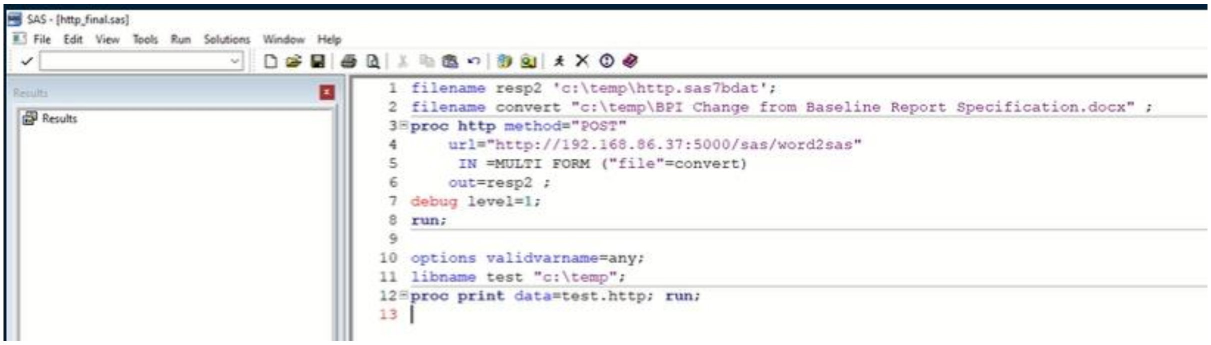
POST http://192.168.86.37:5000/sas/word2sas
POST http://192.168.86.37:5000/run_rcode
POST http://192.168.86.37:5000/run_rcode
POST http://192.168.86.37:5000/python/word2pdf
POST http://192.168.86.37:5000/sas/word2sas
POST https://hcafad.azurewebsites.net/api/HttpTest

	KEY	VALUE	CONTENT TYPE	DESCRIPTION
☑	file	BPI Change from Baseline Report...	Auto	
	Key	Value	Auto	Description

cURL command example:

curl --location --request POST 'http://192.168.86.36:5000/sas/word2sas' --form 'file=@"C:/temp/BPI Change from Baseline Report Specification.docx"' -o "C:/temp/output.sas7bdat" -v

Figure 4. Example of SAS program that will 'post' the Word file to the virtual machine.



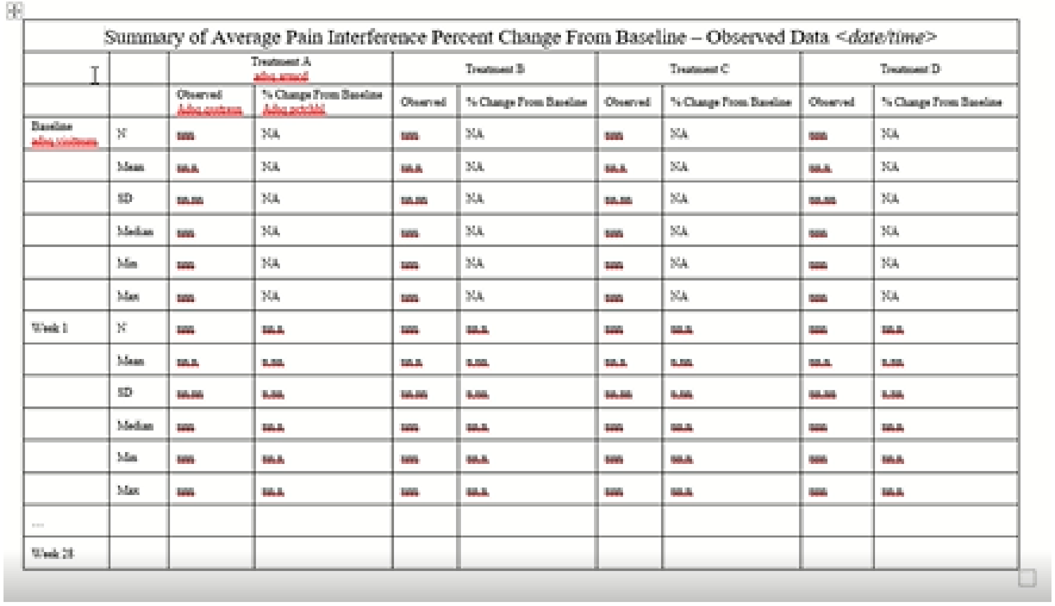
```
filename resp2 'c:\temp\http.sas7bdat';
filename convert "c:\temp\BPI Change from Baseline Report Specification.docx" ;
proc http method="POST"
    url="http://192.168.86.37:5000/sas/word2sas"
     IN =MULTI FORM ("file"=convert)
    out=resp2 ;
debug level=1;
run;

options validvarname=any;
libname test "c:\temp";
proc print data=test.http; run;
```

For the SAS code:

- The resp2 is the location/name of the returning SAS dataset
- The convert is the location/name of the Word file to convert
- The URL contains both the VM IP and the endpoint
- The MULTI / FORM –instructs SAS to use binary to stream the data

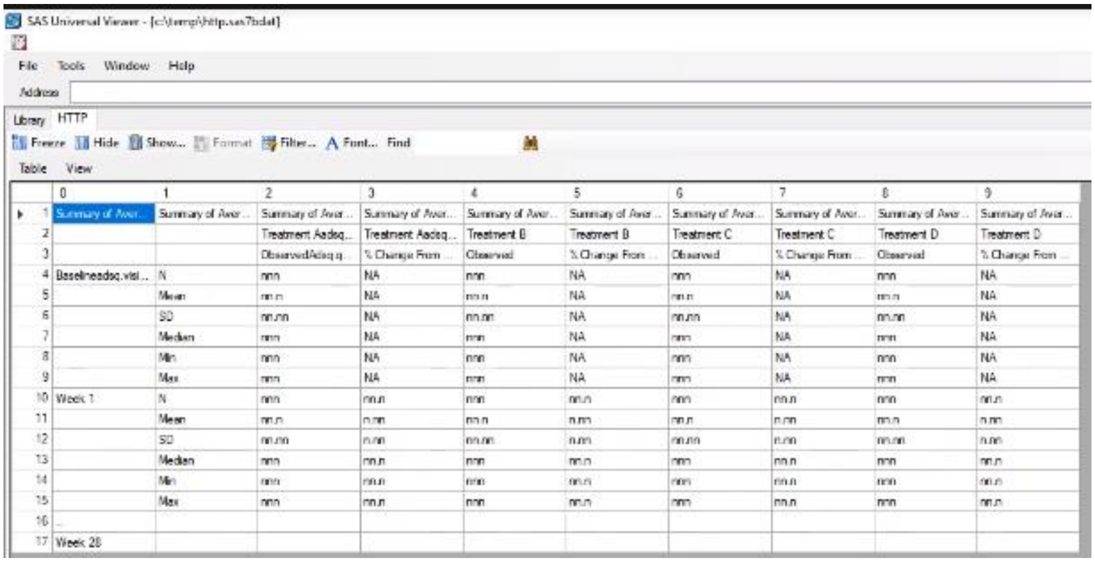
Figure 5. SAS output table in MS format



Summary of Average Pain Interference Percent Change From Baseline – Observed Data <*date/time*>									
		Treatment A adsq.armcd		Treatment B		Treatment C		Treatment D	
		Observed Adsq.cnobsen	% Change From Baseline Adsq.pctchbl	Observed	% Change From Baseline	Observed	% Change From Baseline	Observed	% Change From Baseline
Baseline adsq.visitnum	N	nnn	NA	nnn	NA	nnn	NA	nnn	NA
	Mean	nn.n	NA	nn.n	NA	nn.n	NA	nn.n	NA
	SD	nn.nn	NA	nn.nn	NA	nn.nn	NA	nn.nn	NA
	Median	nnn	NA	nnn	NA	nnn	NA	nnn	NA
	Min	nnn	NA	nnn	NA	nnn	NA	nnn	NA
	Max	nnn	NA	nnn	NA	nnn	NA	nnn	NA
Week 1	N	nnn	nn.n	nnn	nn.n	nnn	nn.n	nnn	nn.n
	Mean	nn.n	n.nn	nn.n	n.nn	nn.n	n.nn	nn.n	n.nn
	SD	nn.nn	n.nn	nn.nn	n.nn	nn.nn	n.nn	nn.nn	n.nn
	Median	nnn	nn.n	nnn	nn.n	nnn	nn.n	nnn	nn.n
	Min	nnn	nn.n	nnn	nn.n	nnn	nn.n	nnn	nn.n
	Max	nnn	nn.n	nnn	nn.n	nnn	nn.n	nnn	nn.n
...									
Week 28									

After the word file has been posted, the conversion process starts and returns the response dataset.

Figure 6. SAS dataset after the conversion process has taken place



SAS Universal Viewer - [c:\temp\http.sas7bdat]

	0	1	2	3	4	5	6	7	8	9
1	Summary of Aver...	Summary of Aver...	Summary of Aver...	Summary of Aver...	Summary of Aver...	Summary of Aver...	Summary of Aver...	Summary of Aver...	Summary of Aver...	Summary of Aver...
2			Treatment Aadsq...	Treatment Aadsq...	Treatment B	Treatment B	Treatment C	Treatment C	Treatment D	Treatment D
3			ObservedAdsq.q...	% Change From ...	Observed	% Change From ...	Observed	% Change From ...	Observed	% Change From ...
4	Baselineadsq.visi...	N	nnn	NA	nnn	NA	nnn	NA	nnn	NA
5		Mean	nn.n	NA	nn.n	NA	nn.n	NA	nn.n	NA
6		SD	nn.nn	NA	nn.nn	NA	nn.nn	NA	nn.nn	NA
7		Median	nnn	NA	nnn	NA	nnn	NA	nnn	NA
8		Min	nnn	NA	nnn	NA	nnn	NA	nnn	NA
9		Max	nnn	NA	nnn	NA	nnn	NA	nnn	NA
10	Week 1	N	nnn	nn.n	nnn	nn.n	nnn	nn.n	nnn	nn.n
11		Mean	nn.n	n.nn	nn.n	n.nn	nn.n	n.nn	nn.n	n.nn
12		SD	nn.nn	n.nn	nn.nn	n.nn	nn.nn	n.nn	nn.nn	n.nn
13		Median	nnn	nn.n	nnn	nn.n	nnn	nn.n	nnn	nn.n
14		Min	nnn	nn.n	nnn	nn.n	nnn	nn.n	nnn	nn.n
15		Max	nnn	nn.n	nnn	nn.n	nnn	nn.n	nnn	nn.n
16	...									
17	Week 28									

The resulting data set can now be manipulated through regular SAS code and compared against a dataset contain the ‘true’ patient treatment counts. Hopefully, the resulting comparison will confirm the different treatment populations have the correct number (N) of patients represented in the WORD output table. By extension, his proves the SAS coder used correct source data / SAS macros to generate the output tables. Various other comparisons can also be made using this same process. Ultimately, comparisons such as these can be conducted across the entire submission of Tables and Listings to prove consistency of the reports before submission to a regulatory agency.

Other use cases:

- While working in cloud environment, without access to MS Office, how can one convert a MS Word file into a PDF file?
- While working in cloud environment, without access to R, how can one execute an R program and capture the results?

CAVEATS

- Need to be able to setup a Virtual Machine with the required software(s) and programming language(s)
- Need to design/configure the code that gets executed by the designated software / programming language
- Need to be able to securely exchange data to and from the VM
- System need to perform at scale (my need to convert 100s of word file to datasets)

CONCLUSION

In simplest terms, the concept covered in this presentation is like a SAS macro. A piece of reusable code used to solve a programming need. The setup and configuration of a VM with web service connection, along with various software products installed, is more difficult to engineer, however the desired output is the same. The need is to build a reusable, qualified process, using the best alternate technologies, to solve a specific business problem. This approach reflects a RAAS model (results as a service), where the service is provided through the web service calling the software best suited to solve the business issue.

ACKNOWLEDGMENTS

Sandeep Juneja has built a prototype of this web service application. Without him, this would all be theory. For details on how the VM is constructed, how various software packages are installed, developing the 'business case' code, using secure communication between systems and performance considerations see: https://github.com/srjuneja/Phuse2021/blob/main/2021_PhUSE_US_TT11_configurations.pdf

.

CONTACT INFORMATION

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

Ben Bocchicchio
SAS Institute Inc.
SAS Campus Drive, Building Q
Cary, NC 27513
Ben.Bocchicchio@sas.com