

Building Hybrid Pipelines in SAS Studio: A Step-by-step Guide Using Synthetic Data from The Department of Veteran's Affairs

Last update: January 10, 2022

CONTENTS

Use Case to Cluster Healthcare Facilities	2
Intro to the Analytics Lifecycle	3
What is SAS Viya?	5
Goals of Demo	5
Accessing The Environment.....	6
Step by Step Guide	6
Step 1: Getting started with Data.....	6
Summary of Data Management phase.....	7
Step 2: Explore and Discover with SAS and Python.....	8
Exploration phase	8
Step 3: Explore and Discover with SAS, Python, and R.....	10
Exploration phase	10
Step 4: Explore and Discover with SAS.....	11
Exploration phase	12
Step 5: Deploy To CAS	13
Exploration phase	13
Conclusions.....	14

USE CASE TO CLUSTER HEALTHCARE FACILITIES

This guide is meant to jump start one's familiarity with SAS Viya, Python, and R using synthetic data from the Department of Veteran's Affairs. SAS PROC Python enables end users to use several Python package that produce dynamic visualization as well as to run R packages that help users gain insights into their data. In this guide, we provide step-by-step guidance to help users new to SAS Viya and open source integration examine healthcare facility data and identify different clusters using machine learning. More details on SAS Viya and open source integration are available in the following paper:

https://communities.sas.com/kntur85557/attachments/kntur85557/proceedings-2021/60/1/Andrews_1192.pdf

This guide is meant to help users understand healthcare facility data. Although the use case is focused primarily on healthcare facilities, they can be replaced with other variables related to quality measure, utilization, costs, etc. Likewise, the observations in the dataset are segmented or clustered using treatment variables, but this can be substituted with other variables related to claims, clinical data, etc.

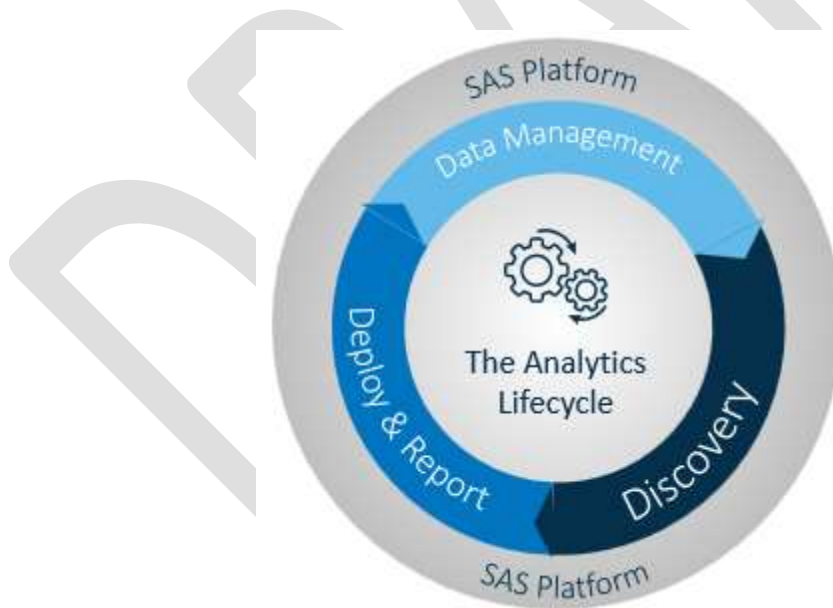
For the purposes of this guide, the healthcare facility data consists of:

- ArealD. The geographic location of a facility
- TreatmentID. The treatment given to patients at a facility. The variable to denote treatment ID for study. Typically around 8 IDs.
- Score_groupA. The score given to group A. Deidentified percentage float data (scalar - between 0-100) for treatment group A.
- Score_groupB. The score given to group B. Deidentified percentage float data (scalar - between 0-100) for treatment group B.
- Score_all. Population average score for all groups.
- diff_all_groupA. Difference between all groups and group A. Population average score minus the score of treatment group A.
- diff_all_groupB. Difference between all groups and group B. Population average score minus the score of treatment group B.
- diff_groupA_groupB. Difference between group A and group B.

Lastly, we used SAS Viya to produce these visualizations in a familiar and easy to use fashion.

INTRO TO THE ANALYTICS LIFECYCLE

Whatever data is used, an effective methodology to begin to transform data into actionable insights is the analytics lifecycle:



This tutorial will guide readers on this analytics lifecycle journey using Viya 2021:



Viya 2020 is the next generation platform for SAS with the following capabilities to better handle machine learning workloads:

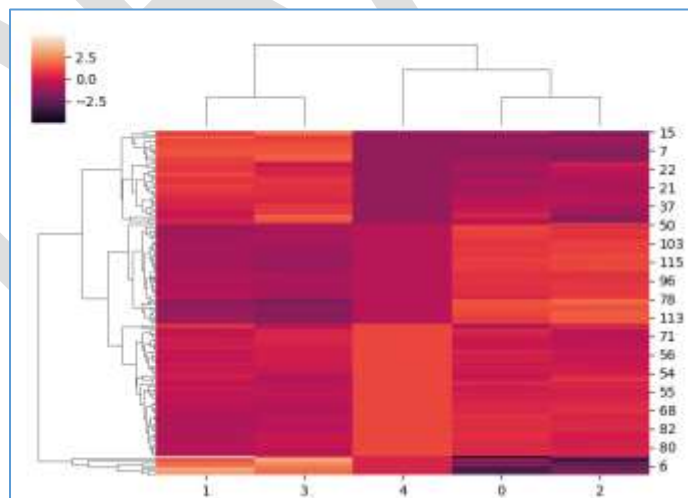
- Integration with open-source
- Cloud-native
- Containerization with Kubernetes
- CI/CD

The code and documentation is available at:

<https://github.com/ManuelFigallo/sasgovernment->

Click on the folder link “Step-by-Step”. We will be accessing the data using APIs and code.

The ultimate goal is to produce an exploration as follows:



Screen 1. Healthcare Facility Data with SAS PROC Python.

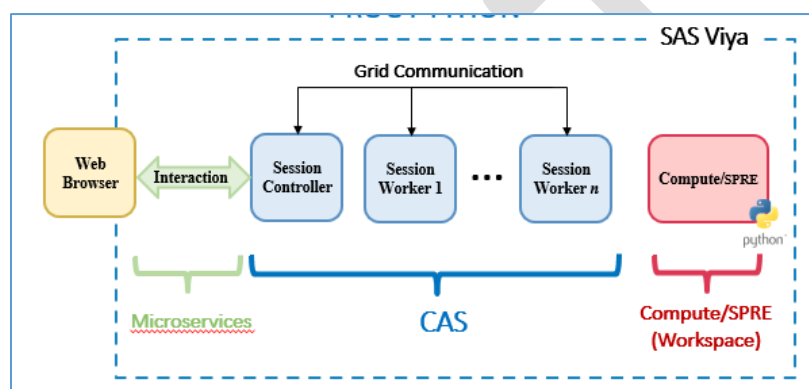
It is meant to prepare end-users of SAS Viya to proactively understand the value that SAS Viya brings and is aligned with Quadrant II thinking to use a Stephen Covey phrase.

WHAT IS SAS VIYA?

To short, SAS Viya requires a web browser to interact with its Cloud-based architecture. The architecture is comprised of:

- Microservices
- CAS (Cloud Analytic Services)
- SPRE (SAS Programming Runtime Environment).

For now, you can think of CAS as the new in-memory engine and SPRE as an instance of SAS' legacy workspace server (without a Metadata Server). Pay particular note that SAS expects that it will take time for customers to realize the full replacement value in SAS Viya, and adoption may take a while, so SAS has architected SAS Viya to include two compute engines – SAS SPRE, the “legacy” engine, and SAS CAS. These two compute engines are technologically bridged.



Screen 2. PROC Python is a Sub-process that runs in SPRE

In SAS Viya and as shown in Screen 2, a web browser interacts with SAS VIYA using microservices. They are a modular set of discrete services, such as Audit, Credentials, and Identities. Each microservice runs in its own process and communicates using HTTP. PROC Python allows you to leverage the SPRE compute server to run Python.

One best practice is to augment Python capabilities with Viya applications in the Cloud for discovery and deployment in a scalable and secure fashion. Running Python with R and SAS allows you to create hybrid pipelines that are composed of many open source technologies (such as seaborn, BERT, etc) that can be run from a .sas program.

GOALS OF DEMO

The purpose of this demo and guide is to show how you can run SAS, Python and R in “one fell swoop”. It is intended to provide basic training of SAS Viya capabilities, and it’s important to note that this guide only covers Iteration 1.

By going through this guide, users will become more familiar with these 5 requirements and understand how to fulfill them:

- Requirement 1. Run SAS Code for DM
- Requirement 2. Run Python for DM, Analytics, and Discovery with Visualizations

- Requirement 3. Run R for DM
- Requirement 4. Run SAS for Discovery
- Requirement 5. Run SAS for Reports and Deployment

The code itself is available in Github and has comments to help the user along.

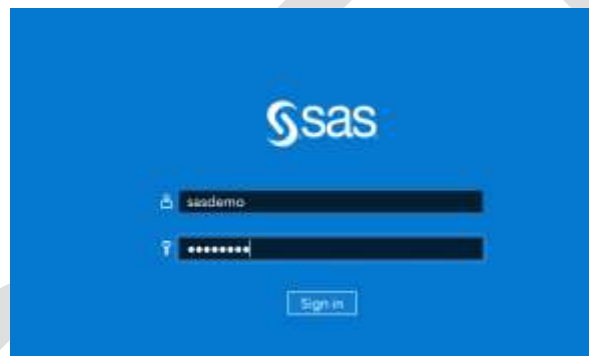
ACCESSING THE ENVIRONMENT

You can access the environment by using mstsc to logon to the jumpbox. From there go to this URL:

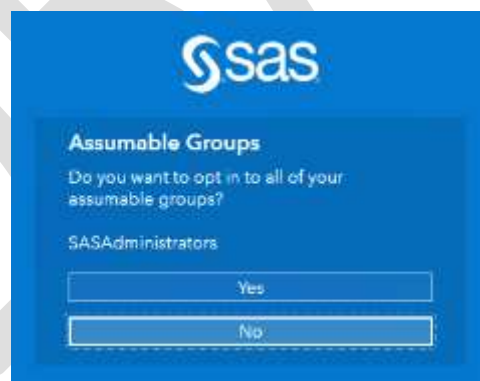
<https://server.demo.sas.com/SASStudio/>

sas

Logon with your SAS Viya credentials. For examples, username=sasdemo and password=Orion123.



When prompted with the following dialogue box, click on “yes”:



Open the program “VetAffairs_Demo_v3_2.sas”.

STEP BY STEP GUIDE

STEP 1: GETTING STARTED WITH DATA

The first step in any journey through the analytics lifecycle is data management. Our first requirement is fulfilled by using SAS to bring data from Github to our local session.

SUMMARY OF DATA MANAGEMENT PHASE

SAS Viya allows you to run data step and proc code that you may already be comfortable with. Follow these steps:

1. Bring in the data:

```

36  options mprint mlogic sasautos=(sasautos,
37  "/home/sasdemo/tmp/");
38
39  /* Fetch the file from the web site */
40  filename pydata temp;
41  ⊖ proc http
42  url="https://raw.githubusercontent.com/ManuelFigallo/sasgovernment-/main/data/python_data.csv"
43  method="GET"
44  out=pydata;
45  run;
46
47  /* Tell SAS to allow "nonstandard" names */
48  options validvarname=any;
49
50  /* import to a SAS data set */
51  ⊖ proc import
52  file=pydata
53  out=work.python_data replace
54  dbms=csv;
55  run;

```

2. Export the data so that to a csv (just in case):

```

57  /* export to a csv */
58  ⊖ proc export data=work.python_data
59  outfile="/tmp/python_data.csv"
60  dbms=csv;
61  run;

```

3. Run Data Step code:

```

63  /* use conditional processing in a data step*/
64  ⊖ data work.python_data2(drop=TreatmentID);
65  set work.python_data (keep = TreatmentID score_groupA diff_all_groupA score_groupB diff_all_groupB);
66  if TreatmentID =8 then delete;
67
68  if TreatmentID = 1 then do;
69  TreatmentDesc="TreatmentID1";
70  end;
71  else if TreatmentID = 2 then do;
72  TreatmentDesc="TreatmentID2";
73  end;
74  else if TreatmentID = 3 then do;
75  TreatmentDesc="TreatmentID3";
76  end;
77  else do;
78  TreatmentDesc="TreatmentID4";
79  end;
80  run;
81

```

4. Run PROC code:

```
82  /* use a common PROC for DM */
83  proc sql;
84      create table work.python_data3
85      as select DISTINCT score_groupA, diff_all_groupA, score_groupB, diff_all_groupB, TreatmentDesc
86      from work.python_data2
87      ;
88  quit;
```

STEP 2: EXPLORE AND DISCOVER WITH SAS AND PYTHON

It's time to write some code to explore the data!

EXPLORATION PHASE

Follow these steps in SAS Studio to run a kmeans visualization in Python using a SAS Dataset:

1. Run Python code:


```

100 %let inputtable1=work.python_data3;
101 ⊖ proc python;
102 submit;
103 # python code goes between submit and ends submit statement
104 sys.path
105
106 import seaborn as sns
107 import matplotlib.pyplot as plt
108 import pandas as pd
109 # get input data maybe Studio generated this too?
110 sasmac_inputtable1 = SAS.symget('inputtable1')
111 dfin = SAS.sd2df(sasmac_inputtable1)
112
113 df_cluster1 = dfin[dfin["TreatmentDesc"]== 'TreatmentID1']
114 df_cluster1.pop("TreatmentDesc")
115 from sklearn.cluster import KMeans
116 from sklearn.preprocessing import StandardScaler
117 scaler = StandardScaler()
118 scaler.fit(df_cluster1.fillna(0))
119 df_std = scaler.transform(df_cluster1.fillna(0))
120 kmeans = KMeans(n_clusters=4, max_iter = 300, random_state=101)
121 kmeans.fit(df_std)
122 clusters = pd.DataFrame(kmeans.predict(df_std)).set_index(df_cluster1.i
123 df_clust = pd.concat([df_cluster1, pd.DataFrame(clusters)], axis = 1)
124 scaler = StandardScaler()
125 scaler.fit(df_clust.fillna(0))
126 clust_std1 = scaler.transform(df_clust.fillna(0))
127 sns.clustermap(clust_std1, figsize=(7, 5))
128 plt.savefig('/tmp/cluster_sns1.png')
129 # done with python code
130 ends submit;
131 quit;
132

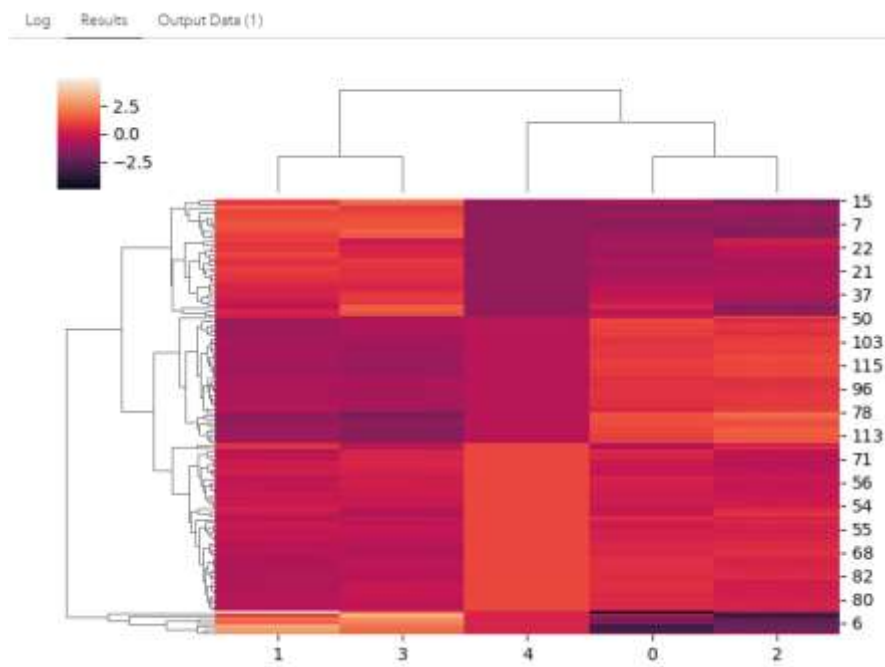
```

2. Run SAS macro to produce output in results pane:

```

133 %displaypythonresults(fn="/tmp/cluster_sns1.png")

```



Note that Seaborn is used for this output.

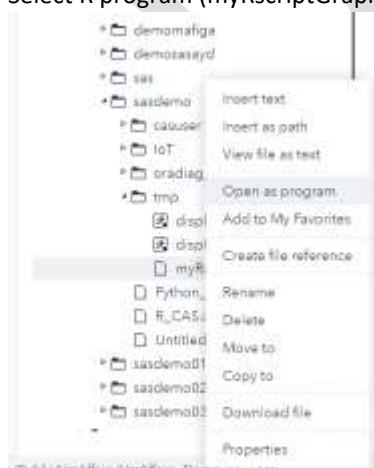
STEP 3: EXPLORE AND DISCOVER WITH SAS, PYTHON, AND R

It's time to write some code to explore the data! Run and execute R code in batch mode to produce a visual.

EXPLORATION PHASE

Follow these steps in SAS Viya:

1. Select R program (myRscriptGraph.R), and "View file as text":



2. Use Python to run R program in batch:

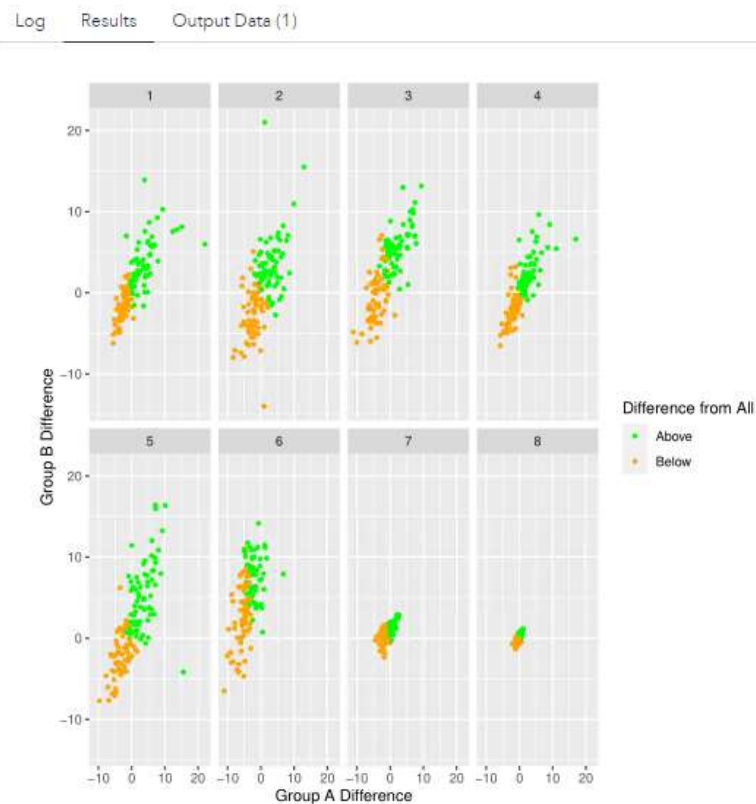
```

144 proc python;
145   submit;
146   import subprocess
147   cmd = 'cd /tmp/; /external-languages/R/lib/R/bin/R CMD BATCH /home
148   subprocess.check_output(cmd, shell=True);
149   endsubmit;
150   quit;
151
152 proc python;
153   submit;
154   import subprocess
155   cmd = 'cat /home/sasdemo/tmp/myRscriptGraph.R'
156   subprocess.check_output(cmd, shell=True);
157   endsubmit;
158   quit;
159   /**/
160 proc python;
161   submit;
162   import fits
163   pdfFile = "/tmp/Rplots.pdf"
164   doc = fits.open(pdfFile)
165   page = doc.loadPage(0) # number of page
166   pix = page.getPixmap()
167   output = "/tmp/Rplots.png"
168   pix.writePNG(output)
169   endsubmit;
170   quit;

```

3. Display ggplot visualization output:

```
172 %displayrresults(fn="/tmp/Rplots.png")
```



Modify R code as desired.

It's time to write some code to explore the data!

EXPLORATION PHASE

Follow these steps in SAS Viya:

1. Use SAS to produce an advanced table for analysis:

```

200 ② PROC TABULATE
201     DATA=WORK.PYTHON_DATA
202     ;
203     WHERE( TreatmentID = 1);
204     VAR score_groupA score_groupB;
205     CLASS better_all_area / ORDER=UNFORMATTED MISSING;
206     TABLE
207     /* Row Dimension */
208     /**/
209     score_groupA
210     score_groupB,
211     /* Column Dimension */
212     better_all_area*(
213         N
214         Mean
215         StdDev)
216     ALL={LABEL="Total (ALL)"}*(
217         N
218         Mean
219         StdDev) ;
220     ;
221     RUN;

```

Here is the output:

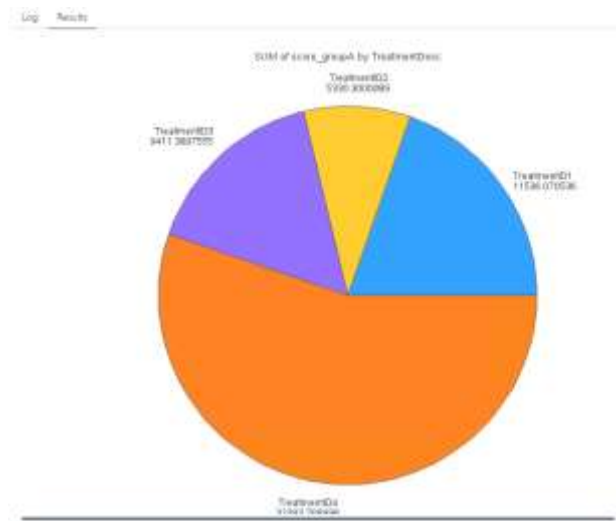
	better_all_area						Total (ALL)		
	0			1					
	N	Mean	StdDev	N	Mean	StdDev	N	Mean	StdDev
score_groupA	59	85.21	4.13	71	91.67	1.39	130	88.74	4.37
score_groupB	59	85.47	3.08	71	90.79	1.84	130	88.38	3.63

2. Use SAS to produce a visual:

```

224 ② proc gchart data=PYTHON_DATA3;
225     pie TreatmentDesc / type=sum sumvar=score_groupA;
226     run;
227     quit;
---
```

Here is the output:



Modify as needed.

STEP 5: DEPLOY TO CAS

It's time to write some code to deploy the data to CAS for dashboarding!

EXPLORATION PHASE

Follow these steps in SAS Viya:

1. Deploy to CAS:

```

239 options casport=5570;
240 cas;
241
242 cas mySession sessopts=(caslib=PUBLIC timeout=1800 locale="en_US");
243
244 cas;
245 caslib _all_ assign;
246
247 proc cas;
248 session mySession;
249 table.tableexists result=results/ caslib="PUBLIC" name="PYTHON_DATA";
250 if(results.exists) then table.dropTable / name="PYTHON_DATA";
251 run;
252
253 data PUBLIC.PYTHON_DATA;
254 set work.PYTHON_DATA;
255 run;
256
257 proc casutil;
258 promote incaslib=PUBLIC casdata="PYTHON_DATA" outcaslib=PUBLIC;
259 run;
    
```

The data is now available in CAS for Dashboarding.

CONCLUSIONS

Python users can gain access to the SPRE engine through SAS PROC Python. Data manipulation and processing is made easy in PROC PYTHON by mimicking much of the Pandas API. Handling data with PROC PYTHON will feel familiar to Pandas users with methods like head, tail, summary, and functionality of both loc and iloc available.

Python is versatile and has a vast user base, especially in the data science community. In particular, it is an excellent tool to use in traversing the entire analytics lifecycle beginning with data management, then data exploration, model building and finally model deployment—as discussed above. Many Python packages are specifically made for parts of the machine learning pipeline. Pandas and NumPy are great for cleaning data and data transformations.