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

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.

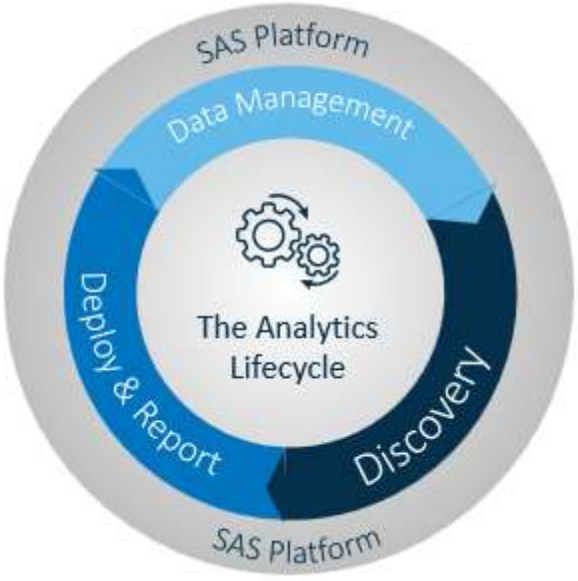
For the purposes of this guide, the healthcare facility data consists of:

- AreaID. The geographic location of a facility
- TreatmentID. The treatment given to patients at a facility. The variable to denote treament 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:

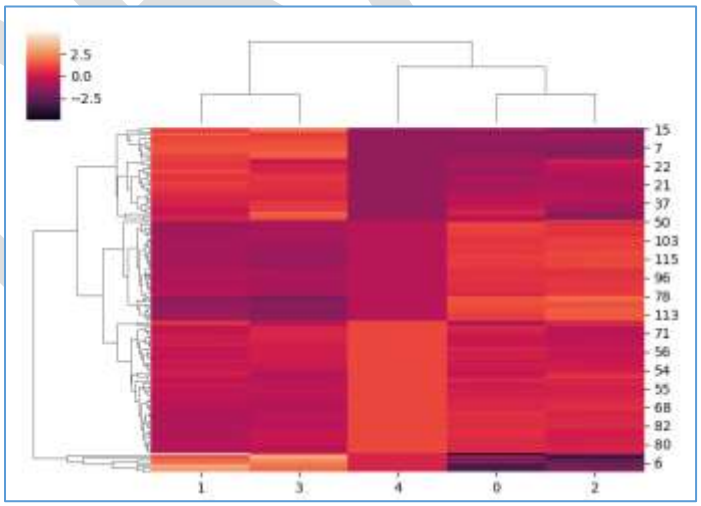
- Integration with open-source
- Cloud-native
- Containerization with Kubernetes
- CI/CD

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.

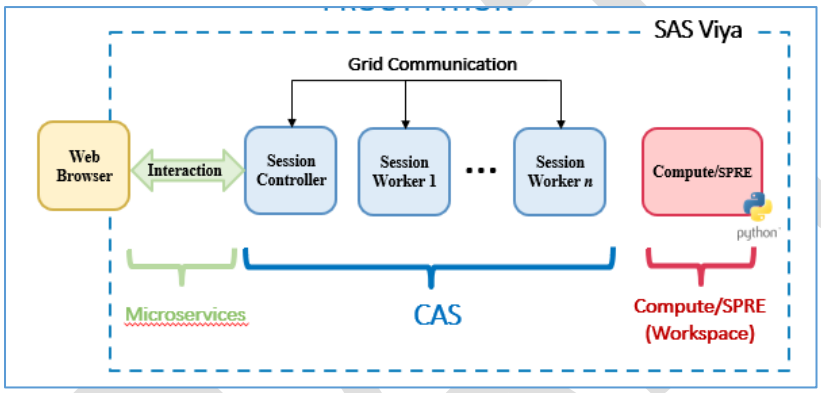
It is meant to prepare end-users of SAS Viya to proactively understand the value that SAS Viya brings and is aligned with Quadrant II thinking to use a Stephen Covey phrase.

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:

```
options mprint mlogic sasautos=(sasautos,
"/home/sasdemo/tmp/");

/* Fetch the file from the web site */
filename pydata temp;
proc http
 url="https://raw.githubusercontent.com/ManuelFigallo/sasgovernment-/main/data/python_data.csv"
 method="GET"
 out=pydata;
run;

/* Tell SAS to allow "nonstandard" names */
options validvarname=any;

/* import to a SAS data set */
proc import
  file=pydata
  out=work.python_data replace
  dbms=csv;
run;
```

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

```
/* export to a csv */
proc export data=work.python_data
    outfile="/tmp/python_data.csv"
    dbms=csv;
run;
```

3. Run Data Step code:

```
/* use conditional processing in a data step*/
data work.python_data2(drop=TreatmentID);
    set work.python_data (keep = TreatmentID score_groupA diff_all_groupA score_groupB diff_all_groupB);
    if TreatmentID =8 then delete;

    if  TreatmentID = 1 then do;
        TreatmentDesc="TreatmentID1";
    end;
    else if  TreatmentID = 2 then do;
        TreatmentDesc="TreatmentID2";
    end;
    else if  TreatmentID = 3 then do;
        TreatmentDesc="TreatmentID3";
    end;
    else do;
        TreatmentDesc="TreatmentID4";
    end;
run;
```

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

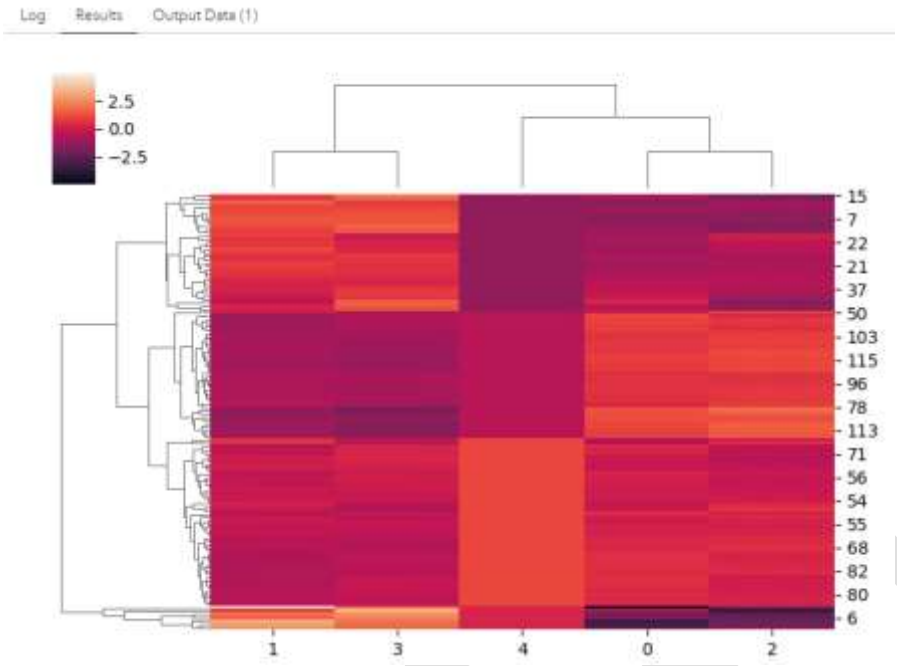
Follow these steps in SAS Studio to run a kmeans visualization in Python using a SAS Dataset:

1. Run Python code:

```
%let inputtable1=work.python_data3;
proc python;
submit;
# python code goes between submit and endsubmit statement
sys.path

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
# get input data maybe Studio generated this too?
sasmac_inputtable1 = SAS.symget('inputtable1')
dfin = SAS.sd2df(sasmac_inputtable1)

df_cluster1 = dfin[dfin["TreatmentDesc"]== 'TreatmentID1']
df_cluster1.pop("TreatmentDesc")
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df_cluster1.fillna(0))
df_std = scaler.transform(df_cluster1.fillna(0))
kmeans = KMeans(n_clusters=4, max_iter = 300, random_state=101)
kmeans.fit(df_std)
clusters = pd.DataFrame(kmeans.predict(df_std)).set_index(df_cluster1.i
df_clust = pd.concat([df_cluster1, pd.DataFrame(clusters)], axis = 1)
scaler = StandardScaler()
scaler.fit(df_clust.fillna(0))
clust_std1 = scaler.transform(df_clust.fillna(0))
sns.clustermap(clust_std1, figsize=(7, 5))
plt.savefig('/tmp/cluster_sns1.png')
# done with python code
endsubmit;
quit;

```

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

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



Note that Seaborn is used for this output.

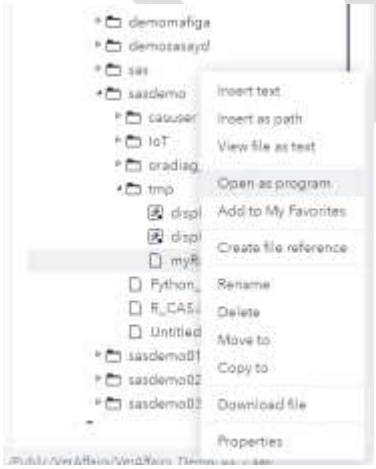
STEP 3: EXPLORE AND DISCOVER WITH SAS, PYTHON, AND R

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":



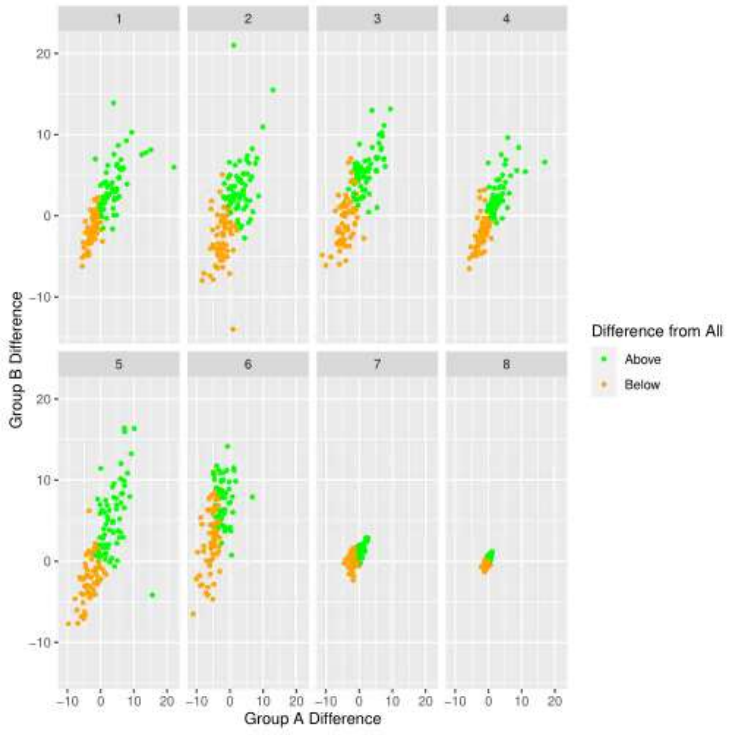
2. Use Python to run R program in batch:

```
144  proc python;
145  submit;
146  import subprocess
147  cmd = 'cd /tmp/; /external-languages/R/lib/R/bin/R CMD BATCH /hom
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 fitz
163  pdffile = "/tmp/Rplots.pdf"
164  doc = fitz.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")
```

Log | Results | Output Data (1)



Modify R code as desired.

STEP 4: EXPLORE AND DISCOVER WITH SAS

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:

```
PROC TABULATE
DATA=WORK.PYTHON_DATA
;
WHERE( TreatmentID = 1);
VAR score_groupA score_groupB;
CLASS better_all_area / ORDER=UNFORMATTED MISSING;
TABLE
/* Row Dimension */
/**/
score_groupA
score_groupB,
/* Column Dimension */
better_all_area*(
  N
  Mean
  StdDev)
ALL={LABEL="Total (ALL)"}*(
  N
  Mean
  StdDev) ;
;
RUN;
```

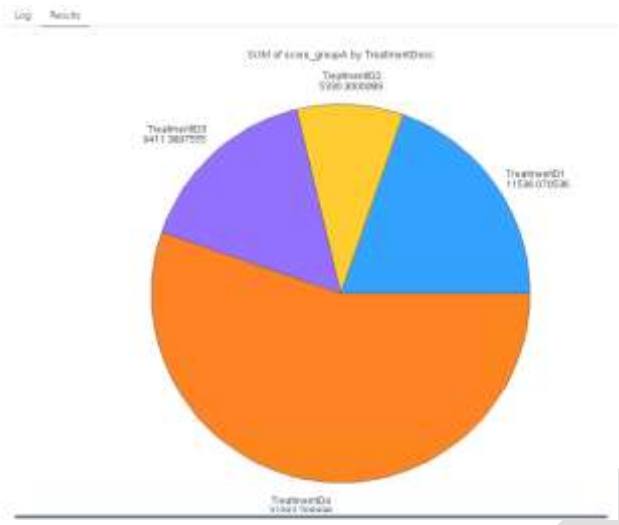
   Here is the output:

	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:

```
proc gchart data=PYTHON_DATA3;
pie TreatmentDesc / type=sum sumvar=score_groupA;
run;
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 userscan 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. 12 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.