Paper 2597-2018

SAS® and Python: The Perfect Partners in Crime

Carrie Foreman, Amadeus Software Limited

ABSTRACT

Python is often one of the first languages that any programmer will study. In 2017, Python ® was named as the world's most popular language by the IEEE Spectrum research group (Smit, 2017) and was named third in the list of popular programming languages taught within the UK (Murphy, Crick and Davenport 2017). In April 2017, SAS ® introduced the SASPy project, a package that can be installed on top of Python 3 to connect to a Python session to a SAS 9.4 or SAS Viya ® 3.1 workspace session (Hemedinger, 2017).

This SAS workspace session connectivity allows any authenticated users to utilize many of the procedures available within the SAS environment, the majority of which are also available by using the Python programming language. This enables users to have the freedom to code in a language of their choice with the view of achieving the same results.

SASPy also includes a method which can convert any compatible Python coding which is submitted, into the SAS coding equivalent. This allows both SAS coders and Python coders to work side by side, in addition to aiding those Python coders in their learning of the SAS programming language.

INTRODUCTION

As a SAS programmer, it is easy to assume that everyone has time to learn the SAS programming language. However, with budgets and time at a premium, it can be hard to set aside time to immerse yourself and learn a new programming language.

The good news is that if you are a Python programmer, making use of the best analytics products and solutions can now be much simpler. SAS have been developing modules which can be installed on top of Python version 3 which can be sourced and used within the Jupyter Notebook ®. These modules allow both SAS users and Python coders to connect to a SAS 9.4 or SAS Viya 3.1 workspace session. These new developments which are emerging are very exciting for everyone within the SAS world as this allows users to have more freedom to code in a language of their choice with a view to achieving identical results.

In this paper all of the coding will be completed within Python to demonstrate the capabilities of the software without any prior knowledge of the SAS programming language. UK crime stop and search data available freely from the UK Home Office (data.police.uk, 2017) will be accessed via an API to demonstrate the real-time connectivity. The data collected through the API will be converted into a SAS data set before being analyzed using SAS software.

To demonstrate the flexibility in the coding languages used by Jupyter ®, the teach_me_SAS method will be explored to display the comparison of the Python coding with the equivalent SAS programming code which can enable Python coders to learn aspects of the language.

THE JUPYTER NOTEBOOK

The Jupyter Notebook is an interface which integrates three main languages; Julia, Python and R into a single platform and is the tool of choice for this paper. In addition to this, the Jupyter Hub is available for enabling the Jupyter Notebook to be available across an organization.

To run the examples created for the Jupyter notebook within this paper, the IPython kernel is required to enable Python code to be run. In addition to this, several extensions are available for Jupyter to add further functionality to the system.

Four examples of extensions available to install for Jupyter specifically created for SAS include;

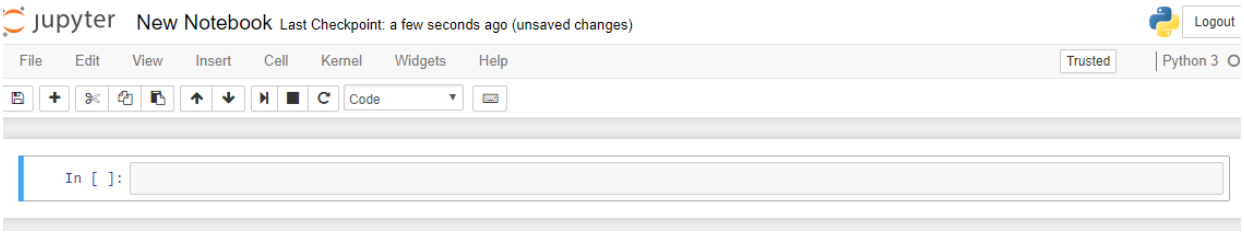
- SASPy: A module used to allow a connection to be made to SAS 9.4. This is required to enable users to create Python coding to surface SAS objects.
- SAS Kernel: A kernel which allows SAS code to be executed within the Jupyter Notebook. *
- Pipefitter: A package used to create and assess SAS Viya 3.1 and SAS 9.4 data mining models. *
- Python-SWAT: A package which allows users to connect to a SAS Viya 3.1 session and execute CAS actions using Python coding.

* Requires the SASPy module.

To run the examples within this paper only the SASPy module will be required.

CONNECTING TO A SAS SESSION

On opening the Jupyter Notebook for the first time a new notebook is displayed containing a single code cell. The toolbars at the top of the page are used to control the workings for the notebook. All code within this paper can be typed directly into a code cell and is run by selecting the run button within the toolbar menu.



Display 1. The Jupyter Notebook

Prior to connecting to a SAS Session within Jupyter Notebook the SASPy module needs to be loaded into Python. This can be completed up by typing the Python import command into a code cell. The code to complete this is as follows:

```
import saspy
```

Once this has been completed, to connect to a SAS Session, a connection statement is required. Within the connection statement a configuration name is specified. This configuration name matches a connection method which is set up within a user's sascfg.py file found within their SASPy configuration. The file contains a number of configurable options which are available for a user to adapt to customize their connection to SAS 9.4 or SAS Viya 3.1. The connection statement is formatted as below:

```
sas = saspy.SASsession(cfgname = 'iomwin')
```

In this example, the IOM method for Windows ® Operating Systems is used to connect to the local machine. IOM methods are available to connect to any Operating System. In addition to this, STDIO methods are also available for Linux ® connections to SAS workspace sessions (SAS, 2017).

On running the connection statement, this will prompt for a username and password. Once these have been entered, if a connection is made successfully a message will be displayed below the code cell:

```
Please enter the IOM user id: carrie.foreman
Please enter the password for IOM user : ........
SAS Connection established. Subprocess id is 1524
```

Output 1. A sucessful connection is made to SAS 9.4

CONNECTING TO THE DATA

The data which will be used within this example is that of the Thames Valley Police stop and search data. The data is available via API and contains information on where stop and searches took place in addition to the ethnicity and gender of those searched. To connect to the JSON API the pandas read_json function is required. This can be used to convert the API data into a pandas dataframe. To complete this task, access to the Python pandas library is required. Pandas is a library created specifically for the Python programming language to perform data analysis and manipulations. To gain access to pandas, the pandas module needs to be imported using an alias (in this example we will use pd). The alias can then be used for any calls to the pandas module:

```
import pandas as pd
```

To convert the data within the API to a pandas dataframe the following code is required:

```
df = pd.read_json('https://data.police.uk/api/stops-force?force=thames-valley&data=2017-07')
```

To view the pandas dataframe `df` can be run within a code cell:

```
df
```

	age_range	datetime	gender	involved_person	legislation	object_of_search	officer_defined_ethnicity	outcome	outcome_link_to_object	removed_clothing	latitude	longitude
0	25-34	2017-07-01 00:00:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	False	51.739670	-0.967298
1	25-34	2017-07-01 00:00:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	False	51.739670	-0.967298
2	18-24	2017-07-01 00:00:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	False	51.739670	-0.967298
3	10-17	2017-07-01 04:03:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	Black	False	NaN	False	51.507752	-0.598295
4	10-17	2017-07-01 04:03:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	Black	False	NaN	False	51.507752	-0.598295
5	over 34	2017-07-01 13:30:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	Article found - Detailed outcome unavailable	True	False	51.628712	-0.751329
6	18-24	2017-07-01 17:32:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	Article found - Detailed outcome unavailable	NaN	False	51.418490	-1.510796
7	18-24	2017-07-01 17:48:00	None	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	Suspect arrested	True	False	51.400908	-1.319032
8	18-24	2017-07-01 17:55:00	None	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	False	51.418490	-1.510796
9	over 34	2017-07-01 18:50:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	None	Suspect arrested	True	False	51.547442	-0.893619
10	10-17	2017-07-01 21:52:00	Male	True	Police and Criminal Evidence Act 1984 (section 1)	Article for use in theft	White	False	NaN	False	51.734236	-1.211889

Output 2. View the Pandas data frame within Jupyter Noteboook

PYTHON DATA MANIPULATION

To convert the pandas dataframe into a SAS data set, some data manipulation is required. This is due to the differences between the storage of some variables within pandas and how this compares with variables types and naming conventions supported by the SAS environment. To view the data types stored within the pandas library `df.info()` can be run:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 497 entries, 0 to 496
Data columns (total 16 columns):
age_range                              456 non-null object
datetime                               497 non-null datetime64[ns]
gender                                 437 non-null object
involved_person                        497 non-null bool
legislation                            325 non-null object
location                               321 non-null object
object_of_search                       497 non-null object
officer_defined_ethnicity              494 non-null object
operation                              497 non-null bool
operation_name                         0 non-null float64
outcome                                497 non-null object
outcome_linked_to_object_of_search     99 non-null float64
outcome_object                         497 non-null object
removal_of_more_than_outer_clothing    325 non-null float64
self_defined_ethnicity                 479 non-null object
type                                   497 non-null object
dtypes: bool(2), datetime64[ns](1), float64(3), object(10)
memory usage: 36.0+ KB
```

Output 3. Pandas dataframe information prior to any manipulation

DATA PREPARATION

Using `df.info()` and viewing the dataset in full, four steps have been identified with the data which need to be completed before the data can be converted:

1. The columns “outcome_linked_to_object_of_search” and “removal_of_more_than_outer_clothing” have names which are 34 and 35 characters and in length respectively. SAS 9.4 and SAS Viya 3.1 require that the names of variables within a data set are of a maximum length of 32 characters.
2. The columns “operation” and “involved_person” are of type boolean. SAS 9.4 accepts only character or numeric variables, therefore any variables of type Boolean need to be changed to either character strings or numeric values of 1 or 0 prior to conversion to a SAS data set.
3. The “outcome” column contains values which are recorded as a set of quotes “”. Within SAS this is not interpreted as a value and therefore misaligns the values for any variables to the right of this within the data set. These values need to be replaced to allow them to be identified within the data.
4. The column “location” has two outcomes; either multiple pieces of information which includes latitude and longitude coordinates, or a missing value. On reading this data into SAS without any manipulation any non-missing values are not identified, and the variable is assigned to missing. These values need to be extracted from this column before the data is converted into a SAS data set.

Each of these steps requires manipulation using Python coding prior to conversion within SAS.

Task 1: Column names

To rectify the issue of the two columns which have names that are greater than 32 characters we need to rename the columns within the pandas dataframe. The original column name will be replaced with a new shorter column name.

Original Column Name	New Column Name
outcome_linked_to_object_of_search	outcome_link_to_object
removal_of_more_than_outer_clothing	removed_clothing

Table 1. Column name changes within Pandas

We can use the dataframe rename function provided by the pandas module to complete this task. This function allows us to assign a new name for a column within a pandas dataframe:

```
df = df.rename(columns={'outcome_linked_to_object_of_search':
'outcome_link_to_object',
'removal_of_more_than_outer_clothing': 'removed_clothing'})
```

Task 2: Boolean data types

Any Boolean data types within pandas need to be converted to character or numeric data to be read into SAS. In this example character strings will be used. To complete this, we can use the dataframe replace function provided by the pandas module to complete this task. This function allows us to specify a list of values that we would like to assign to an alternative value. For the police stop and search data any values within the data that are defined as a Boolean true or false will be assigned to the text strings of “True” or “False”:

```
booleanDictionary = {True: 'True', False: 'False'}
df = df.replace(booleanDictionary)
```

Task 3: Outcome column values empty

The outcome column has values which are recorded as “”. This is an issue, as on conversion to a SAS data set, the data is misaligned causing values to be read into incorrect variables. To rectify this issue, we need to iterate using the iterrows generator through the data set to identify those records where the length of the string stored for outcome is less than one character. To complete this the numpy module can be used which allows access to mathematical functions and arrays.

To import the numpy module you can use the code below:

```
import numpy as np
```

Two additional columns are added to the pandas dataframe.

New Column	Value
tmpstring	Stores the value of the outcome column as a string.
length	Calculates the length of the value stored within the tmpstring column.
outcome	For any rows where the length column has a value less than 1, the outcome column will be assigned to the value of “False”. In any other cases, the outcome will not be changed.

Table 2. Additional columns within Pandas

The code to complete this is as follows:

```
for index, row in df.iterrows():
    df['tmp_string'] = df['outcome'].astype(str)
    df['tmp_length'] = df['tmp_string'].str.len()
    df['outcome'] = np.where(df['tmp_length'] < 1, 'False', df['outcome'])
```

Task 4: The location column

Within the API additional items of information are stored within the location column. This includes the following two additional values which we would like to include within the data:

- Latitude
- Longitude

These values can be extremely useful for analysis within SAS and can be used to plot police stop and search locations on a map. To extract this information an ‘if’ statement has been used within Python coding. This statement identifies whether a location variable is blank. If the variable is blank, then two new columns named latitude and longitude will be set to none within the data set. Where this is not the case, the values have been extracted from the string using point identifiers to extract the relevant parts.

```
for index, row in df.iterrows():
    df['tmp_location'] = df['location'].astype(str)
    if row['location'] is None:
        df['latitude'] = None
        df['longitude'] = None
    else:
        df['latitude'] = df['tmp_location'].str[14:23]
        df['longitude'] = df['tmp_location'].str[-11:-2]
```

Task 5: Convert latitude and longitude to numeric

As the latitude and longitude were extracted from a string using Python code, these are currently stored as non-null objects. To perform calculations on these, or to plot them on a map, it is required that these values are stored as numeric values. A numeric column stored within Python would be a non-null float64. To convert to this variable type, the to_numeric function is required. The coerce option has been used on the to_numeric function to allow any values which cannot be converted to be recorded as missing values. The code to complete this is as follows:

```
df['latitude'] = pd.to_numeric(df['latitude'], errors='coerce')
df['longitude'] = pd.to_numeric(df['longitude'], errors='coerce')
```

Task 6: Dropping unnecessary columns

The following columns can then be dropped from the pandas dataframe as they are not required further for manipulation within SAS.

- location
- operation
- operation_name
- outcome_object
- self_defined_ethnicity
- type
- loc_string

To remove these variables, the del operator can be used:

```
del df[`location']
del df['operation']
del df['operation_name']
del df['outcome_object']
del df['self_defined_ethnicity']
del df['type']
del df['loc_string']
```

VERIFY THE DATA TYPES ARE CORRECT

Once the pandas manipulation has been completed, `df.info()` can be run again to verify that the data types are set correctly.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 12 columns):
age_range                    471 non-null object
datetime                     499 non-null datetime64[ns]
gender                       462 non-null object
involved_person              499 non-null object
legislation                  340 non-null object
object_of_search             481 non-null object
officer_defined_ethnicity    491 non-null object
outcome                      499 non-null object
outcome_link_to_object       95 non-null object
removed_clothing             347 non-null object
latitude                     330 non-null float64
longitude                    330 non-null float64
dtypes: datetime64[ns](1), float64(2), object(9)
memory usage: 46.9+ KB
```

Output 4. Pandas dataframe information after manipulation

The data types within the dataframe are now prepared for analysis in SAS. All SAS numeric variables are stored in the dataframe as either a non-null float 64 or a non-null datetime64[ns]. In addition to this, any character columns are defined as non-null objects and are therefore suitable to convert into SAS character variables.

The data can also be viewed using the `df.head(20)` function.

```
df.head(20)
```

	age_range	datetime	gender	involved_person	legislation	object_of_search	officer_defined_ethnicity	outcome	outcome_link_to_object	removed_clothing	latitude	longitude
0	over 34	2017-06-02 13:25:00	Male	True	None	Controlled drugs	Black	Suspect arrested	True	NaN	NaN	NaN
1	over 34	2017-06-06 15:20:00	Female	True	Police and Criminal Evidence Act 1984 (section 1)	Stolen goods	White	False	NaN	False	51.612171	-0.787758
2	over 34	2017-06-25 00:48:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	False	51.722832	-1.202112
3	over 34	2017-06-23 11:00:00	Male	True	None	Stolen goods	White	False	NaN	NaN	NaN	NaN
4	over 34	2017-06-26 12:05:00	Male	True	None	Stolen goods	White	False	NaN	NaN	NaN	NaN
5	over 34	2017-06-15 09:28:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	False	51.628712	-0.751329
6	over 34	2017-06-04 13:45:00	Male	True	None	Stolen goods	White	False	NaN	NaN	NaN	NaN
7	over 34	2017-06-06 20:20:00	Male	True	None	Controlled drugs	White	False	NaN	NaN	NaN	NaN
8	over 34	2017-06-01 17:16:00	Female	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	False	51.850595	-1.091347
9	over 34	2017-06-14 15:02:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	True	51.627520	-0.749093
10	over 34	2017-06-05 18:11:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	Black	False	NaN	False	52.004930	-0.731680
11	over 34	2017-06-19 05:28:00	Male	True	None	Stolen goods	White	False	NaN	NaN	NaN	NaN
12	over 34	2017-06-03 05:02:00	Male	True	None	Stolen goods	Asian	False	NaN	NaN	NaN	NaN
13	over 34	2017-06-11 04:32:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	Suspect arrested	True	False	52.040209	-0.766140
14	over 34	2017-06-22 08:41:00	Male	True	Police and Criminal Evidence Act 1984 (section 1)	Offensive weapons	White	False	NaN	False	51.455472	-0.967433
15	over 34	2017-06-23 17:30:00	Male	True	None	Stolen goods	Black	Suspect arrested	True	NaN	NaN	NaN
16	over 34	2017-06-08 18:35:00	Female	True	None	None	White	False	NaN	False	NaN	NaN
17	over 34	2017-06-30 08:52:00	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	NaN	True	51.627520	-0.749093
18	over 34	2017-06-10 20:01:00	Male	True	None	Controlled drugs	Black	False	NaN	NaN	NaN	NaN
19	over 34	2017-06-06 15:27:00	Male	True	Police and Criminal Evidence Act 1984 (section 1)	Stolen goods	White	False	NaN	False	51.612171	-0.787758

Output 5. Viewing the Pandas data frame after manipulation

CONVERT A PANDAS DATA FRAME TO SAS DATA

Once the data has been manipulated into a form which SAS is able to convert, this can be read into a SAS data set. By default, SASPy will store a new SAS data set into the SAS WORK location.

To store a data set permanently, a library can be created to send the data set to. To create a SAS library, the sas.saslib statement can be used. This contains three options: including the name of the SAS library, the type of library and the location where the SAS data sets should be stored. In this example a library named mydata will be created using the SAS BASE engine in the location specified. To complete this the following code is required:

```
sas.saslib('mydata', 'BASE', 'C:\SAS\mydata')
```

If a library is assigned successfully the following message will be displayed:

```
6                                                    The SAS System                                15:41 Friday, March 2,
2018

39
40         libname mydata BASE  'C:\SAS\mydata'  ;
NOTE: Libref MYDATA was successfully assigned as follows:
      Engine:        BASE
      Physical Name: C:\SAS\mydata
41
42
```

Output 6. Viewing the Pandas data frame after manipulation

Once the library is available, the `sas.dataframe2sasdata` or `sas.df2sd` function can be used. This assigns a new Python variable which will point to the SAS data set. Within the function brackets the pandas dataframe is specified in addition to the library which will be used to store the SAS data set.

```
sassearchdata = sas.df2sd(df, libref='mydata')
```

To verify that the SAS data set has been created successfully, the contents function can be used on the new data set.

```
sassearchdata.contents()
```

If “The SAS System” is displayed at the top of the output, the pandas dataframe has been successfully converted to a SAS data set.

The SAS System

The CONTENTS Procedure

Data Set Name	MYDATA_DF	Observations	499
Member Type	DATA	Variables	12
Engine	BASE	Indexes	0
Created	06/03/2018 08:40:01	Observation Length	192
Last Modified	06/03/2018 08:40:01	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	2
First Data Page	1
Max Obs per Page	340
Obs in First Data Page	329
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\SAS\mydata_df.sas7bdat
Release Created	9.0401M4
Host Created	X64_10PRO
Owner Name	AMADEUS\carrie.foreman
File Size	192KB
File Size (bytes)	196608

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
1	age_range	Char	7	
2	datetime	Num	8	E8601DT26.6
3	gender	Char	6	
4	involved_person	Char	5	
11	latitude	Num	8	
5	legislation	Char	49	
12	longitude	Num	8	
6	object_of_search	Char	35	
7	officer_defined_ethnicity	Char	5	
8	outcome	Char	44	
9	outcome_link_to_object	Char	5	
10	removed_clothing	Char	5	

Output 7. The Contents Procedure within Jupyter Notebook

SAS PROCEDURES, USING PYTHON CODE

Now that the data is available within SAS, we can use Python code to source SAS procedures. A few procedures that are available include;

- SGPLOT
- PRINT
- MEANS
- CONTENTS

HEAD (PRINT PROCEDURE)

To source the PRINT procedure, the head function can be used. Use the data set name followed by the head function and within the brackets specify the number of observations you would like to view. This will perform a PROC PRINT of the observations you would like to view. By not specifying a number within the brackets, only the first five observations will be displayed.

```
sassearchdata.head(8)
```

The SAS System

Obs	age_range	datetime	gender	involved_person	legislation	object_of_search	officer_defined_ethnicity	outcome	outcome_link_to_object	removed_clothing	latitude	longitude
1	over 34	2017-06-02T13:25:00.000000	Male	True	None	Controlled drugs	Black	Suspect arrested	True	nan	.	.
2	over 34	2017-06-06T15:20:00.000000	Female	True	Police and Criminal Evidence Act 1984 (section 1)	Stolen goods	White	False	nan	False	51.6122	-0.78776
3	over 34	2017-06-25T00:48:00.000000	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	nan	False	51.7228	-1.20211
4	over 34	2017-06-23T11:00:00.000000	Male	True	None	Stolen goods	White	False	nan	nan	.	.
5	over 34	2017-06-26T12:05:00.000000	Male	True	None	Stolen goods	White	False	nan	nan	.	.
6	over 34	2017-06-15T09:28:00.000000	Male	True	Misuse of Drugs Act 1971 (section 23)	Controlled drugs	White	False	nan	False	51.6287	-0.75133
7	over 34	2017-06-04T13:45:00.000000	Male	True	None	Stolen goods	White	False	nan	nan	.	.
8	over 34	2017-06-06T20:20:00.000000	Male	True	None	Controlled drugs	White	False	nan	nan	.	.

Output 8. The PRINT procedure within Jupyter Notebook

DESCRIBE (MEANS PROCEDURE)

The MEANS procedure within SAS is used to generate summary statistics for any numeric variables within a SAS data set. The procedure allows users insight into the skewness of any data and identifies any variables which have missing values within the data set. The statistics produced by the procedure include:

- N
- NMiss
- Median
- Mean
- Standard Deviation
- Minimum
- 25th Percentile
- 50th Percentile
- 75th Percentile
- Maximum

To source the MEANS procedure, the describe function can be used. This will produce all of the available statistics for all numeric variables within the stop and search data set.

```
sassearchdata.describe()
```

The function produces the following SAS output:

The SAS System

The MEANS Procedure

Variable	N	N Miss	Median	Mean	Std Dev	Minimum	25th Pctl	50th Pctl	75th Pctl	Maximum
datetime	499	0	1813001460	1813109491	733649	1811896200	1812576540	1813001460	1813764780	1814481960
latitude	330	169	51.612171	51.644976	0.258232	51.332407	51.449778	51.612171	51.802931	53.949318
longitude	330	169	-0.798114	-0.941903	0.262051	-1.751645	-1.217150	-0.798114	-0.749093	-0.549811

Output 9. The MEANS procedure within Jupyter Notebook

THE SGPLOT PROCEDURE

The SGPLOT procedure is used to create plots or overlays on a set of axes. The procedure can be used to produce a range of plots including box plots, series plots, histograms, bar charts and heatmaps. The SAS SGPLOT procedure can be called using the bar or heatmap functions within SASPy.

Basic Bar Chart

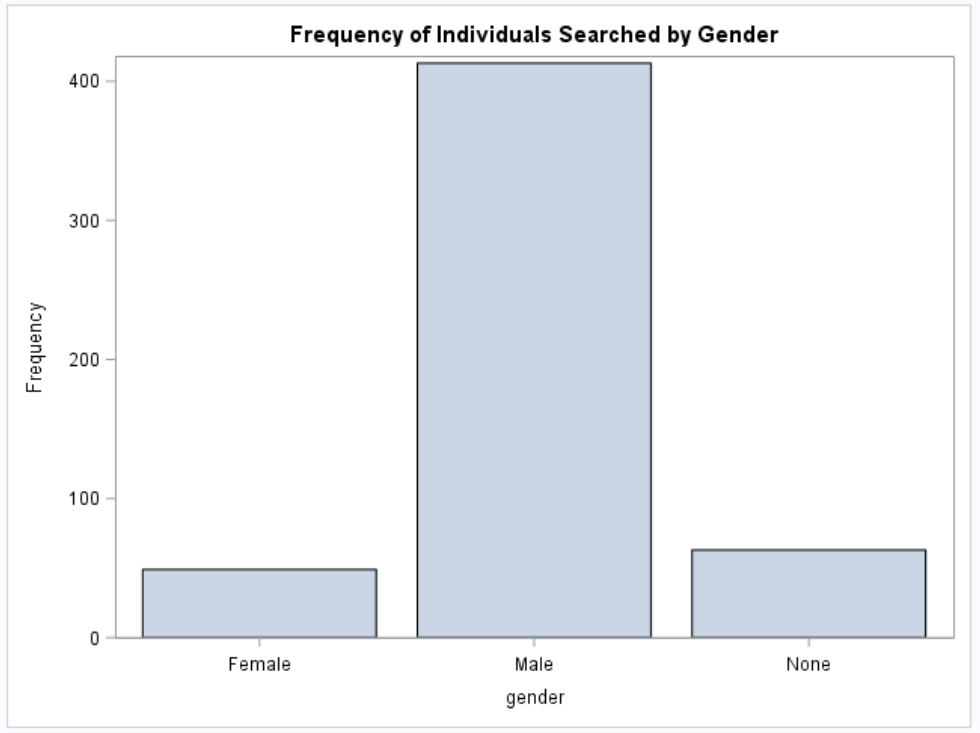
The bar function allows SAS to create bar charts based on the data using the SGPLOT procedure. The simple syntax for creating a bar chart using the bar function would be as follows:

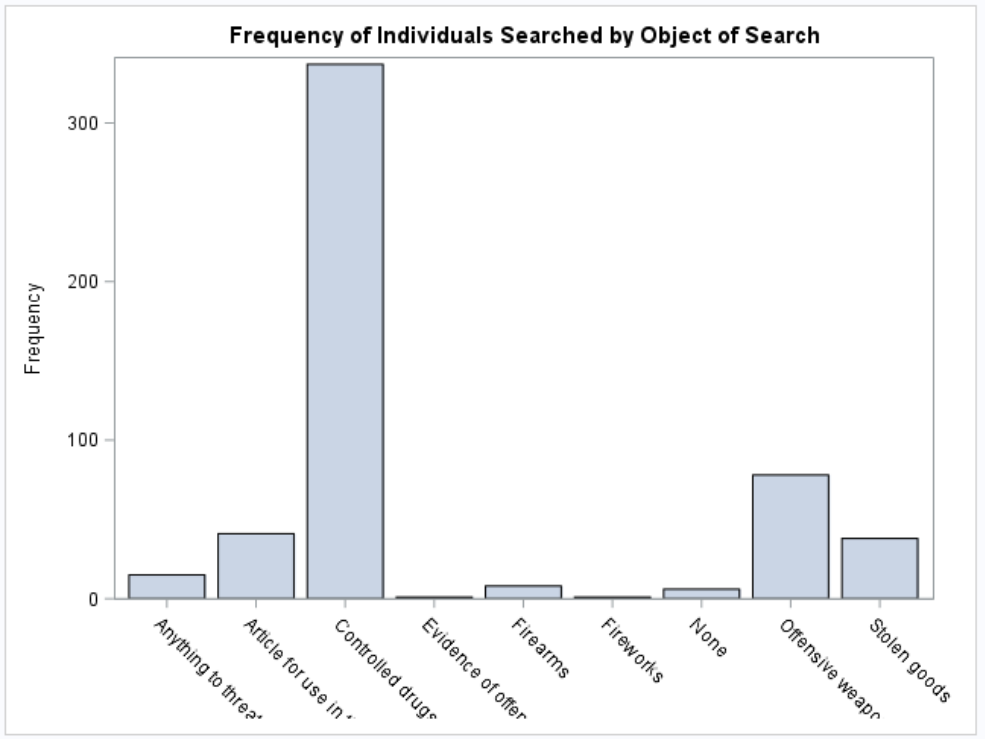
```
sassearchdata.bar('gender')
```

This code creates a simple bar chart displaying the gender values within the data set and the frequency of each value within the data. To create multiple bar charts two options are possible. Multiple occurrences of the bar function within the same code cell would allow two bar charts to be created.

```
sassearchdata.bar('gender', title='Frequency of Individuals by Gender')
sassearchdata.bar('object_of_search', title='Frequency of Individuals by
Object of Search')
```

This would produce one bar chart for the frequencies of the genders within the data set. In addition to this, a second bar chart would also be produced detailing the frequency of each object of search appearing within the data.





Output 10. Using the SGPLOT procedure within Jupyter Notebook to produce bar charts

An alternative way of coding this, would be to use a for loop. We could enter the two variable names into a for loop and use the variable as an argument to the bar function:

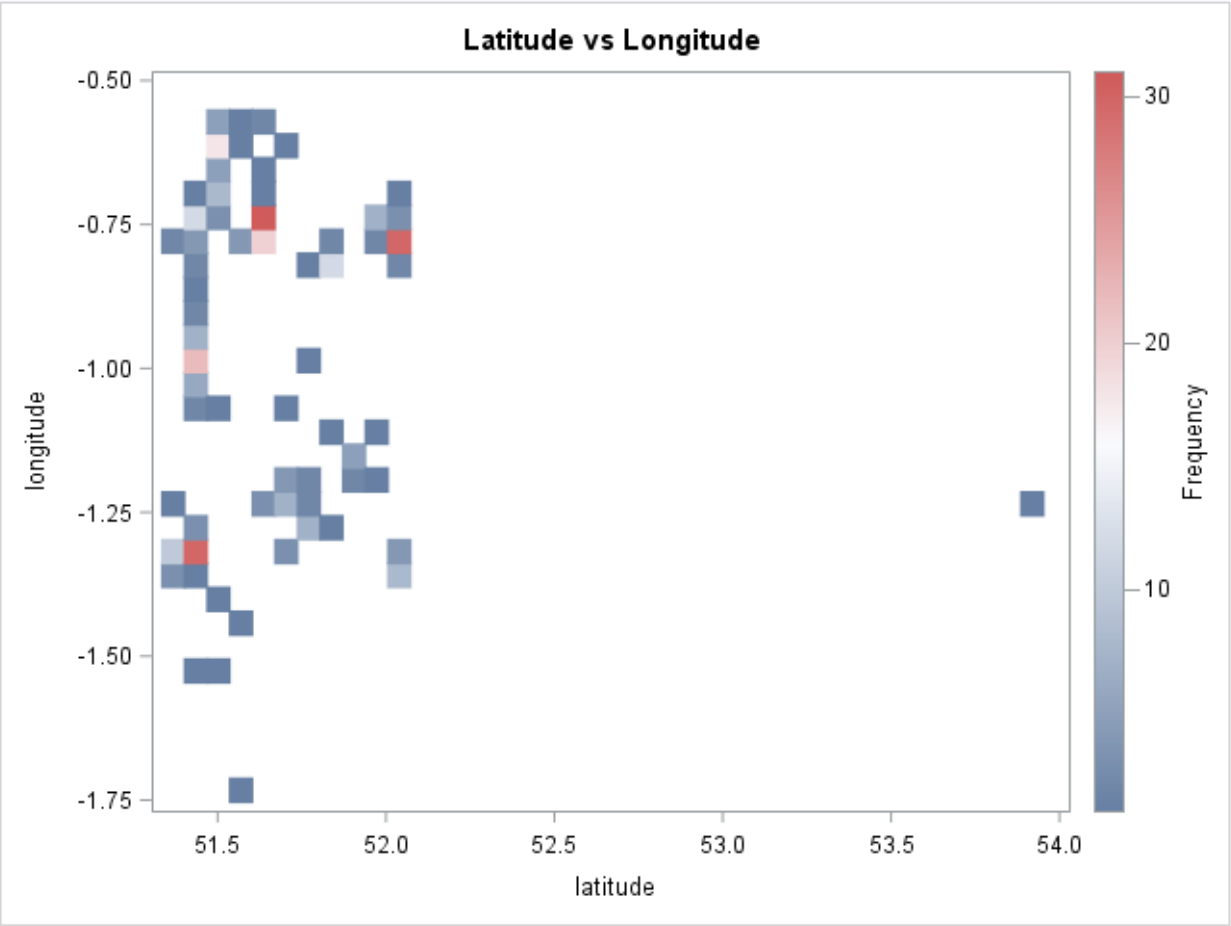
```
for col in ['gender', 'object_of_search']:
    sassearchdata.bar(col)
```

Basic Heatmap

The heatmap function also uses the SGPLOT procedure. Within the search data, we have readings which display the latitude and longitude of each stop and search. These values can be plot on a set of scales. A heatmap can be used to display this, with the intensity of the color determining the number of individuals which were stopped and searched within each area. To display a title for the heatmap, a title option can be added as an argument for the function after the variables have been specified. Using the gmaps module, it is possible to plot a heatmap over a Google Map to identify the locations of this data geographically.

```
sassearchdata.heatmap('latitude', 'longitude', title="Latitude vs
Longitude")
```

The heatmap displays once the code has run:



Output 11. Using the SGPLOT procedure within Jupyter Notebook to produce a heat map

TEACH ME SAS

For those who are not familiar with SAS programming, but are keen to learn the language, SAS have included a function which is bundled within SASPy named teach_me_SAS. It converts submitted Python SASPy code into the SAS language and displays this below the code cell. While the teach_me_SAS function is active, SAS will no longer run the code, but instead will give the SAS equivalent of any Python code run.

To activate teach_me_SAS the function needs to be assigned to the value of True:

```
sas.teach_me_SAS(True)
```

We can now run the same Python code as we used previously to display the SAS procedure steps which were generated by these functions.

HEAD (PRINT PROCEDURE)

On running the head function, SAS runs a Proc PRINT. With teach_me_SAS switched on, we can see that the head method generates a Proc PRINT step with the obs=10 data set option, corresponding to the number within the brackets of the function. If no number is specified, this defaults to the first five observations instead.

```
sassearchdata.head(10)
```

```
proc print data=mydata._df(obs=10 );run;
```

Display 2. Teach_me_SAS: The PRINT procedure

DESCRIBE (MEANS PROCEDURE)

In the previous section we ran a describe function on the sassearchdata data as below:

```
sassearchdata.describe()
```

By running the code with the teach_me_SAS function active, we can look at the SAS code which was run for this procedure. The is shown below:

```
sassearchdata.describe()
```

```
proc means data=mydata._df stackodsoutput n nmiss median mean std min p25 p50 p75 max;run;
```

Display 3. Teach_me_SAS: The MEANS Procedure

Proc MEANS generates default statistics including n, median, mean and standard deviation for all numeric variables. In addition to this, the STACKODSOUTPUT option has been added to the list of options enabling the Jupyter Notebook to resemble the default output from a Proc MEANS. In addition to this, the STACKODSOUTPUT option allows all the analysis variables to be stacked in one column and adds a new column for each statistic within the table.

THE SGPLOT PROCEDURE

On running the bar function, we can see that two Proc SGPLOTs are run. This produces two bar charts of the data using the vbar statement to define gender as the bars within the first bar chart and object_of_search for the second.

```
sassearchdata.bar('gender', title='Frequency of Individuals Searched by Gender')
sassearchdata.bar('object_of_search', title='Frequency of Individuals Searched by Object of Search')
```

```
proc sgplot data=mydata._df;
        vbar gender ;
        title "Frequency of Individuals Searched by Gender";
run;
title;
proc sgplot data=mydata._df;
        vbar object_of_search ;
        title "Frequency of Individuals Searched by Object of Search";
run;
title;
```

Display 4. Teach_me_SAS: The SGPLOT procedure – Bar Chart

The heatmap function also produces a Proc SGPLOT of the data. Within this procedure the first variable specified is assigned to the x axis with the second being assigned to the y axis. The equivalent code to complete this within SAS is displayed below the box.

```
sassearchdata.heatmap('latitude', 'longitude', title="Latitude vs Longitude")
```

```
proc sgplot data=mydata._df ;
        heatmap x=latitude y=longitude;
        title 'Latitude vs Longitude';
run;
title;
```

Display 5. Teach_me_SAS: The SGPLOT procedure - Heatmap

To run the SAS code again, the teach_me_SAS function will need to be set back to False:

```
sas.teach_me_SAS(False)
```

CONCLUSION

SAS and Python work in parallel using the SASPy module which can be exploited within the Jupyter Notebook to allow users more freedom to code in a language of their choice. By integrating the languages, those who are Python coders can harness the power of SAS without the need to learn the language. Those who would like to learn the language can take advantage of the teach_me_SAS function which converts any compatible Python coding into the SAS equivalent code.

REFERENCES

DATA.POLICE.UK (2017) *Police API Documentation*. [Online] DATA.POLICE.UK. Available from: https://data.police.uk/docs/ [Accessed 05/11/17].

HEMEDINGER, C (2017) Introducing SASPy: Use Python code to access SAS. [Weblog] *The SAS Dummy.* 8th April. Available from: http://blogs.sas.com/content/sasdummy/2017/04/08/python-to-sas-saspy/ [Accessed 02/08/17].

MURPHY, E. CRICK, T and DAVENPORT, J (2017) An Analysis of the Introductory Programming Courses at UK Universities. *The Art, Science, and Engineering of Programming.* [Online] 1 (18) Available from: https://arxiv.org/ftp/arxiv/papers/1609/1609.06622.pdf [Accessed 05/12/17].

SAS (2017) *Installation and* Configuration. [Online] GitHub. Available from: https://sassoftware.github.io/saspy/install.html#configuration [Accessed 05/03/18].

SMIT, D (2017) *Python grabs top honors as 2017's most popular language, Go growing rapidly*. [Online] NEOWIN. Available from: https://www.neowin.net/news/python-grabs-top-honors-as-2017s-most-popular-language-go-growing-rapidly [Accessed 05/12/17].

RECOMMENDED READING

SAS (2017) *A Python interface to MVA SAS.* [Online] GitHub. Available from: https://github.com/sassoftware/saspy [Accessed 05/12/17].

SAS (2017) *SAS Kernel for Jupyter.* [Online] GitHub. Available from: https://github.com/sassoftware/sas_kernel [Accessed 05/12/17].

SAS (2017) *SAS Pipefitter.* [Online] GitHub. Available from: https://github.com/sassoftware/python-pipefitter [Accessed 05/12/17].

SAS (2017) *SAS Scripting Wrapper for Analytics Transfer (SWAT).* [Online] GitHub. Available from: https://github.com/sassoftware/python-swat/ [Accessed 05/12/17].

CONTACT INFORMATION

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

Carrie Foreman
Amadeus Software Limited
+44 (0)1993 848010
carrie.foreman@amadeus.co.uk
www.amadeus.co.uk