Everything is Better with Friends

Using SAS in Python Applications with SASPy and Open-Source Tooling (Beyond the Basics)

Setup for Part 3

Getting setup to use Google Colab with SAS OnDemand for Academics (ODA)

1. Please enable line numbers using the Tools menu: **Tools** -> **Settings** -> **Editor** -> **Show line numbers** -> **Save**
2. To execute code examples, you'll need credentials for the following accounts:
    - Google. (If you're not already signed in, you should see a **Sign In** button in the upper right corner. You can also visit https://accounts.google.com/signup to create an account for free.)
    - SAS OnDemand for Academics. (You can create an account for free at https://welcome.oda.sas.com/ using an existing SAS Profile account. If you don't already have a SAS Profile account, you can create one for free using the "Don't have a SAS Profile?" link on the ODA login page.)
3. To save a copy of this notebook, along with any edits you make, please use the File menu: **File** -> **Save a copy in Drive**
4. We also recommend enabling the Table of Contents using the View menu: **View** -> **Table of contents**
5. Looking for "extra credit"? Please let us know if you spot any typos!

Connect to SAS OnDemand for Academics (ODA) and start a SAS session

Instructions:

1. Determine the Region for your ODA account by logging into https://welcome.oda.sas.com/. You should see the value `Asia Pacific`, `Europe`, or `United States` next to your username in the upper-right corner. (For more information about Regions, please see the ODA documentation.)

2. If your ODA account is associated with a Region other than `United States`, comment out Line 11 by adding a number sign (`#`) at the beginning of the line, and then do the following:
    - If your ODA account is associated with the Region `Europe`, uncomment Line 14 by removing the number sign (`#`) at the beginning of the line.
    - If your ODA account is associated with the Region `Asia Pacific`, uncomment Line 17 by removing the number sign (`#`) at the beginning of the line.
3. Click anywhere in the code cell, and run the cell using Shift-Enter.
4. At the prompt `Please enter the IOM user id`, enter either your SAS ODA user ID or the email address associated with your ODA account.
5. At the prompt `Please enter the password for IOM user`, enter the password for your SAS ODA account.

```
1 !pip install saspy
2
3 import saspy
4
5 sas = saspy.SASsession(
6     java='/usr/bin/java',
7     iomport=8591,
8     encoding='utf-8',
9
10    # The following line should be uncommented if, and only if, your ODA account i
11    iomhost = ['odaws01-usw2.oda.sas.com', 'odaws02-usw2.oda.sas.com', 'odaws03-us
12
13    # The following line should be uncommented if, and only if, your ODA account i
14    #iomhost = ['odaws01-euw1.oda.sas.com','odaws02-euw1.oda.sas.com'],
15
16    # The following line should be uncommented if, and only if, your ODA account i
17    #iomhost = ['odaws01-apse1.oda.sas.com','odaws02-apse1.oda.sas.com'],
18
19 )
20 print(sas)
```

```
Collecting saspy
  Downloading saspy-3.7.6.tar.gz (7.8 MB)
     |████████████████████████████████| 7.8 MB 21.4 MB/s
Building wheels for collected packages: saspy
  Building wheel for saspy (setup.py) ... done
  Created wheel for saspy: filename=saspy-3.7.6-py3-none-any.whl size=7858616 sh
  Stored in directory: /root/.cache/pip/wheels/b7/3c/56/3cec00a83001d0ffb2293f09
Successfully built saspy
Installing collected packages: saspy
Successfully installed saspy-3.7.6
Using SAS Config named: default
```

```
Please enter the IOM user id: matthew.t.slaughter@kpchr.org
Please enter the password for IOM user : ··········
SAS Connection established. Subprocess id is 128

Access Method         = IOM
SAS Config name       = default
SAS Config file       = /usr/local/lib/python3.7/dist-packages/saspy/sascfg.py
WORK Path             = /saswork/SAS_work34E5000155F4_odaws03-usw2.oda.sas.com/S
SAS Version           = 9.04.01M6P11072018
SASPy Version         = 3.7.6
Teach me SAS          = False
Batch                 = False
Results               = Pandas
SAS Session Encoding  = utf-8
Python Encoding value = utf-8
SAS process Pid value = 87540
```

Note: This establishes a connection from Python in Google Colab to a SAS session running in SAS ODA.

Install and import additional packages

```
# Install the faker module for generating fake data
!pip install faker

# Install the rich module for colorful printing
!pip install rich

# Initialize a Faker session called fake
from faker import Faker
fake = Faker()

# We'll use IPython to display DataFrames or HTML content
from IPython.display import display, HTML

# We'll use the pandas package to create and manipulate DataFrame objects
import pandas

# We're overwriting the default print function with rich.print
from rich import print
```

```
Collecting faker
  Downloading Faker-9.9.0-py3-none-any.whl (1.2 MB)
     |████████████████████████████████| 1.2 MB 30.9 MB/s
Requirement already satisfied: text-unidecode==1.3 in /usr/local/lib/python3.7/d
Requirement already satisfied: python-dateutil>=2.4 in /usr/local/lib/python3.7/
Requirement already satisfied: typing-extensions>=3.10.0.2 in /usr/local/lib/pyt
```

```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-package
Installing collected packages: faker
Successfully installed faker-9.9.0
Collecting rich
  Downloading rich-10.15.2-py3-none-any.whl (214 kB)
     |████████████████████████████████| 214 kB 27.3 MB/s
Requirement already satisfied: typing-extensions<5.0,>=3.7.4 in /usr/local/lib/p
Requirement already satisfied: pygments<3.0.0,>=2.6.0 in /usr/local/lib/python3.
Collecting colorama<0.5.0,>=0.4.0
  Downloading colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Collecting commonmark<0.10.0,>=0.9.0
  Downloading commonmark-0.9.1-py2.py3-none-any.whl (51 kB)
     |████████████████████████████████| 51 kB 6.5 MB/s
Installing collected packages: commonmark, colorama, rich
Successfully installed colorama-0.4.4 commonmark-0.9.1 rich-10.15.2
```

Part 3. Merging and appending datasets in SAS and Python

Section 3.1. Create class_df

```
# Let's start by importing and displaying the dataset sashelp.class from SAS ODA.

# Use the sas.sasdata2dataframe method to copy the contents of sashelp.class into
# class_df, and use dataset options to get only the first few rows and specific co
class_df = sas.sasdata2dataframe(
    table='class',
    libref='sashelp',
    dsopts={
        'keep': ['Name','Age','Height'],
        'obs': 7
    },
)

# Display the resulting DataFrame.
display(class_df)
```

	Name	Age	Height

Concept Check 3.1

- True or False: The DataFrame `class_df` is a data structure kept in memory in our local Google Colab session.

- *True.* *`sasdata2dataframe`* *imports data from a SAS dataset on disk in the remote ODA session into a* *`DataFrame`* *stored in memory in the local Colab session.*

6	Jane	12.0	59.8

Section 3.2. Create additional_students_df

```
# Now imagine we want to create additional example student records to append to sa
# we don't feel like being creative.

# Instead, let's use a "standard recipe" to have Python make a DataFrame with fake

# Note: This example uses more advanced Python features, including functions and l
# comprehensions. There's no need to focus on anything other than the overall inte
# DataFrame with mock data in it.

# Set the number of rows of mock data to generate
number_of_rows = 5

# Define a function that returns a fake first name that's not already in the Name
# DataFrame class_df defined above.
def create_distinct_first_name():
    random_first_name = fake.first_name()
    while random_first_name in class_df['Name'].unique():
        random_first_name = fake.first_name()
    return random_first_name

# Generate and display a DataFrame called additional_students_df with the specifie
additional_students_df = pandas.DataFrame(
  [
    {
        'Name': create_distinct_first_name(),
        'Age': fake.pyint(min_value=10,max_value=19),
        'Height': fake.pyfloat(right_digits=1,min_value=50,max_value=75),
    }
    for _
    in range(number_of_rows)
  ]
)
display(additional_students_df)
```

	Name	Age	Height
0	Kevin	12	72.5
1	Joy	10	72.1
2	Michele	17	51.3
3	Kayla	17	60.2
4	Michael	16	62.7

Concept Check 3.2

- True or False: `fake.first_name()` could have been used in place of `create_distinct_first_name()` on Line 25.
- Fun Fact: It's standard convention in the Python community to use an underscore (`_`) for a variable whose actual value isn't important

- *True, but using `fake.first_name()` directly instead of `create_distinct_first_name()` could introduce duplicate names.*
- *This article has some interesting additional examples: https://www.datacamp.com/community/tutorials/role-underscore-python*

Section 3.3. Create appended_df

```
# Now that we have two DataFrames with identical columns, we can vertically combin
# union) them to create a new, taller dataset.

# Starting with class_df, append each row from additional_students_df, and display
appended_df = class_df.append(additional_students_df, ignore_index=True)
display(appended_df)
```

	Name	Age	Height
0	Alfred	14.0	69.0
1	Alice	13.0	56.5
2	Barbara	13.0	65.3
3	Carol	14.0	62.8

Concept Check 3.3

- True or False: Deleting the option `ignore_index=True` on Line 5 won't affect the contents of the resulting DataFrame.
- *False. Deleting `ignore_index=True` would cause each row to retain its original index.*

Section 3.4. Create appended_sds

```
1  # TANGENT/ASIDE: We could have instead copied the DataFrame additional_students_df
2  # used PROC SQL to perform the union. Let's see how these two methods compare!
3
4  # Make a SAS dataset from additional_students_df.
5  sas.dataframe2sasdata(
6      additional_students_df,
7      table="additional_students_sds",
8      libref="Work"
9  )
10
11 # Use the sas.submit method to submit a PROC SQL step directly to ODA, and capture
12 # dict in sas_submit_return_value.
13 sas_submit_return_value = sas.submit(
14     '''
15         proc sql;
16             create table appended_sds as
17                 select * from sashelp.class
18                 union all corr
19                 select * from additional_students_sds
20             ;
21         quit;
22         proc print data=appended_sds;
23         run;
24     '''
25 )
26
27 # Output the SAS log, which corresponds to the key 'LOG' in the dict returned by s
28 sas_submit_log = sas_submit_return_value['LOG']
29 print(sas_submit_log)
```

```
30
31 # Render and display the SAS HTML results, which corresponds to the key 'LST' in t
32 # by sas.submit.
33 sas_submit_results = sas_submit_return_value['LST']
34 display(HTML(sas_submit_results))
```

```
24                                                    The SAS System
Monday, December  6, 2021 02:49:00 AM

152        ods listing close;ods html5 (id=saspy_internal) file=_tomods1
options(bitmap_mode='inline') device=svg style=HTMLBlue;
152      ! ods graphics on / outputfmt=png;
153
154
```

Concept Check 3.4

1. True or False: The SAS dataset `appended_sds` is a data structure kept in memory in our local Google Colab session.
2. Short Answer: List some others ways to vertically combine datasets in SAS.

```
164
```

1. *False.* `appended_sds` *is a pointer to a SAS dataset stored on disk in the remote ODA session.*
2. *Options include the SET statement in a DATA step or PROC APPEND.*

```
25                                                    The SAS System
```

Section 3.5. Create age_ranges_df

```
                                                      The SAS System
```

```
1  # Now suppose we want to add a column describing the age range for each student in
2  # DataFrame.
3
4  # We'll start by building a lookup table:
5
6  # As a first step, let's make a DataFrame with a single column called Age, using t
7  # function to populate this column with the integer values 10 through 19. (Note: T
8  # always stops at one less than the specified upper bound.)
9  age_ranges_df = pandas.DataFrame(
10     {
11         'Age': range(10,20)
12     }
13 )
14
15 # Now add a column to age_ranges_df by "binning" integers in age_ranges_df['Age']
16 # categories 10-12 (tween) and 13-19 (teen).
17
18 # In other words, use the pandas.cut method with these arguments:
19 # * bins=[10,12,19], which creates the values ranges [10,12] and (12,19]
20 # * labels=['tween','teen'], which specifies the label for each range, in order
21 age_ranges_df['Age_Range'] = pandas.cut(
22     age_ranges_df['Age'],
23     bins=[10,12,19],
24     labels=['tween','teen'],
25     include_lowest=True,
```

```
26 )
27 display(age_ranges_df)
```

	Age	Age_Range
0	10	tween
1	11	tween
2	12	tween
3	13	teen
4	14	teen
5	15	teen
6	16	teen
7	17	teen
8	18	teen
9	19	teen

Concept Check 3.5

1. True or False: Deleting the option `include_lowest=True` on Line 25 won't affect the contents of the resulting DataFrame.
2. True or False: The `pandas.cut` method could have instead been used directly on DataFrame `appended_df` to bin values of `Age`.

1. *False. Deleting `include_lowest=True` will cause the value 10 to not have a corresponding label.*
2. *True. We could apply `pandas.cut` to any numeric column in any DataFrame. However, if the values in the column aren't in any of the corresponding bins, the results might not be very interesting.*

Section 3.6. Create merged_df

```
1 # Given the lookup table age_ranges_df, which has some (but not all) of its column
2 # appended_df, we can horizontally combine (aka merge or join) them to create a ne
3
4 # Specifically, starting with appended_df, we'll add an Age_Range column whose val
5 # by matching values of Age in age_ranges_df as part of a left join, resulting in
6 # the same height as appended_df.
7 merged_df = appended_df.merge(
8     age_ranges_df,
```

```
    on='Age',
    how='left',
)

# Since we're not using the Height column, drop it from merged_df, and display the
merged_df.drop(columns=['Height'], inplace=True)
display(merged_df)
```

	Name	Age	Age_Range
0	Alfred	14.0	teen
1	Alice	13.0	teen
2	Barbara	13.0	teen
3	Carol	14.0	teen
4	Henry	14.0	teen
5	James	12.0	tween
6	Jane	12.0	tween
7	Kevin	12.0	tween
8	Joy	10.0	tween
9	Michele	17.0	teen
10	Kayla	17.0	teen
11	Michael	16.0	teen

Concept Check 3.6

- True or False: Deleting the option `inplace=True` on Line 14 won't affect the contents of the resulting DataFrame.

- *False. The column will not be dropped successfully unless you use `inplace=True`, or unless you assign the result `merged_df.drop(columns=['Height'])` to a variable.*

Section 3.7. Create merged_sds

```
# TANGENT/ASIDE: Since the DataFrame additional_students_df was already copied ove
# above, we could have instead copied over age_ranges_df and used PROC SQL to perf
# Let's see how these two methods compare!

# Make a SAS dataset from age_ranges_df.
sas.dataframe2sasdata(
```

```
    age_ranges_df,
    table="age_ranges_sds",
    libref="Work"
)

# Use the sas.submit method to submit a PROC SQL step directly to ODA, and capture
# dict in sas_submit_return_value.
sas_submit_return_value = sas.submit(
    '''
        proc sql;
            create table merged_sds as
                select
                     A.Name
                    ,A.Age
                    ,B.Age_Range
                from
                    appended_sds as A
                    left join
                    age_ranges_sds as B
                    on A.Age = B.Age
            ;
        quit;
        proc print data=merged_sds;
        run;
    '''
)

# Output the SAS log, which corresponds to the key 'LOG' in the dict returned by s
sas_submit_log = sas_submit_return_value['LOG']
print(sas_submit_log)

# Render and display the SAS results HTML, which corresponds to the key 'LST' in t
# by sas.submit.
sas_submit_results = sas_submit_return_value['LST']
display(HTML(sas_submit_results))
```

Concept Check 3.7

1. True or False: The SAS dataset `age_ranges_sds` is a data structure kept in memory in our local Google Colab session.
2. Short Answer: List some others ways to get the same result in SAS without a join.

1. *False.* *`age_ranges_sds`* *is a pointer to a SAS dataset stored on disk in the remote ODA session.*
2. *DATA step MERGE or UPDATE statements, or create age ranges using PROC FORMAT.*

Section 3.8. Additional Exercises

For practice, we recommend the following:

1. Run the code cell below to convert the `Height` column of `class_df` to integer values.
2. Repeat the steps in Sections 3.5-6, creating `height_ranges_df` and then merging it with `class_df` on `Height`. (Alternatively, you could try applying `pandas.cut` directly to `class_df`).

```
class_df['Height'] = class_df['Height'].apply(int)
display(class_df)
```

	Name	Age	Height
0	Alfred	14.0	69
1	Alice	13.0	56
2	Barbara	13.0	65
3	Carol	14.0	62
4	Henry	14.0	63
5	James	12.0	57
6	Jane	12.0	59

```
# Create a range of heights
height_ranges_df = pandas.DataFrame(
    {
        'Height': range(50,75)
    }
)

# Bin height
height_ranges_df['Height_Range'] = pandas.cut(
    height_ranges_df['Height'],
    bins=[50,62,75],
    labels=['short','tall'],
    include_lowest=True,
)
display(height_ranges_df)

merged_height_df = class_df.merge(
    height_ranges_df,
    on='Height',
```

```
    how='left',
)

# Since we're not using the Age column, drop it from merged_df, and display the re
merged_height_df.drop(columns=['Age'], inplace=True)
display(merged_height_df)
```

	Height	Height_Range
0	50	short
1	51	short
2	52	short
3	53	short

```
class_df['Height_Range'] = pandas.cut(
    class_df['Height'],
    bins=[50,62,75],
    labels=['short','tall'],
    include_lowest=True,
)
display(class_df)
```

	Name	Age	Height	Height_Range
0	Alfred	14.0	69	tall
1	Alice	13.0	56	short
2	Barbara	13.0	65	tall
3	Carol	14.0	62	short
4	Henry	14.0	63	tall
5	James	12.0	57	short
6	Jane	12.0	59	short
7	67		tall	

Notes and Resources

Want some ideas for what to do next? Here are our suggestions:

1. For more about the `faker` package, which can generate many other types of fake data, see https://faker.readthedocs.io/
2. For more about the `pandas` package, including the methods used above, see the following:
    - https://pandas.pydata.org/docs/reference/api/pandas.cut.html
    - https://pandas.pydata.org/docs/reference/api/pandas.unique.html
    - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.append.html
    - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html
    - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html

3. For more about the `saspy` package, including the methods used above, see the following:
    - https://sassoftware.github.io/saspy/api.html#saspy.SASsession.dataframe2sasdata
    - https://sassoftware.github.io/saspy/api.html#saspy.SASsession.sasdata2dataframe
    - https://sassoftware.github.io/saspy/api.html#saspy.SASsession.submit
4. For more about some of the Python features used, such as functions and list comphrensions, we recomend the following chapters of A Whirlwind Tour of Python:
    - https://jakevdp.github.io/WhirlwindTourOfPython/08-defining-functions.html
    - https://jakevdp.github.io/WhirlwindTourOfPython/11-list-comprehensions.html
5. We welcome follow-up conversations. You can connect with us on LinkedIn or email us at isaiah.lankham@gmail.com and matthew.t.slaughter@gmail.com
6. If you have a GitHub account (or don't mind creating one), you can also chat with us on Gitter at https://gitter.im/saspy-bffs/community